Error Detection and Recovery in Spoken Dialogue Systems

Edward Filisko and Stephanie Seneff *

Spoken Language Systems Group
MIT Computer Science and Artificial Intelligence Laboratory
200 Technology Square, Cambridge, MA 02139
{filisko,seneff}@csail.mit.edu

Abstract

This paper describes our research on both the detection and subsequent resolution of recognition errors in spoken dialogue systems. The paper consists of two major components. The first half concerns the design of the error detection mechanism for resolving city names in our MERCURY flight reservation system, and an investigation of the behavioral patterns of users in subsequent subdialogues involving keypad entry for disambiguation. An important observation is that, upon a request for keypad entry, users are frequently unresponsive to the extent of waiting for a time-out or hanging up the phone. The second half concerns a pilot experiment investigating the feasibility of replacing the solicitation of a keypad entry with that of a "speak-and-spell" entry. A novelty of our work is the introduction of a speech synthesizer to simulate the user, which facilitates development and evaluation of our proposed strategy. We have found that the speak-and-spell strategy is quite effective in simulation mode, but it remains to be tested in real user dialogues.

1 Introduction

Spoken dialogue systems are emerging as an intuitive interface for providing conversational access to online information sources (Eckert et al., 1997; Gorin et al., 1997; Dahlback et al., 1999; Zue et al., 2000; Walker et al., 2001; Glass and Seneff, 2003; Pieraccini et al., 1997; Quast et al., 2003; J. Gustafson, 1999; Polifroni and Chung, 2002; Denecke, 2002; Seneff, 2002; Zue and Glass, 2000). While the effectiveness of such systems has improved significantly over the past several years, a critical barrier to widespread deployment remains in the form of communication breakdown at strategic points in the dialogue, often when the user is trying to convey a critical piece of information that the system repeatedly misunderstands.

This paper focuses on the specific two-stage problem of error detection and subsequent recovery, in a situation where the user is attempting to provide a named entity which the system fails to understand. It is not a straightforward process for the system even to notice that it has made a mistake. Tedious confirmation subdialogues for every attribute provided would lead to annoyance and widespread unwillingness to use the system at all. Hence, the system should only invoke a confirmation subdialogue when it perceives there to be a communication breakdown.

The second aspect of the problem, error recovery, is also challenging. The system may persist in misunderstanding repeated spoken renditions of the same named entity, unless a substantially different tactic can be adopted to assure higher communicative success.

The remainder of the paper is organized as follows. Section 2 motivates why we think this is an interesting and important problem. In Sections 3 and 4, we describe the error detection and recovery strategies that have been adopted in our MERCURY flight reservation system (Seneff, 2002; Seneff and Polifroni, 2000), and we provide an analysis of the degree to which error recovery was successful, specifically for the case of entering a source or destination city name. The approach used was to solicit a keypad entry of the city in cases where the system detected a communication breakdown. We have analyzed a set of 172 cases where keypad entry of a city was solicited. One of the observations was that users were often not very receptive to the idea of switching into keypad mode to map the spelling of the city to a numeric code. Whether this is the result of cognitive overload, confu-

* This research was supported by an industrial consortium supporting the MIT Oxygen Alliance.

sion, or some other reason is not clear, however, since we were unable to interview users to identify why they chose not to use the keypad.

Motivated by the apparent need for a more intuitive error recovery strategy, we describe in Sections 5 and 6 a set of experiments that explore an alternative approach whereby the user is instead asked to speak and spell the problematic city name. We have recently developed the capability to utilize a pronounced version of a word to greatly enhance the accuracy of a letter recognition task, and have successfully integrated this technology into a personal name enrollment task (Seneff et al., 2003; Chung et al., 2003). Our interest here was in evaluating whether a similar technique would be useful for the error recovery problem.

It is difficult, however, to develop and perfect an algorithm involving multiple recognition passes, that is only triggered sporadically in user conversations. Hence, we discuss a novel approach to system development based on simulating the completion of user dialogues beginning at the point where the system had detected a communication breakdown. In other words, we utilize a speech *synthesizer* to produce a speak-and-spell waveform that is solicited in lieu of the keypad entry in the MERCURY dialogues we have analyzed. DECTalk[1] acts as a user continuing the conversation from the point where the original MERCURY system detected communication breakdown. The synthetic speech is processed through the speak-and-spell recognition system. An analysis of the rate of success is then an indicator of how promising the method might be for real user dialogues. At the same time, we have expanded the set of cities from the original 500 known to the MERCURY system to a much larger set of nearly 17,000 city names within the United States. If successful, this new speak-and-spell mode would thus greatly expand the number of cities that the system could theoretically recognize.

