Hypothesis Selection and Resolution in the Mercury Flight Reservation System

Stephanie Seneff and Joseph Polifroni
Spoken Language Systems Group
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139 USA
{seneff,joe}@sls.lcs.mit.edu) *

ABSTRACT

In a spoken dialogue system, the degree to which the dialogue manager informs and controls the behavior of other human language technology components is an important research topic. Although each separate server can be developed and trained on its own, it must function as part of an entire system, and do so in the context of a complex dialogue with a human user. The dialogue manager is the one component that has not only local information from each server, but also global knowledge about the task and specific knowledge about a particular user's constraints. In this paper, we describe various algorithms we have developed for exploiting the knowledge of the dialogue manager in the selection of recognition hypotheses in the context of human-machine interactions. We describe enhancements we have made to other human language technology servers for the purpose of providing useful information to the dialogue manager, as well as new capabilities in the dialogue manager itself aimed at detecting and repairing problematic spots in the dialogue. We conclude by describing some evaluation metrics and tools we have developed for monitoring system performance.

1. INTRODUCTION

In a spoken dialogue system, one of the most difficult aspects is assuring that the system understood correctly each user query, or, if not, that the system is able to recover gracefully and efficiently from the errors. A tedious though effective strategy is to prompt the user at each turn, soliciting only one piece of information, subsequently verifying through a confirmation subdialogue that it has been correctly understood. A more natural interface would allow the user much greater freedom, but at the price of significantly higher perplexity. In such a mixed-initiative system, it becomes important to draw on as many constraints as possible to aid in the hypothesis selection task. Explicit confirmation can yield greater confidence in the validity of hypothesized utterances, but, again, at the risk of increased tediousness.

This paper discusses how the MIT MERCURY flight reservation system [8] deals with the issues of hypothesis selection and verification. It utilizes a mixed-initiative dialogue strategy supported by confirmation subdialogues that are invoked only when the system actively suspects miscommunication. The system is implemented within the Galaxy Communicator architecture [7], where a central hub mediates interactions among a distributed set of specialized servers. For hypothesis selection, relevant information is retained from prior turns, stored by the hub and distributed to the appropriate servers as requested, mediated via the hub program. The recognizer and the parser, as well as the discourse, dialogue, and generation components, all play a role in the selection process.

The MERCURY system poses interesting and challenging problems for dialogue systems in that the interaction is complex and involves multiple variables. Once these variables are specified, users can become quite confused and the dialogue can be derailed if a serious misrecognition occurs.

In the remainder of this paper, we will first give a brief overview of the entire MERCURY system, and describe both the hypothesis selection process and the method that is used to control dialogue management. Next we describe the confirmation algorithm, which as a policy only confirms when it detects an unexpected response from the user. Following this is a section on our various knowledge sources and their representations. After giving a few example dialogues, we present some evaluation results, including a discussion of how we evaluate the system. We conclude with a summary and a look to the future.

2. OVERALL SYSTEM DESCRIPTION

MERCURY is a spoken dialogue system that allows users to plan air travel between over 500 cities worldwide. The system currently gets all of its flight and fare information from the Travelocity Web site (www.travelocity.com). Following log-on, the user interacts with the system to select the flights of their itinerary. When the flight plan is completed, the system takes the initiative in offering to price and email the itinerary. Finally, the system optionally asks the user a

*This research was supported by DARPA under contract N66001-99-8904 monitored through Naval, Command, Control and Ocean Surveillance Center.

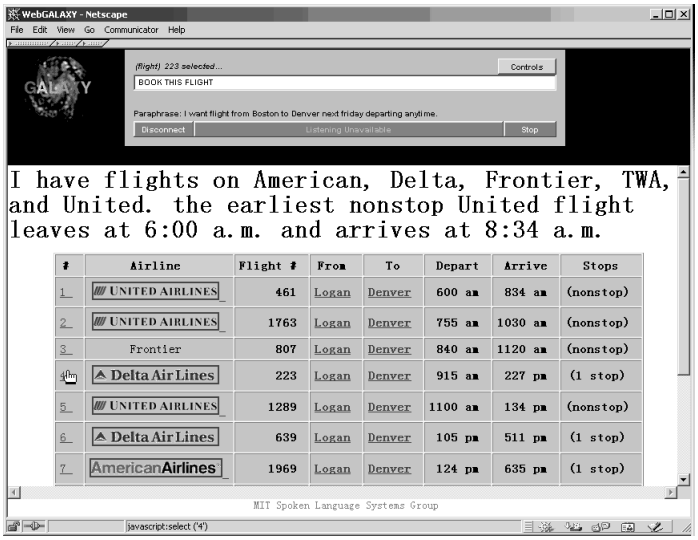

Figure 1: An illustration of the graphical interface to MERCURY. Airline names and source and destination are linked to Web pages for the corresponding airlines and airports. Th user has clicked on the fourth flight, and is asking to add it to the itinerary.

