JUPITER: A Telephone-Based Conversational Interface for Weather Information

Victor Zue, *Member, IEEE*, Stephanie Seneff, James R. Glass, *Member, IEEE*, Joseph Polifroni, Christine Pao, Timothy J. Hazen, and Lee Hetherington

***Abstract*—In early 1997, our group initiated a project to develop JUPITER, a conversational interface that allows users to obtain worldwide weather forecast information over the telephone using spoken dialogue. It has served as the primary research platform for our group on many issues related to human language technology, including telephone-based speech recognition, robust language understanding, language generation, dialogue modeling, and multilingual interfaces. Over a two year period since coming on line in May 1997, JUPITER has received, via a toll-free number in North America, over 30 000 calls (totaling over 180 000 utterances), mostly from naive users. The purpose of this paper is to describe our development effort in terms of the underlying human language technologies as well as other system-related issues such as utterance rejection and content harvesting. We will also present some evaluation results on the system and its components.**

***Index Terms*—Conversational interfaces, dialogue systems, speech understanding, telephone-based speech recognition.**

I. INTRODUCTION

FOR MORE than a decade, our group has been involved in the development of *conversational interfaces*, interfaces that enable a user to interact with a computer as if it were a conversational partner. To realize such interfaces, several human language technologies must be developed and integrated. On the input side, speech recognition must be augmented with natural language processing, so that utterances can be *understood*, in the context of the preceding dialogue. On the output side, language generation must be integrated with speech synthesis, so that the information sought by the user, as well as any clarification dialogue generated by the system, can be verbalized. In 1989, we first demonstrated such a conversational interface in the form of the VOYAGER urban navigation and exploration system [1]. In 1994, we introduced PEGASUS, a spoken language interface to the on-line EASYSABRE reservation system [2]. PEGASUS evolved from our DARPA air travel information service (ATIS) common task system, but included a far more sophisticated dialogue model.

Manuscript received July 14, 1999; revised August 1, 1999. This work was supported by DARPA under Contract N66001-96-C-8526, monitored through Naval Command, Control and Ocean Surveillance Center. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ronald Rosenfeld.

The authors are with the Spoken Language Systems Group, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: palay@mit.edu).

Publisher Item Identifier S 1063-6676(00)00286-8.

Increasingly, we have found our research agenda being shaped by a strong desire to develop human language technologies and utilize them in a way that will enable universal information access. This has led to our introduction of the GALAXY client-server architecture [3], [4], in which the client can be lightweight, relying on remote servers to perform the compute and knowledge intensive tasks. A logical outgrowth of this research direction is to make use of the most lightweight of all clients: the telephone. Telephone-based information access and delivery is important because the telephone is so much more pervasive when compared to PC's equipped with Internet access. By using the telephone as a means of accessing information, we can empower a much larger population. In the scenario that we envision, a user can conduct *virtual browsing* in the information space without ever having to point or click, or even be tethered to a PC.

Telephone-based interactions pose several research challenges. For example, telephone speech is often hard to recognize and understand due to the reduced channel bandwidth and the presence of noise. In addition, *displayless interaction* relies on only the human auditory channel to receive the information, and thus potentially increases the cognitive load. Such a setting demands a high-quality verbal response. Perhaps even more importantly, the language generation and dialogue management components of the system must work in concert to present the information in digestible chunks, so that the user can easily absorb it. Furthermore, near real-time performance is necessary, since prolonged delay over the phone can be quite annoying to users and render the system unusable.

In 1997, we began to develop JUPITER, a conversational interface that allows users to access and receive on-line weather forecast information for over 500 cities worldwide over the phone. JUPITER utilizes the client-server architecture of GALAXY. It can give a weather report for a particular day or several days, and answer specific questions about weather phenomena such as temperature, wind speed, precipitation, pressure, humidity, sunrise time, advisories, etc. Over a two year period since coming on line in May 1997, JUPITER has received over 30 000 calls (totaling over 180 000 utterances) via a toll-free number in North America, mostly from naive users.