2 Background and Motivation

Spoken conversational systems have great potential for providing spoken language access to information sources. However, such systems are only useful if they can understand most of the content words in each utterance. Many a user has been aggravated by conversational systems that hypothesize the same incorrect words over and over, often due to ignorance of critical words the user is speaking. The dialogue manager component is often of little or no help, suggesting, "Please try using shorter sentences," or "I did not understand. Please rephrase your query." The system must be able to recognize complications such as misrecognitions, repetitive mistakes, and out-of-vocabulary (OOV) words, and to react appropriately. A more successful interaction is achievable if the dialogue manager is able to work with the user to resolve an error in the course of the dialogue.

A common strategy used in human-human dialogue for handling an OOV word is for one participant to ask the other to repeat or spell the unknown word. This provides the inquisitor with more detailed information about the word, which can facilitate his understanding of the word. Several dialogue systems have employed such a strategy in dealing with a confusing or unknown word (Bauer and Junkawitsch, 1999; Schramm et al., 2000). We aim to employ such a speak-and-spell strategy in our system.

In this work, we focus on the class of place names, specifically cities, states, and airports. Such names are prevalent and problematic in any domain where geography plays a dominant role. For example, a weather information system or a flight reservation system must have the city, state, or country names exactly correct in order to be useful. In real user interactions, it is inevitable that some city will be mentioned that is unknown to the system. Such a name will often be misrecognized as a known city. The dialogue manager must determine whether a poorly scoring hypothesized city name is, in fact, a misrecognized but *known* city, or an entirely new word. Such uncertainty must be resolved in a manner that is time-efficient and does not overburden the user.

For example, in the case of a weather information system, very common geographic names would most likely be known to the recognizer (e.g., *New York*, *Moscow*). If a user wants to know the weather in his hometown of Menominee, for example, which is unknown to the recognizer, the system is faced with a problem. If the recognizer is not equipped with an unknown word detector, the closest word in the recognizer's vocabulary will be chosen as the best hypothesis. The user will then have to proceed through potentially many clarification turns in which the system repeatedly hypothesizes incorrect city names. Since "Menominee" is OOV, the system will never find the correct city name.

If, however, a large external database is available (e.g., US Census data), that the system could consult given a hypothesized spelling of the unknown city, there is hope that the system will find the user's intended city name. Such an approach of access to a much larger vocabulary in spell mode has been successfully applied, for example, in (Schramm et al., 2000). In the most extreme case, the large external database would be the World Wide Web. One could imagine the system posing a search query on the Web to determine that the closest major city is Green Bay, Wisconsin, located 60 miles south of Menominee, so it may suffice to provide the weather for Green Bay.

It should be mentioned that the flexibility of allowing the user to spell a word creates a new task for the system. Instead of simply looking up a hypothesized word in a

[1] http://research.compaq.com/wrl/decarchives/dtj/dtjk01/

User:	I would like to fly to austin tomorrow morning [mis-recognized as “Boston”]
System:	Sorry. I’m not sure what you said. There are no flights from Boston to Boston.
User:	austin texas
System:	Okay, flights from Austin to Boston departing between 5:00 am and noon on Monday March tenth. One moment please. If I have misunderstood you, you should say, Scratch that. There are no nonstop flights. I have flights on American, Continental, and Delta. There are at least 6 flights departing between 8:00 a.m. and 9:00 a.m. Can you be more specific?
User:	I would like to fly from boston massachusetts to austin texas
System:	Please enter the departure city name using the telephone keypad ending with the pound key.
User:	267866# [user enters “Boston” via keypad]
System:	Okay from Boston to Austin. What date will you be traveling?
USER:	I would like to leave tomorrow morning
System:	Okay, flights from Boston to Austin departing between 5:00 am and noon on Monday March tenth. One moment please. . . .

Figure 1: Example of a dialogue involving a successful entry of a departure city using the telephone keypad, initiated by the system based on perceived confusion.