S:	Welcome to the MIT air travel planning system. . . . For enrolled users, please say your first and last name. Otherwise, say, new user.
U:	john doe
S:	Please speak the month and day you registered as your password, as in, 'May first'.
U:	october nineteenth
S:	Welcome John. How may I help you?
U:	i need to fly to salt lake city on may eighth in the afternoon
S:	Okay, flights from Boston to Salt Lake City departing between noon and 6:00 p.m. on Tuesday May eighth. One moment please . . . I have a nonstop Delta flight leaving at 5:55 p.m. and arriving at 9:19 p.m.? Would that work?
U:	is there an earlier flight
S:	Okay, flights from Boston to Salt Lake City departing before 5:55 p.m. on Tuesday May eighth. One moment please . . . What about a Delta flight connecting in New York's Kennedy leaving at 3:25 p.m.? Would that work? I also have a United flight connecting in Chicago O'Hare leaving at 3:30 p.m.
U:	i would like a nonstop flight leaving anytime during the day
S:	Okay, nonstop flights from Boston to Salt Lake City departing between 5:00 am and 8:00 p.m. on Tuesday May eighth. One moment please . . . I have 3 nonstop Delta flights: a flight leaving at 7:00 a.m., a flight leaving at 10:25 a.m., and a flight leaving at 5:55 p.m. Please select one of these flights or change any constraint you have already specified.
U:	i will take the ten twenty [five] flight please
S:	Delta flight 1035 has been added to your itinerary. you depart Boston at 10:25 am and arrive in Salt Lake City at 1:45 p.m. What date will you be returning on?
U:	i would like to return on friday
S:	Okay, Delta flights from Salt Lake City to Boston on Friday May eleventh. One moment please . . . I have 3 nonstop flights: a flight leaving at 9:55 a.m., a flight leaving at 2:35 p.m., and a flight leaving at 5:00 p.m. Please select one.
U:	what time does the two thirty five flight arrive in boston
S:	Delta flight 1473 arrives at 9:05 p.m. in Boston. Shall I add this flight to your itinerary?
U:	what time is [does] that flight leave
S:	I have a nonstop Delta flight leaving at 2:35 p.m. and arriving at 9:05 p.m.? Would that work?
U:	yes please
S:	Delta flight 1473 has been added to your itinerary. you depart Salt Lake City at 2:35 p.m. and arrive in Boston at 9:05 p.m. Would you like me to find out how much your trip will cost?
U:	yes please
S:	Pricing your itinerary will take a while. In the meantime I'll summarize your itinerary for you.On Tuesday May eighth on Delta flight 1035, you depart from Boston at 10:25 a.m. and arrive in Salt Lake City at 1:45 p.m., on Friday May eleventh on Delta flight 1473, you depart from Salt Lake City at 2:35 p.m. and arrive in Boston at 9:05 p.m. The cheapest price for your trip is $455.00. Shall I email your itinerary to you?
U:	yes please
S:	Your itinerary has been emailed to you. Can I help you with anything else?

Figure 2: Example dialogue between a user and the system involving a real trip. "U" denotes a user utterance, and "S" a system utterance. The name has been changed for reasons of anonymity. Words in brackets under the user's turn represent what was actually said rather than what was recognized.

:Week | :Day | :RelDate	$\longrightarrow$	ResolveRelativeDate
:ReturnDate : :Date	$\longrightarrow$	CheckInvalidDate
:HypList & :ImplausibleReturnDate | :RejectedDate	$\longrightarrow$	SelectAlternateDate
:RequestDateConfirmation	$\longrightarrow$	PromptDateConfirm
:ConfirmDate deny	$\longrightarrow$	RequestKeypadDate

Figure 3: Selected entries from MERCURY's dialogue control table concerning dates.

few questions to help determine user satisfaction. MERCURY is intended to be up at all times, and can be reached via a toll-free telephone number (877-628-8255). It is also accessible in displayful mode from a Web page, in which case the spoken interaction is augmented with a graphical display of the set of retrieved flights. A multimodal interface supports clicking on a displayed flight and referring to it verbally: "Book this one," as illustrated in Figure 1.

A telephone dialogue between a naive user and MERCURY is shown in Figure 2. It should be clear from the dialogue that the system offers specific suggestions when appropriate: "Shall I add this flight to your itinerary?" "Can you provide a departure or arrival time?" However, the user is free to say anything at any time; i.e., the full recognition vocabulary is always present. We have always been interested in building dialogue systems that were flexible in this regard, but we are fully aware that a consequence is that recognition errors, which are inevitable, may lead to incoherent dialogues, unless a great deal of attention is devoted to error recovery.