JUPITER also represents our first attempt at *content processing*. Increasingly, we see the need to utilize human language technologies for understanding the content in order to manipulate and deliver the information to the user, since so much of the available content (e.g., in newspapers, radio broadcasts, or Web pages) is linguistic in nature. JUPITER addresses several key issues in content processing, including information selection,

JUPITER:	Hi! Welcome to JUPITER, the MIT Lab for Computer Science weather information system. This call is being recorded for system development. You may hang up or ask for help at any time. How can I help you?
User:	CAN YOU TELL ME WHAT'S THE WEATHER LIKE IN BOSTON TODAY
JUPITER:	In Boston this afternoon. Partly sunny early. Then becoming cloudy with 30 percent chance of showers toward evening. Tonight: rain. Possibly heavy at times. Near 100 percent chance of rain. Can I help you with something else?
User:	YES WHAT ABOUT SAN FRANCISCO
JUPITER:	In San Francisco today: patchy fog near the ocean morning. Otherwise mostly sunny. Tonight: areas of coastal fog and low clouds. Otherwise fair. Is there something else I can do for you?
User:	YES HOW COLD IS IT IN DALLAS TEXAS TODAY
JUPITER:	In Dallas today: high upper 40s. Tonight: lows near 32. What else would you like to know?
User:	I'D LIKE TO KNOW WHETHER IT'S SNOWING IN DETROIT
JUPITER:	I'm sorry, I didn't understand you. Please rephrase your query.
User:	WHAT IS THE WEATHER IN DETROIT
JUPITER:	In Detroit this afternoon: occasional light rain and drizzle. Tonight: rain showers likely. Becoming snow showers by late tonight. 70 percent chance of precipitation.

Fig. 1. Example spoken dialogue between JUPITER and a user.

sampling, redundancy reduction, and semantic representation. The linguistic information that exists online often contains much more detail than the user would like. In the case of JUPITER, for example, the weather information for a given city can be quite extensive, whereas the user may only be interested in the temperature. To be able to provide *information on demand* and deliver just the information that a user needs, JUPITER must fully understand the information it is presenting to the user, so that it will be able to select a subset relevant to the question and avoid boring the user with long and possibly irrelevant monologues. Furthermore, JUPITER's forecasts are obtained from multiple sites on the World Wide Web, whose information content complement one another. It must recognize when two sources are providing overlapping information, and select one or the other based on quality considerations. Finally, by representing all linguistic data in a language-transparent semantic frame format, the notion of translating the weather reports into other languages becomes far more feasible.

The purpose of this paper is to describe our development effort, and to present some evaluation results on the system and its components. The paper is organized as follows. We first provide a system overview. This is followed by a discussion of the human language technology development effort. We then describe our data collection effort, and present some evaluation results. We conclude with a discussion of lessons learned and future work.

II. SYSTEM OVERVIEW

To access JUPITER, a user calls a toll-free number in North America.[1] After a connection has been established, JUPITER speaks a greeting message. After the greeting, the user is free to engage in a conversation with JUPITER, inquiring about weather forecasts for selected cities. The system signals the completion of its turn by playing a brief high tone, indicating its readiness to accept new input. When the system detects that the user has stopped talking, it plays a brief low tone, indicating that it is no longer recording. At this writing, users can only interrupt JUPITER by pressing the "*" key; *verbal* barge-in has not yet been implemented. Fig. 1 gives an example of interactions between JUPITER and a real user.

[1]The number is 1-888-573-8255. For overseas calls, the number is 1-617-258-0300. For more information about the system, users can also access JUPITER's home page at http://www.sls.lcs.mit.edu/jupiter.

A. System Architecture

The initial implementation of JUPITER makes use of our GALAXY conversational system architecture [3]. Since its introduction in 1994 as a client-server architecture, GALAXY has served as the testbed for our research and development of human language technologies, resulting in systems in different domains and languages, and with different access mechanisms. In 1996, we made our first significant architectural redesign to permit universal access via any Web browser. The resulting WebGalaxy system made use of a hub to mediate between a Java GUI client and various compute and domain servers [4].

In 1998, GALAXY was designated as the first reference architecture of the newly launched DARPA Communicator initiative in the US. As a result, we have developed a new version of the GALAXY architecture, this time with the specific goals of promoting resource sharing and plug-and-play capability across multiple sites [5]. To enable multiple system developers to experiment with different domains, components, and control strategies, we made the hub "programmable," i.e., a scripting language controls the flow through each dialogue, such that the same executable can be specialized to a variety of different system configurations. The hub communicates with the various servers via a standardized protocol.