SYSTEM:	Welcome Angie. How may I help you?
USER:	I would like to <uh> [mis-recognized as: I would like to Atlanta]
SYSTEM:	Okay from Hartford to Atlanta. What date will you be traveling?
USER:	October thirty first.
SYSTEM:	Okay, from Hartford to Atlanta on Wednesday October 31. Can you provide the approximate departure time or airline?
USER:	Late morning.
SYSTEM:	Okay, flights from Hartford to Atlanta departing between 10:00 a.m. and noon on Wednesday October 31. One moment please... I have a nonstop Delta flight leaving at 11:20 a.m. and arriving at 1:45 pm. Would that work?
USER:	No.
SYSTEM:	There are no nonstop flights. I have a U S Airways flight connecting in Charlotte leaving at 10:45 a.m., a United flight connecting in Washington Dulles leaving at 11:20 a.m., and a U S Airways flight leaving at noon. Please select one of these flights or change any constraint you have already specified.
USER:	I would like to change the arrival city.
SYSTEM:	Please enter the destination city using the telephone keypad ending with the pound key.
USER:	6752636# [user enters “Orlando” via keypad]
SYSTEM:	I heard you say you want to arrive in Orlando. Is that correct? If you need help, you can say, for example, what cities do you know in Oregon.
USER:	That is correct.

Figure 2: Example of a dialogue involving a successful entry of an arrival city using the telephone keypad, provoked by a specific request by the user to change the arrival city.

lexicon, the system must now remain open to the possibility that the hypothesized word (i.e., sequence of letters) is misspelled. In other words, a spelling checker must be utilized. Depending on the robustness of the spelling checker, the system may or may not be able to identify the intended word from a misspelling. With a large database, a potential misspelling could correspond to multiple data items. In such a case, the system would need to initiate a disambiguation subdialogue to resolve the ambiguity.

There are several pragmatic issues to consider in obtaining spelled data from a user whether via keypad or speech. The problem of disambiguating keypad sequences has been addressed using both dictionary-based (Davis, 1991) as well as probabilistic (MacKenzie et al., 2001) approaches. In both input modes, the user may use abbreviations such as “S T P E T E R S B U R G” for “Saint Petersburg”. Spoken spelling is especially difficult, because the recognition accuracy for spoken letters can be quite low. For instance, the members of the “E-set” (B, C, D, E, G, P, T, V, Z) are well-known for being confusable to a recognizer, as discussed in previous studies (Marx and Schmandt, 1994). This problem is compounded by the fact that humans spell words in creative ways. Some may spell in military style (e.g., “Alpha Bravo Charlie” for “A B C”) or in simile (e.g., “B as in ‘Boy”’). Some users may include the word “space” to mark the word boundaries of a multi-word sequence, such as “N E W space Y O R K”. Some may simply enter a letter sequence containing several meaningful chunks, as in “N E W Y O R K J F K N Y C” for Kennedy Airport in New York City. Many of these issues have been addressed in (Schramm et al., 2000).

3 MERCURY Error Recovery Strategy

The MERCURY system, accessible via a toll free number[2], provides information about flights available for over 500 cities worldwide. We have invested considerable effort into making MERCURY intuitive to use and robust in handling a wide range of different ways users might express their flight constraints and select the flights of the itinerary. A typical user begins by logging on, providing both his name and password, which allows the system to look up some personalized information such as the e-mail address and the preferred departure city. MERCURY's dialogue plan involves arranging a trip one leg at a time. Once the itinerary is fully specified, MERCURY offers to price the itinerary and, subsequently, to send a detailed record of the itinerary to the user via e-mail, which can then be forwarded to a travel agent for the actual booking.

A critical aspect of flight dialogues is the successful communication of the source, destination, and date, all of which are susceptible to recognition error. MERCURY's default policy is to use implicit confirmation to communicate to the user its interpretation of his utterances. In the meantime, it monitors the evolution over time of these three critical attributes. When it detects odd behavior, it switches into a mode where keypad entry is solicited. The keypad entry is matched against existing hypotheses and, if a successful match is obtained, is assumed to be correct. Otherwise, a verbal confirmation subdialogue, soliciting a "yes/no" answer, is invoked.