We have thus far collected over 2000 dialogues with users, mostly over the course of the last year. These dialogues were all recorded in detail in log files, and the user queries were also digitally recorded to be used later for training both the recognizer and the natural language component. Perusal of the log files has led to the design of several interrelated strategies for hypothesis resolution, where we make use of diverse sources of information to infer the most plausible solution, including, at times, an explicit request for confirmation of a suspicious hypothesis.

2.1 System Architecture

MERCURY makes use of the GALAXY architecture [6, 7], consisting of a number of specialized servers that communicate with one another via a central programmable hub. An audio server captures the user's speech via a Dialogic board, and transmits the waveform to the speech recognizer [2]. The language understanding component [9] parses a word graph produced by the recognizer and delivers a semantic frame, encoding the meaning of the utterance, to the discourse component. The output of the discourse component [5] is the frame-in-context, which is transformed into a flattened E-form (electronic form) by the generation server. This E-form is delivered to the dialogue manager, and provides the initial settings of the dialogue state.

The dialogue manager consults a *dialogue control table* to decide which operations to perform, and typically engages in a module-to-module subdialogue to retrieve tables from the database. It prepares a response frame, which may or may not include tabular entries. The response frame is sent to the generation component [1] which transforms it in parallel into both a text string and an annotated string that specifies the input controls for the speech synthesizer. Finally, the speech synthesizer [10] transmits a waveform to the audio server which then relays the spoken response to the user over the telephone. The entire dialogue is recorded in detail in a log file for later examination.

2.2 Hypothesis Selection Process

Hypothesis selection is a complex process in MERCURY that involves several steps, including interactions among multiple servers. This process is represented schematically in Figure 4. The recognizer provides a *word graph* representing multiple sentence hypotheses, with associated confidence scores for each word in the graph. The NL component parses the graph, producing an N-best list of *semantic frames*, capturing alternative candidates for the meaning of the utterance. A selection process singles out the most promising of these frames, taking into account possible discourse context, and presents this candidate to the dialogue manager. The dialogue manager then decides whether this request is consistent with the prior dialogue. If some part of the query is problematic, it may do one of several things:

1. Ask the user for explicit confirmation,
2. Seek an alternative hypothesis from the N-best list, that may be more appropriate pragmatically,
3. Reject (delete) certain attributes that are both pragmatically inappropriate and poorly scoring,
4. Initiate a subdialogue asking for confirmation,
5. Ask the user to keypad in the information, as a redundant, but less errorful, source.

The recognizer processes the recorded user waveform and produces a word graph with associated confidence scores for each word in the graph [4]. The confidence scores are based mainly on the log likelihood probabilities of the words, obtained from the acoustic models for their component phones. The confidence scores are obtained from a set of ten features that are combined into a single score using linear discriminant techniques. In addition to the mean and minimum log likelihood score of the word in all of its possible local alignments, the combined score takes into account also the difference between the word's score and the best score obtainable over the same acoustic space, and also against the score of a "catch-all" model. The number of competitors for the acoustic region is also taken into account.

The first step in hypothesis selection is to parse the recognizer's word graph into a set of candidate semantic frames. This is done with our TINA natural language system [9], which parses from a context free grammar augmented with feature unification and a trace mechanism for movement. A stochastic grammar, trained on a large corpus of within-domain sentences, guides the Viterbi search through the word graph. Acoustic and linguistic scores are combined to give an overall sentence score. In addition to the total combined score for each hypothesis, critical content words (e.g., cities and dates) retain their confidence score associ-

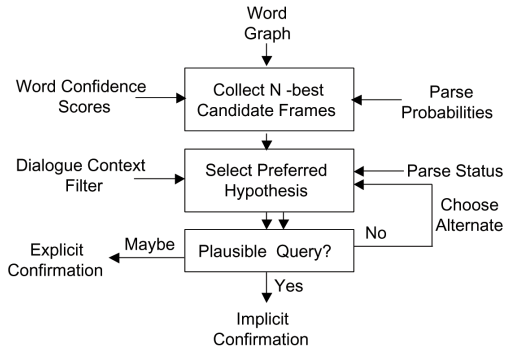

Figure 4: A block diagram of the process of hypothesis selection and verification.

ated with the corresponding element in the semantic frame, for possible later consideration by the dialogue manager.