In January 1999, we switched JUPITER to this new, configurable hub architecture, illustrated in Fig. 2. As illustrated in the figure, human language technology servers communicate through the programmable hub using a scripting language. The audio server interacts with the user over the phone line. The turn management server interprets the user query and prepares the system response. The turn manager communicates with the application back-end via a module-to-module subdialogue mediated by the hub. The application back end server retrieves database tuples from a relational database using SQL.

B. Creating the Content

JUPITER can provide weather forecast information for more than 500 cities worldwide. It currently obtains its information from several complementary weather sources available either from the Web, including CNN, the National Weather Service, and *USA Today*, or through direct satellite feeds from WSI. Weather information from the Web is updated three times a day, by polling the various sources for any changes in predictions. Some web sites, such as CNN and *USA Today*, provide

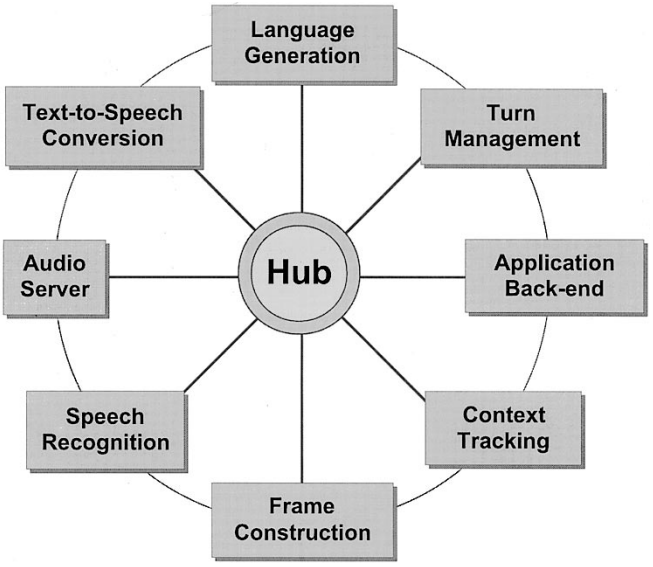


Fig. 2. Illustration of the GALAXY-II architecture.

multiday weather related information on a large number of domestic and international cities, although the information is limited to general weather conditions (e.g., sunny, partly cloudy, etc.) and temperature ranges. The satellite feed provides up-to-the-minute weather information on such things as temperature and humidity.

The National Weather Service provides detailed weather forecasts for 279 JUPITER cities in unrestricted and unformatted text. Since they provide a rich description of the weather, including predictions of amounts of precipitation, advisories for hurricanes, floods, etc., we feel it is worth the extra effort to process them. Extraction of the information is done using our natural language understanding component, TINA, described in Section III-B2.

III. HUMAN LANGUAGE TECHNOLOGIES

In this section, we briefly describe JUPITER's human language technology servers shown in Fig. 2. These include speech recognition, frame construction (i.e., language understanding), language generation, and turn management.[2]

A. Speech Recognition

For speech recognition, we use the SUMMIT segment-based speech recognition system developed in our group [6], [7]. Some of the relevant aspects of the recognizer are described below.

1) Vocabulary: JUPITER's vocabulary has evolved with our periodic analyses of the growing corpus. It currently contains 1957 words, including 650 cities and 166 countries; nearly half of the vocabulary contains geography-related words. The design of the geography vocabulary was based on the cities for which we were able to provide weather information, as well as commonly asked cities. Other words were incorporated based on frequency of usage and whether or not the word could be used in a query which the natural language component could understand. JUPITER currently has an out-of-vocabulary (OOV) rate of 1.9% on a 2507 utterance test set (versus 1.3% on training data).

2) Phonological Modeling: In the current JUPITER recognizer, baseform pronunciations for words are represented using 62 different phonetic units. After drawing the pronunciations for the JUPITER vocabulary from the LDC PRONLEX dictionary, alternate pronunciations are explicitly provided for some words.[3] In addition to the standard pronunciations for single words provided by PRONLEX, the baseform file was also augmented with common multi-word sequences which are often reduced, such as "gonna," "wanna," etc.