For source and destination, the system tabulates at each turn whether the attribute was inherited, repeated, or changed. If a change is detected after flights have already been retrieved, the system prompts for spoken confirmation of the surprising move, anticipating possible recognition error. After two consecutive turns where the user has either apparently repeated or replaced the departure or arrival city, the system requests the user to enter the city by spelling it using the telephone keypad. This strategy is also used if a substitution/repetition of the city is followed by an utterance that is not understood, or whenever the user explicitly requests to change the departure or arrival city. It turns out that MERCURY's 500 cities are uniquely identifiable through their keypad codes; however, if this were not the case, a follow-up disambiguation subdialogue could be arranged. This keypad mechanism also provides the opportunity to confirm whether the desired city is known or unknown.

A similar process takes place for dates. If the user appears to repeat the date, without providing any other information, there is the suspicion that a misrecognized date has again been misrecognized the same way. In this case, the system tries to find an alternative hypothesis for the date by re-examining the N-best list of recognizer hypotheses, and, in any case, also asks for user confirmation. As is the case for cities, the system invokes the keypad upon repeated date corrections.

Figures 1 and 2 provide two examples of user dialogues involving keypad city entry. Figure 1 illustrates a dialogue where the conversation is clearly confused, and the system eventually takes the initiative to invite a keypad entry of the departure city. The user wanted to go to "Austin", which the system misunderstood as "Boston". This particular user had a default departure city of "Boston", which caused the system to suppose that the user had requested a pragmatically unreasonable flight from "Boston" to "Boston". The user's follow-up fragment, "Austin, Texas", was correctly understood, but misinterpreted as the departure city instead of the arrival city, leading to further confusion. It was only after the user had cleared up the confusion, with the complete utterance, "I would like to fly from Boston, Massachusetts to Austin, Texas," that the system was finally on the right track, but by this point, it had identified difficulty with the source, reacting by launching a keypad entry request, with subsequent resolution.

Figure 2 shows an example subdialogue where the destination city was successfully entered using the telephone keypad, based on an explicit request on the part of the user to change the destination. Interestingly, the user delayed the correction until the system invited him to change any constraint that was already specified. This particular user probably believed that she was required to respond to the prompts, although it is conceivable that the user's delayed response was due to inattentiveness. This dialogue thus reveals some of the potential difficulties encountered due to users' false assumptions about the system's behavior.

4 MERCURY Analysis

We have been collecting MERCURY data over the telephone for the past several years (Seneff and Polifroni, 2000), involving user interactions with the system to make flight reservations. In examining these dialogues, we have come to the realization that, while keypadding the date (as a four digit numeric code for month and day) seems to be intuitive to users and therefore an effective mechanism for correcting misunderstandings, the situation is far less effective in the case of city names.

A detailed analysis has thus been performed on all instances where the system requested a source or destination entry via the keypad, and the user's reactions to the requests were observed and quantified. We found that this strategy, when users were compliant, was generally successful for determining the user's desired source or destination. For example, if the user were to enter "3387648", the system would understand "DETROIT", and the dialogue would smoothly continue.

[2] 1-877-MIT-TALK.

In addition to many successful responses, however, several errorful responses were also observed, including misspelled words (e.g., "TEIPEI" for "TAIPEI"), out-of-vocabulary words (e.g., "DOMINICA"), or a string of valid references that could not be resolved as a single place name (e.g., "NEWYORKJFKNYC" for "New York's Kennedy Airport"). A user time-out or hang-up was also common, and constituted a significant number of responses.

A total of 172 instances were observed in which the system prompted users to enter a source or destination via the keypad. The number of occurrences is rather low since this solicitation was only activated as a last resort. The system then entered a state where speech was not an option. The users' responses to these prompts are summarized in Table 1. Most surprising is that nearly half of the time, the user did not even attempt to use the keypad. In only 88 of the cases did the user actively enter a keypad code. The user let a time-out occur in 50 cases, and hung up the telephone in an additional 34 cases.

Description	*Count*	*Percentage*
user attempts at keypad entry	88/172	51.1%
time-outs	50/172	29.1%
hang-ups	34/172	19.8%

Table 1: Summary of users' responses to 172 system prompts to enter a source or destination using the telephone keypad.

This attempt rate of 51.1% is significantly lower than originally hoped. Even within the 88 compliant cases, the results are disappointing, as shown in Table 2. In 61 cases, the keypad sequence entered by the user corresponded to a valid city or airport name. Most of these were known to the system and were processed successfully. The remaining 30.7% of attempts consisted of misspellings (such as a double tap on a key, substituting the number '0' for the letter 'o', or terminating with '*' instead of '#') or apparent garbage.