Each candidate semantic frame is also labelled according to its parse status, with one of four possible categories: "full parse," "robust parse," "phrase spot," and "no parse." "Full parse" means that a single coherent parse tree accounted for every word in the hypothesis. "Robust parse" means that every word was accounted for, but the parse structure consists of a sequence of parsed fragments with possibly interspersed licensed "skip words." "Phrase spot" means that large parts of the hypothesis may have been totally ignored, but certain critical, high scoring, content words were singled out for parsing. Even with all of these back-off mechanism, it is still the case that some user utterances are unparseable. The dialogue manager is responsible for providing a context-dependent response for this "no parse" category (see below).

The next step is to use a simple heuristic to select the most promising candidate from the set of parsed frames. In the absence of any directives from the dialogue component, the system simply chooses the highest scoring full-parse theory, backing off to robust-parse, and finally phrase-spotting. However, it is often the case that the dialogue component has set up context conditions that will preferentially favor an otherwise sub-optimal theory. This can include a list of one or more semantic categories that are in focus, and/or, in some cases, individual words that are highlighted, or individual words that are to be selected *against*. For example, if the system has just asked the user for a return date, then all dates are given preferential treatment. Similarly, if it has just listed the cities it knows in Kentucky, those cities will be highlighted.

Once the most promising hypothesis has been singled out, it is processed through context resolution and delivered to the dialogue manager for consideration. If all goes well, the new information is interpreted and a response is prepared that moves the dialogue plan closer to a conclusion. The alternative hypotheses are retained, but utilized only when there is reason to believe the selected hypothesis is erroneous, as discussed in the following section.

2.3 Dialogue Control Mechanism

The dialogue manager is tasked with the difficult responsibility of determining how best to answer each user's query. With each turn, it processes the user's query, represented as a semantic frame, and prepares its meaning response, also represented as a semantic frame. The generation component converts the reply *frame* into a well formed *reply string*, to be spoken back to the user.

As mentioned earlier, dialogue control is managed in MERCURY through the use of a *dialogue control table*. This table is a simple device for managing complexity – it enforces a linear organization of the complex planning tasks of dialogue management, and provides a high-level representation of dialogue activities in an outline form. The table takes the form of a set of rules, specifying functions to be called when specified conditions are met. The conditions are tests (boolean, arithmetic, string match, etc.) on variables maintained in a dynamic *dialogue state* frame. The variables are initialized from the user's query (in context), and are augmented in the course of a dialogue turn by the various functions that are executed. Each function, which has access to the entire knowledge base (see Section 4), is allowed to return one of three possible "move" states: *continue*, *restart*, and *stop*, with the obvious meanings. *Restart* is typically used to reevaluate a query after dropping a constraint, given that no flights match against the original set of constraints. A final "exit function" is executed at the conclusion of each turn, which updates the dialogue history and finalizes the various parameters that are to be returned to the hub program.

It is up to the system developer to partition the dialogue tasks into a set of specific functions, and to choreograph the order in which, and conditions under which each function should be called. Ideally, each function has a very specific role, some having to do with verifying that the query is fully specified, others involved with retrieving the information from the database, and still others involved with preparing the reply frame. MERCURY's dialogue control table currently contains over 350 rules. Typically, up to twenty or more rules may fire in a single turn.

A selected subset of the rules concerned with managing dates is shown in Figure 3. The first rule is concerned with resolving references such as "the following Friday" or "three days later." The second rule tests whether the understood date is within the time window of MERCURY's knowledge base (ten months into the future), and whether the dates of the itinerary are causal and plausible. It sets up keys that are used by the third rule to search for a more plausible date hypothesis from the N-best list. The fourth rule sets up a prompt to confirm with the user whether the understood date is indeed what they said. The final rule initiates a request for a keypad entry of the date, after the user has rejected a confirmation.

3. CONFIRMATION

By default, the system confirms implicitly what the user said, as illustrated in figure 2, by repeating the understood constraints in the reply. The user then has the opportunity to override any incorrectly understood constraints in a follow-up utterance. However, a number of different conditions can trigger an *explicit* confirmation subdialogue, in which the system delays action pending further input from the user. The general strategy is as follows: if the system detects an unanticipated request from the user, it asks for confirmation. In some cases, it also requests redundant entry via the telephone keypad. The critical pieces of information that may invoke a confirmation subdialogue include signing on, source and destination, travel date, and signing off. The system also considers "no parse" to be a problematic condi-

tion, and a context-dependent response to this situation has been implemented.

Signing On The user signs on by providing orally their name and a password encoded as a date. If an incompatibility is detected, the system then invites the user to enter the password using the telephone keypad. At this point, if the password still appears to be incorrect, the system reexamines alternative hypotheses for the user name, applying a strict filter on the subset that are supported by the password. If this last step fails, the system defaults to a guest-user status.