Description	*Count*	*Percentage*
valid city/airport entered	61/88	69.3%
misspelled city/airport entered	19/88	21.6%
garbage entered (e.g, "***#")	8/88	9.1%

Table 2: Summary of a total of 88 user attempts at entering a source or destination city or airport name using the telephone keypad after being prompted by the system.

4.1 Discussion

Our results suggest that the strategy of prompting for keypad entry of questionable parameters shows potential for recovering from situations in which the system is confused about what the user has said. We believe that such recovery can contribute to successful dialogue completion, as well as elevating the user's tolerance level. Nevertheless, our results also pose two questions that need to be addressed: why do some users' attempts at keypad entry contain errors, and, more importantly, why do some users not even attempt keypad entry?

It is not possible to know why an individual user was unable to enter a valid keypad sequence; we had no mechanism to interview users about their behavior. We can, however, speculate that the errorful sequences were due to the non-intuitive nature of spelling with a telephone keypad, a user's unfamiliarity with the spelling of a given word, typos, or a user's confusion as to what qualified as an acceptable entry (e.g., Are abbreviations and nicknames allowed?).

We must also acknowledge the fact that what qualifies as a valid keypad sequence depends on the spelling correction capabilities of the system. Even a simple spelling checker (not utilized during the MERCURY data collection) could potentially allow the system to make sense of an errorful keypad sequence.

In the case of a time-out, it is difficult to know what each user was thinking as he waited. It is likely that the user was hoping for a return to speech mode after the time-out. The user may have hesitated for fear of sending the system down an even more divergent path. It is also possible that users were inattentive when the system instructed them to terminate with the pound key, and that they therefore entered the entire city, but without a termination code. Clearly a strategic modification to automatically generate a '#' after a significant pause might help reduce this type of error.

The reason for a hang-up is more obvious, given the dialogue context. For example, if the user had repeatedly said that he wanted to fly to Anchorage and the system had already hypothesized three other cities, it is understandable that he would have hung up in frustration.

The telephone keypad would seem to be a very practical mode of information entry given its physical accessibility and limited ambiguity per key. This small set of data in the flight domain, however, suggests that it is confusing, annoying, or simply intimidating to many users. The next challenge, then, is to utilize a similar error recovery strategy, but to adopt a different mode of information entry, one that is more intuitive and less intimidating. We discuss such an option in the next section.

5 Spoken Spelling

Allowing a user to spell a word has several benefits, including maintaining a single mode of communication (i.e., speech), as well as being less taxing, more efficient,

and more intuitive. Our goal is to make the user feel confident that spelling a city name is a plausible request and that it *can* be the most effective path to task completion.

Undeniably, spelling recognition comes with its own set of problems, especially misrecognition of the spoken letters. One way to minimize such errors is to incorporate limited spelling checking, such as allowing a single insertion, deletion, or substitution per word. For example, a spelling sequence recognized as "T E N V E R" could be mapped to "D E N V E R" as the closest match in the database. Obviously, a trade-off exists where overgenerous spelling correction could lead to a false hypothesis.

A great challenge in developing conversational systems is that dialogue strategies can only evolve through extensive experimentation, which requires a large amount of data, particularly for situations that occur rarely in actual dialogues. To expedite development and evaluation of the recovery strategy, we decided to make use of simulated user data to artificially continue MERCURY dialogues beyond the point where the system had originally asked for a keypad entry, as described in the next section.

6 User Simulation

To streamline exploration of alternative dialogue strategies for error recovery, we have implemented a simulated user that speaks and spells a city name using DECTalk. A block diagram of our simulated user system is shown in Figure 3. Each synthesized waveform[3] contains a pronunciation of the city name that a user was trying to communicate in the original dialogue, immediately followed by a spoken spelling of that city name (e.g., "Boston B O S T O N"). The waveform is passed to a first stage speech recognizer, which treats the spoken word as an unknown word and proposes an N-best list of hypothesized spellings for the synthesized letter sequence. For speech recognition, we use the SUMMIT framework (Glass et al., 1996), and the unknown word is modeled according to techniques described in (Bazzi and Glass, 2002).