Source and Destination A number of difficult situations can arise regarding cities, each of which is given special treatment. For example, whenever the user appears to change the source or destination at a point in the dialogue where this is unexpected, the system asks for confirmation. If confirmation fails, it then offers help by informing the user that they can ask what cities it knows in a particular state or country. It is essentially hypothesizing that the user may want to travel to a city that is outside of its known vocabulary.

An interesting case is when the user appears to have identified both a city and a state/country, but they are pragmatically incompatible, e.g., "to Dallas New Jersey." In such cases, it compares the confidence scores to decide which one is most likely to be trustworthy. If it is the state, it lists the cities it knows for that state. Otherwise it accepts the hypothesized city and discards the state.

Under special circumstances, when the system determines that a hypothesized source or destination is likely to be incorrect, it invites the user to enter the city using the telephone keypad. We have determined that, although MERCURY knows over 500 cities, they are uniquely represented by the letter mappings corresponding to the keypad, despite the 3-to-1 ambiguity in spelling words using the keypad. Cities are highly ambiguous if only the first three characters are entered. Thus, the system is licensed to accept a partially spelled city only if it matches a prior hypothesis available from the dialogue history. The algorithm that triggers keypadding of the city is conservative, as this is a rather tedious process and should be avoided unless repeated spoken attempts are failing.

Dates The first time the user provides a date for the next leg of a trip, the system assumes it was correctly recognized, unless it violates pragmatic constraints. However, if the user appears to change a prespecified date, the system prompts for confirmation prior to accepting the changed date. If confirmation fails, the user is invited to key in the date using the telephone keypad. If the user appears to be repeating the same date in isolation, the system suspects a miscommunication. It then browses through any alternative candidate frames, seeking one that provides a novel date. Regardless of whether it succeeds, it prompts the user for confirmation of the selected date candidate, again invoking the telephone keypad upon failure.

We make use of a "date history" (see below) as a way of determining whether a given recognized date conforms with what we know about the dialogue so far, or if the user should be prompted for confirmation. The heuristics around this date history are an ongoing research issue, but this detailed record has proven to be a valuable source of knowledge about the dialogue.

```
{c city_history
  :source "BOS"  :source_status "inherited"
  :dest "GSO"  :dest_status "changed"
  :history {c city_history
             :source "BOS" :source_status "inherited"
             :dest "GSP" :dest_status "repeated"
             :history {c city_history
                         :source "BOS"
                         :source_status "first"
                         :dest "GSP"
                         :dest_status "first" } } }
```

Figure 5: Example city history frame representing the activities in the last three utterances concerning source and destination. This record shows that the source was inherited in the last two utterances, whereas the destination was apparently first repeated and then changed, a likely condition for recognition error. Note: GSO = Greensboro, GSP = Greenville, and BOS = Boston.

Signing Off A surprisingly difficult aspect of mixed-initiative dialogue is deciding when the user is saying "good bye." The problem is that a false recognition is catastrophic here, because the effect is to terminate the conversation. When the user is apparently terminating, the system uses pragmatics to determine whether this is a point at which reasonable closure has been reached. If not, it asks for confirmation.

No Parse Whenever the system fails to understand the user's query, there is the potential that the user's plan and the system's plan may be in conflict. It could be that the user is trying to go to a city that is not supported, or that the system is making a false presupposition about the itinerary. Thus, for example, a "no parse" will trigger a request for keypadding the source or destination, if there is any evidence of confusion with regard to this attribute at that time.

The system asks, "where would you like to go next?" after each flight is selected for the itinerary, and if the user intends a one-way trip, they may phrase that information in a way that is not covered by the grammar. Therefore, in this context, the system always responds to a "no parse" with the reply, "Is this a one-way trip?"

A single "no parse" without any clear evidence for a dialogue disconnect yields a simple apology: "I'm sorry, I'm not sure what you said." A subsequent "no parse" is treated as a request for "help," and invokes a sophisticated dialogue-context dependent *help* mechanism, to try to dislodge the user from an ineffective communication strategy.

4. KNOWLEDGE SOURCES

In order to inform hypothesis selection at any point in the dialogue, we make use of several knowledge sources that are maintained and updated continually throughout the user's conversation with the system. The dialogue state is, of course, one of the most useful of these knowledge sources. The dialogue state encodes parts of both sides of the conversation, in that it identifies any preceding system-initiated query as well as all user-specified constraints. The dialogue state also contains information on how far the user has come in the overall travel plan, which is helpful in determining if

S:	Welcome . . . For enrolled users, please say your first and last name. Otherwise, say 'new user.'
U:	Jane Doe
S:	Please speak the month and day you registered as your password, as in 'May first.'
U:	September six [misrecognized]
S:	Sorry. I didn't catch that. Please enter your four digit date password using the telephone keypad. [successful enrollment]

Figure 6: A dialogue with a user involving logging on, where the keypad entry of the date was successful. The name has been altered for privacy reasons.

	. . .
S:	Continental flight 1662 has been added to your itinerary. You depart San Francisco at 6:30 a.m. and arrive in Houston at 12:20 p.m. What date will you be returning on?
U:	May third [misrecognized]
S:	Okay. Continental flights from Houston to San Francisco on Thursday May third. One moment please . . . Can you provide a departure or arrival time? The earliest nonstop flight leaves at 7:40 a.m. and arrives at 9:59 a.m.
U:	Nope March third
S:	Please enter the four digit month and day using the telephone keypad.
U:	0303 [keypad]
S:	Okay. Continental flights from Houston to San Francisco on Saturday, March third. One moment please . . .

Figure 7: A dialogue with a user where the system prompted for a keypad entry of the date, with subsequent success.

U:	When is the next flight from Anchorage to Seattle?
S:	Flights from Anchorage to Seattle. What airline are you interested in?
U:	Delta
S:	Okay. Delta flights from Anchorage to Seattle on Thursday February 22. One moment please . . . I couldn't find any flights on Delta. I have a nonstop Alaska Airlines flight leaving at 8:59 p.m. and arriving at 1:14 a.m. You arrive the following day. Would that work?
U:	Yes, connecting to San Diego
S:	I heard you say you want to arrive in San Diego. Is that correct? If you need help, you can say, for example, 'What cities do you know in Oregon.'
U:	[I] know Portland.
S:	Please enter the full name of the destination city using the telephone keypad.

Figure 8: A dialogue in which the system was provoked to request a keypad input on the destination city. This request was unsuccessful (the user hung up).

a particular dialogue move is likely.

The system also retains in history a *user model* which is continually augmented as the dialogue progresses through the itinerary plan. It includes any as yet unverified constraints that have been specified but are not yet enforceable, such as an early specification of the return date or the mention of a desired fare class before the itinerary is completed. It also includes the particular details of the selected partial itinerary, which are useful for applying date and source constraints to later legs. In addition, a set of frames is maintained for alternative recognizer hypotheses of dates, for possible later reselection. If the original date is pragmatically implausible, alternative dates are selected from this set. The alternates are flushed each time a new leg is selected.

We have instrumented a detailed record-keeping mechanism for tracking source and destination throughout a dialogue. We have found these two keys to be especially problematic, particularly in cases where the user may be attempting to travel to or from a city that is not in the recognizer's vocabulary. In these cases, the same misrecognition tends to occur repeatedly, as the recognizer continues to substitute the same incorrect hypothesis for the intended city, or the source/destination in question varies from query to query, as the recognizer hypothesizes different cities within its known vocabulary. By monitoring the patterns of source/destination keys from query to query, we hope to be able to decide when to prompt for verification or to solicit keypad input.

Each source and destination city is entered into this history throughout the course of a single dialogue. This history is updated for each turn in which these values are present, either from the user utterance or from inheritance. A status is stored along with the city, indicating whether the city was newly introduced in that turn, changed, repeated, or inherited from a previous turn. The record is stored in a nested frame structure, as illustrated in Figure 5. For each query containing source or destination keys, this record is consulted to determine if the values are consistent with what has appeared before in the dialogue. The city history is flushed whenever a flight is selected for the itinerary. We are currently developing heuristics for determining how to proceed when specific patterns of activity are showing up. Options, as discussed above, are to enter a subdialogue to confirm a newly introduced destination, or to seek a redundant (but in some cases more reliable) entry using the telephone keypad.

5. EXAMPLE DIALOGUES

In this section, we present a number of real dialogues, to illustrate various situations where keypad entry was requested. Figure 6 shows a segment of a dialogue where keypad entry was successful for enrolling the password during the logging on stage. Figure 7 provides an example dialogue where successful keypad entry of a date was triggered. Fig-

To: sls-developers@sls.lcs.mit.edu
From: mercury
Subject: successful dialogue with mercury

The fourth successful dialogue with john doe has just finished. The itinerary was priced at 567.25. The following itinerary was reserved. on Wednesday April 18 on American flight 277, you depart from Boston at 9:30 am and arrive in San Diego at 12:49 pm., on Friday April 20 on American flight 2790, you depart from San Diego at 6:04 pm and arrive in San Jose California at 7:26 pm., on Monday April 23 on American flight 108, you depart from San Jose California at 8:00 am and arrive in Boston at 4:45 pm.

Figure 9: Example of the email message that is sent to system developers when a MERCURY dialogue is completed.

ure 8 shows a rather confusing dialogue in which the system was provoked to request a keypad entry of the departure city. The system did not understand how to interpret the user's rather cryptic utterance, "Yes, connecting to San Diego." One might surmise that the user was not attentive to the system's response during the third turn, and therefore *answered* the question, "What cities do you know in Oregon," rather than asking it. The user hung up at this point, so the keypad request was *not* successful.