Following the first stage recognition, a two-stage matching process first consults a list of "cities in focus" that were extracted as hypotheses from the original user's final utterance before the keypad turn. Subsequently, if a match or conservative partial match is not found from the short list, a large database of 17,000 city and state names is consulted for a match or a partial match. In this case a confirmation subdialogue ensues.

If a match is found, a geography server determines whether the name is ambiguous. If so, a disambiguating item (e.g., state name) is requested by the dialogue manager. The simulated user then randomly chooses from a list of candidate state names provided by the geography server. This utterance is currently also processed as a speak-and-spell utterance, mainly because we are interested in obtaining more data on the performance of our speak-and-spell system.

If no match is found in either the short list or the external lexicon of known city names, another recognition cycle is initiated, in which the phonetic content of the spoken word is used to enhance the performance of the spelling recognizer, following procedures described in (Chung et al., 2003). A letter-to-sound model is used to map from a graph of letter hypotheses proposed by the first stage recognizer to their corresponding plausible pronunciations, using techniques described in (Seneff et al., 1996). The final set of hypotheses is obtained by merging hypotheses produced from both halves of the user utterance. Once again, both the short list and the large lexicon are searched for a match.

The idea is that this second stage should only be invoked upon failure, in order to reduce the amount of computation time required. An alternative strategy would be for the system to unconditionally execute a second recognition to obtain a potentially more correct hypothesis. Such a strategy, however, would increase the system's overall processing time.

6.1 Results and Discussion

The simulation was performed on a total of 97 user utterances, all of which MERCURY had designated as trouble situations in the original dialogues. The utterances utilized are those for which the system's hypotheses contained city names, whether or not the user had actually mentioned a city name.

The simulation results are shown in Table 3. Out of 97 problematic sources and destinations generated by the simulated user, 58 required disambiguation with a state name (e.g., "Boston in Georgia"). Therefore, 155 speak-and-spell utterances were ultimately passed through the synthesize-and-recognize simulation cycle. All but one of the state names were correctly recognized. This high performance is likely due to the correct state's guaranteed existence in the short list used by the spelling checker.

Our algorithm dictates that a second pass, which integrates the spoken name portion of the waveform with letter-to-sound hypotheses derived from the spoken spelling portion, be omitted if a match is found in the first pass. One question to ask is whether the system is being overconfident in this strategy. The results in the table support the notion of using the second pass sparingly. In 68 cases, the system was sufficiently confident with its hypothesized city after the first recognition pass to omit the second pass; it made no errors in these decisions.

About a third of the time (29 cases), the system, finding no match, initiated a second pass to incorporate pro-

[3]While DECTalk speech is artificial, we have not explicitly trained our recognizer on it, and thus we argue that it can serve as an effective stand-in for real human speech.

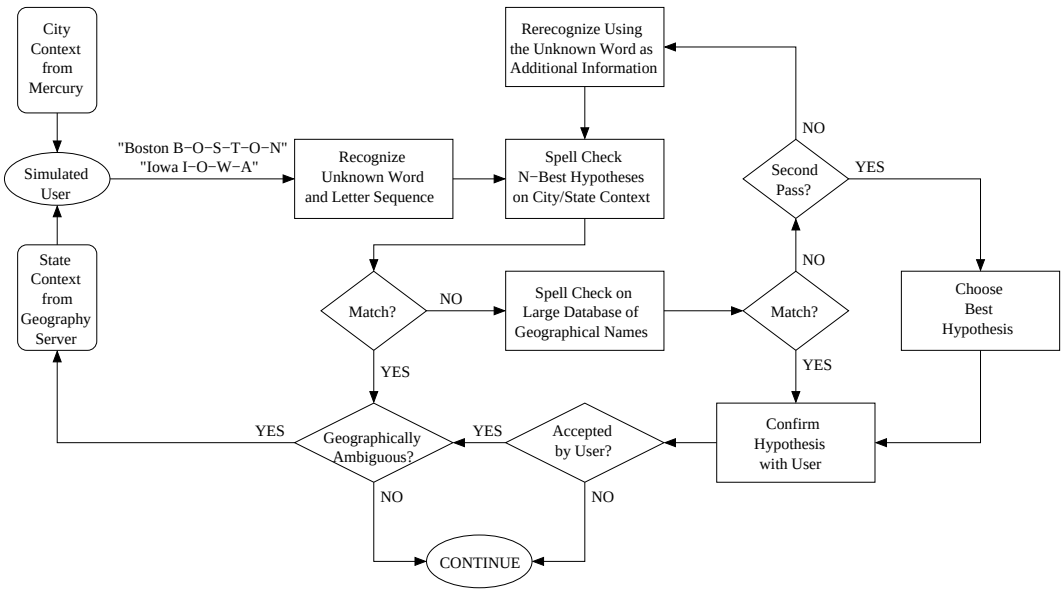

Figure 3: Flow chart detailing the two-pass dialogue strategy for recovering from a problematic source or destination.