We have seen several cases where keypadding was effective for both passwords and dates. Since we have only recently introduced the option to keypad cities into the live system, we are not yet able to say whether this is a productive strategy. At issue is whether the user can keypad an entire city name without errors. We will also probably need to refine the algorithm based on the outcomes of continued dialogue collections.

6. EVALUATION

Dialogue is a notoriously difficult aspect of human language technology to evaluate. The dialogue manager informs and affects the performance of many other parts of the system. The intelligence built into the dialogue manager, exemplified above, is essential for the correct selection and interpretation of utterances in the context of a dialogue. Word, sentence, and concept error rate, all applied on a per-utterance basis, are not sufficient by themselves to indicate that a particular dialogue strategy is more effective for a particular task.

It is not possible to compare two dialogue strategies on the same data. Furthermore, it is difficult both to implement and to interpret a re-evaluation of an enhanced version of the system, because of problems related to both dialogue incoherence and dynamic knowledge sources. We have had some success in reprocessing log files, although the results must be interpreted with care. Furthermore, it is essential to maintain detailed log files that contain representations of all knowledge sources, in order to be able to use them for reprocessing.

6.1 Dialogue Evaluation Metrics

We have recently developed two new metrics, *Query Density* and *Concept Efficiency* to attempt to measure system performance at the dialogue level [3]. These metrics are meant to quantify how effectively a user can convey new information to a system (the "query density"), and how efficiently the system can absorb information from a user ("concept efficiency").

Computing the QD and CE metrics requires reprocessing of dialogue data, after an orthographic transcription has been supplied by hand. Two parallel paths through the entire system are mediated by hub scripts. In the first, the recognizer hypothesis from the time the data were collected is processed; in the second, the orthography of what the user actually said is similarly processed. For each of these paths, a separate key-value representation is obtained and sent to the evaluation server for processing. However, the discourse and dialogue content is maintained exclusively by the branch dealing with the recognizer hypothesis. In this way, the dialogue proceeds as it did at the time of data collection, modulo changes to the data sources and the dialogue manager. Because all of our systems make use of continually updated, dynamic data sources, it is virtually impossible to guarantee that the dialogue interactions which occur during a subsequent evaluation will be coherent.

In typical evaluations on these measures, the system obtains around 1.5 for QD, i.e., on average, one and a half successfully communicated attributes per query, and .92 on CE, i.e., 8% of the attributes had to be repeated.

6.2 "Living" Evaluation

We have found that one of the most important assessment procedures is to manually examine log files of interactions with users, and to guide system development based on interactions where it is clear that alternative approaches would have benefited. This is an iterative procedure tightly coupling data collection efforts with system development. To expedite this process, we have developed mechanisms for monitoring MERCURY's performance on a daily basis, which have been instrumented both in hub programs and by automatic post-processing of session log files.

When a dialogue is completed with the MERCURY system, mail is immediately sent to system developers. This mail is triggered by a rule in a hub program, and provides a summary in English (generated by the system's generation component) of the itinerary obtained, as illustrated in Figure 9. In addition, the mail specifies who the user was and how much experience the user has had in using the MERCURY system (i.e., how many previous calls have been logged to that user). The system also sends a daily email to system developers, summarizing MERCURY's activity on that day. It includes statistics on itineraries obtained and utterances parsed per dialogue, as well as providing a to-date summary of total data collected for the MERCURY system.

Each call to the MERCURY system produces a detailed log file of the interaction, as well as digitized waveform files for each utterance spoken. We have set up a web-based interface to these data, summarizing the interactions for any given day on one page and providing links to separate web pages for each dialogue. By going to a particular dialogue page, a developer can see at a glance the entire interaction, listen to what was spoken, and examine the frames that were used by the MERCURY system in answering each query. In addition, the developer can transcribe or edit a transcript of the speech. Figure 10 shows an example of such a webpage for a recent MERCURY dialogue.