Description	*Count*	*Percentage*
correct city after pass 1	68/68	100%
correct city after pass 2:		
short list match	2/2	100%
database match	11/14	78.6%
no match (last resort)	5/13	38.5%
total cities correct	86/97	88.7%

Table 3: Simulation results for 97 speak-and-spell city names showing the number of correct cities hypothesized by the system, after each of two recovery passes. For pass 2, a match was found on the short list or in the geographic database. No match resulted in resorting to the best recognition hypothesis.

nunciation information. There were two instances where the second-pass hypothesized city was found on the short list of focus cities from the original user utterance; both were correct. For the remainder, the large database was consulted. The system proposed the correct city in nearly 79% of the cases. After failing to find any match, the system attempted its last resort of proposing the best hypothesis from the second-stage recognizer. Not surprisingly, the system determined the correct city name in only 39% of these cases. Nevertheless, this percentage suggests that it is certainly better to perform the confirmation rather than to simply tell the user that the city is unknown, given that the recognizer may be correct without the aid of any external lexicon.

The majority of incorrect city hypotheses were due to limitations in the spelling checker and the absence of international names in the geographic database. The current spelling checker, while quite powerful, allows only a single insertion, deletion, or substitution of a letter, or a swap of two letters. We believe that a more robust spelling checker can minimize many of these errors.

The system's performance in hypothesizing the correct candidate for nearly 89% of the problematic city names is encouraging. These results show that this error recovery strategy is largely successful in the synthesize-and-recognize user simulation cycle. The simulated results are, of course, biased in that the simulated user was cooperative with all system requests. The results of the MERCURY analysis in Section 4 show that an errorful or nonexistent response from a user is a very likely possibility. The installation of this strategy in a real system will require that user behavior be carefully monitored.

Although the prospects for the speak-and-spell input mode are promising, we would not want to entirely abandon the use of the telephone keypad. It has been and remains a convenient and effective means by which to spell words. A more appropriate use of the keypad could be as a back-off strategy after the spoken spelling has failed, or in very noisy environments, where speech would be nearly impossible. One advantage of the keypad is that, barring mistakes, the system can be confident that when '3' is pushed, one of the letters, 'D', 'E', or 'F', is intended. When combined with the spoken word being spelled, such keypad ambiguity can be reduced even further (Chung et al., 2003).

7 Future Work

While we feel that the speak-and-spell subdialogue interaction represents a promising strategy for error recovery

in situations of compromised recognition performance, it remains to be seen whether it will work well in real user dialogues. An obvious next step is to incorporate this strategy into our MERCURY system in place of the keypad entry request, and then to assess how well users are able to recover from errors and complete their dialogue interactions.

We are quite encouraged by the effectiveness of the strategy of involving DECTalk to simulate user utterances, and we believe this idea has merit far beyond the experiments conducted here. For example, we are interested in developing a new CSAIL Information domain, which would allow users to call up and ask about the phone number, e-mail address, etc., for the 800 members of our laboratory. Many of these people have foreign-derived names, which would likely be difficult to pronounce, both for users and for DECTalk. However, it would be straightforward to process all 800 first and last names through simulated dialogues, and to obtain a list of those names that failed to be understood by the system. Developers' attention could then be drawn toward the task of assuring these names are included explicitly and are correctly pronounced in the main recognizer, as well as augmenting the letter-to-sound system to include those problematic words in its training corpus. It would even be feasible to lexicalize difficult names within the first stage of the speak-and-spell recognizer, such that an explicit word hypothesis could be extracted from the spoken word at that point.

References

J.G. Bauer and J. Junkawitsch. 1999. Accurate recognition of city names with spelling as a fall back strategy. In *Proc. Eurospeech*, volume 1, pages 263–266, Budapest, Hungary.

I. Bazzi and J. Glass. 2002. A multi-class approach for modelling out-of-vocabulary words. In *Proc. ICSLP*, pages 1613–1616, Denver, Colorado.