Finally, we ask the users themselves to rate the system

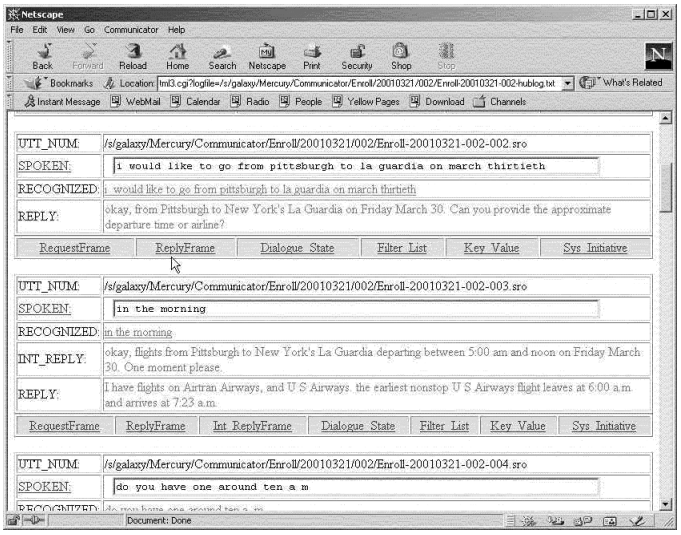

Figure 10: A web page showing a dialogue with a Mercury user. Links point to multiple knowledge sources derived from the log file, as well as enabling playing and transcribing user utterances.

at the end of every dialogue. When a user completes an itinerary or otherwise ends a session, the system asks the user to remain on the line to answer two Yes/No questions ("Was this a real trip you were planning?", "Were you satisfied with the system?") and one query to elicit any comments or suggestions the user has. By transcribing and parsing the responses to these queries, we can automatically correlate user satisfaction and system relevance to other more easily quantified measures of system performance.

7. SUMMARY

This paper has focused on the process involved in determining what the user has plausibly intended at each dialogue turn in a mixed-initiative dialogue, conditioned on a recognizer word graph with associated word confidence scores. The dialogue component directly influences the initial selection process, at least whenever it has provided a specific context. While a set of N-best semantic frames is produced, most of the attention is directed towards the primary selected candidate. After perusal, several problematic situations trigger a response that involves confirmation requests and/or help messages. Sometimes components of the frame are ignored, either because the system can find no appropriate interpretation for them, they have low confidence scores, and/or they conflict with other information present in the same frame. The general strategy is to invoke confirmation subdialogues only when the user appears to make a surprise move. Similarly, alternative hypotheses are only considered when the top hypothesis leads to pragmatically implausible outcomes.

We have found the strategy of backing off to the telephone keypad to be an effective way to ensure successful communication in the face of compromised recognition. We have had extensive experience with keypadding the login password and the dates of the itinerary. Keypadding source and destination city has only been introduced very recently, and it is too early to tell if this method will be effective for cities. Plans are underway to extend this capability to apply to the enrollment of the user name, and ultimately, as an aid in the enrollment of unknown words (e.g., new city or user name).

8. REFERENCES

[1] L. Baptist and S. Seneff, "Genesis-II: A Versatile System for Language Generation in Conversational System Applications," *Proc. ICSLP '00*, Beijing, China, Oct. 2000.

[2] J. Glass, J. Chang, and M. McCandless. "A Probabilistic Framework for Feature-based Speech Recognition," *Proc. ICSLP '96*, pp. 2277–2280, Philadelphia, PA, 1996.

[3] J. Glass, J. Polifroni, S. Seneff, and V. Zue, "Data Collection and Performance Evaluation of Spoken Dialogue Systems: The MIT Experience," *Proc. ICSLP '00*, Vol. IV, pp. 1–4, Beijing, China, 2000, Oct. 2000.

[4] T. Hazen, T. Burianek, J. Polifroni, and S. Seneff, "Integrating Recognition and Confidence Scoring with Language Understanding and Dialogue Modelling," *Proc. ICSLP-2000*, pp. 1042–1045, Beijing, China, Oct., 2000.

[5] S. Seneff, D. Goddeau, C. Pao, and J. Polifroni, "Multimodal Discourse Modelling in a Multi-user Multi-domain Environment," *Proc. ICSLP-96*, pp 192-195, Oct., 1996.

[6] S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue, "Galaxy-II: A Reference Architecture for Conversational System Development," *Proc. ICSLP '98*, pp. 931-934, Sydney, Australia, Dec., 1998.

[7] S. Seneff, R. Lau, and J. Polifroni, "Organization, Communication, and Control in the GALAXY-II Conversational System," *Proc. Eurospeech '99*, Budapest, Hungary, pp. 1271–1274, Oct., 1999.

[8] S. Seneff and J. Polifroni, "Dialogue Management in the MERCURY Flight Reservation System," *Proc. ANLP-NAACL 2000, Satellite Workshop*, Seattle, WA, May, 2000.

[9] S. Seneff, "TINA: A Natural Language System for Spoken Language Applications," *Computational Linguistics*, Vol. 18, No. 1, pp. 61–86, 1992.

[10] J. R. Yi, and Glass, J. R., 1998. Natural-sounding Speech Synthesis using Variable-length Units. *Proc. ICSLP '98*, Sydney, Australia, pp. 1167-1170, Nov., 1998.