G. Chung, S. Seneff, and C. Wang. 2003. Automatic acquisition of names using speak and spell mode in spoken dialogue systems. In *Proc. HLT-NAACL '03*, pages 32–39.

N. Dahlback, A. Flycht-Eriksson, A. Jonsson, and P. Qvarfordt. 1999. An architecture for multi-modal natural dialogue systems. In *Proc. of ESCA Tutorial and Research Workshop (ETRW) on Interactive Dialogue in Multi-Modal Systems*.

J.R. Davis. 1991. Let your fingers do the spelling: Implicit disambiguation of words spelled with the telephone keypad. *Avios Journal*, 9:57–66.

M. Denecke. 2002. Rapid prototyping for spoken dialogue systems. In *Proc. COLING*.

W. Eckert, E. Levin, and R. Pieraccini. 1997. User modelling for spoken dialogue system evaluation. In *Proc. IEEE ASR Workshop*.

J. Glass and S. Seneff. 2003. Flexible and personalizable mixed-initiative dialogue systems. In *HLT-NAACL 2003 Workshop on Research Directions in Dialogue Processing*, Edmonton, Canada.

J. Glass, J. Chang, and M. McCandless. 1996. A probabilistic framework for feature-based speech recognition. In *Proc. ICSLP*, pages 1–4, Philadelphia, PA.

A. Gorin, G. Riccardi, and J. Wright. 1997. How may i help you. *Speech Communication*, 23:113–127.

M. Lundeberg J. Gustafson, N. Lindberg. 1999. The august spoken dialogue system. In *Proc. Eurospeech '99*.

I.S. MacKenzie, H. Kober, D. Smith, T. Jones, and E. Skepner. 2001. Letterwise: Prefix-based disambiguation for mobile text input. In *Proc. UIST*, pages 111–120, Orlando, FL.

M. Marx and C. Schmandt. 1994. Putting people first: Specifying proper names in speech interfaces. In *Proc. UIST*, pages 29–37, Marina del Rey, CA.

R. Pieraccini, E. Levin, and W. Eckert. 1997. AMICA: The AT&T mixed initiative conversational architecture. In *Proc. EUROSPEECH*, pages 1875–1878.

J. Polifroni and G. Chung. 2002. Promoting portability in dialogue management. In *Proc. ICSLP*, pages 2721–2724.

H. Quast, T. Scheideck, P. Geutner, A. Korthauer, and R. Bosch. 2003. Robodima: A dialogue-object based natural language speech dialog system. In *Proc. ASRU Workshop*, pages 174–179, Saint Thomas, Virgin Islands.

H. Schramm, B. Rueber, and A. Kellner. 2000. Strategies for name recognition in automatic directory assistance systems. *Speech Communication*, 31(4):329–338.

S. Seneff and J. Polifroni. 2000. Dialogue management in the mercury flight reservation system. In *Proc. ANLP-NAACL 2000, Satellite Workshop*, pages 1–6.

S. Seneff, R. Lau, and H. Meng. 1996. ANGIE: A new framework for speech analysis based on morpho-phonological modelling. In *Proc. ICSLP '96*, pages 110–113.

S. Seneff, G. Chung, and C. Wang. 2003. Empowering end users to personalize dialogue systems through spoken interaction. In *Proc. EUROSPEECH*, pages 749–752.

S. Seneff. 2002. Response planning and generation in the mercury flight reservation system. *Computer Speech and Language*, 16:283–312.

M. Walker, J. Aberdeen, J. Boland, E. Bratt, J. Garofolo, L. Hirschman, A. Le, S. Lee, S. Narayanan, K. Papineni, B. Pellom, J. Polifroni, A. Potamianos, P. Brabhu, A. Rudnicky, G. Sanders, S. Seneff, D. Stallard, and S. Whittaker. 2001. Darpa communicator dialog travel planning systems: The June 2000 data collection. In *Proc. EUROSPEECH*, pages 1371–1374.

V. Zue and J. Glass. 2000. Conversational interfaces: Advances and challenges. *Proc. IEEE*, 88(11):1166–1180.

V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. J. Hazen, and L. Hetherington. 2000. Jupiter: A telephone-based conversational interface for weather information. *IEEE Transactions on Speech and Audio Processing*, 8(1):85–96.