A series of phonological rules were applied to the phonetic baseforms to expand each word into a graph of alternate pronunciations. These rules account for many different phonological phenomena such as place assimilation, gemination, epenthetic silence insertion, alveolar stop flapping, and schwa deletion. These phonological rules primarily utilize phonetic context information when proposing alternate pronunciations, although syllabification and stress information can also be used. We have made extensive modification to these rules, based on our examination of the JUPITER data.

Arcs in the pronunciation graph are augmented with probabilities that give preference to more likely pronunciations and penalize less likely pronunciations. Currently, these probabilities are maximum likelihood estimates taken from forced alignments of the training data. The addition of pronunciation graph probabilities reduced error rates by nearly 9% (1% absolute) on a 2500 utterance development set.

In addition to the basic set of 62 units, we have also explored the use of larger inventories of units which incorporate stress information, or which represent larger phonetic sequences which are highly coarticulated (e.g., "or," "all"). However, we have thus far been unable to achieve consistent gains with these more complex inventories. We plan to continue exploring alternative representations in the future.

3) Language Modeling: The JUPITER system makes use of both class bigram and trigram language models. Nearly 200 classes were defined to improve the robustness of the bigram. When trained on a set of nearly 54 000 utterances and evaluated on a test set of 2507 utterances, the word-class bigram and trigram had perplexities of 20.8 and 18.7, respectively. These are slightly lower than the respective *word* bigram and trigram perplexities of 21.6 and 19.9. Note that the class bigram also improved the speed of the recognizer, as it has 20% fewer connections to consider during the search.

During recognition, the class bigram language model is used in the forward Viterbi search. The class trigram language model was originally deployed in a second pass as part of a backward A^* search which used the bigram scores as a look ahead estimate. However, we observed that the A^* search was susceptible to severe thrashing when there were significant differences between the bigram and trigram language models. We subsequently modified the search to first use the backward class bigram to produce an intermediate word graph representation. This word graph is then rescored with class trigram language

[2]We omit text-to-speech generation, which is currently implemented using a commercially available text to speech system, DECtalk.

[3]Vocabulary words missing from the PRONLEX dictionary were entered manually.

model scores and can be converted to N-best outputs if desired. This strategy significantly reduced the worst-case latency of a development set from 12 s to 2 s, with a median latency under 1 s.

4) Acoustic Modeling: For acoustic modeling, the current JUPITER configuration makes use of context-dependent landmark-based diphone models which require the training of both *transition* and *internal* diphone models [6]. Internal diphones model the characteristics of landmarks occurring within the boundaries of a hypothesized phonetic segment, while transition diphones model the characteristics of landmarks occurring at the boundary of two hypothesized phonetic segments. Since there is not enough data to compute acoustic models for all possible diphones, a set of equivalence classes are used to pool data. The current set of 715 classes was determined manually, since they perform slightly better than the automated methods we have explored.

For each landmark, 14 MFCC averages were computed for eight different regions surrounding the landmark, creating a 112-dimensional feature vector. This feature set was reduced to 50 dimensions using principal component analysis. The 715 class diphone models were trained with mixture Gaussian models, with up to 50 components per class. The current models were trained on over 58 000 utterances, collected during system interactions with JUPITER. There are nearly 18 000 Gaussian components in total.

5) Lexical Access: We have recently re-implemented the lexical access search components of SUMMIT to use weighted finite-state transducers with the goals of increasing recognition speed while allowing more flexibility in the types of constraints. We view recognition as finding the best path(s) through the composition $A \circ U$, where A represents the scored (on demand) acoustic segment graph and U the complete model of an utterance from acoustic model labels through the language model. We compute $U = C \circ P \circ L \circ G$, where C maps context-independent labels on its right to context-dependent (diphone in the case of JUPITER) labels on its left, P applies phonological rules, L is the lexicon mapping pronunciations to words, and G is the language model. Any of these transductions can be weighted. A big advantage of this formulation is that the search components operate on a *single* transducer U; the details of its composition are not a concern to the search. As such, U can be precomputed and optimized in various ways or it can be computed on demand as needed. This use of a cascade of weighted finite-state transducers is inspired by work at AT&T [8], [9].

We have achieved our best recognition speed by precomputing $U = C \circ \text{minimize}(\text{determinize}((P \circ L) \circ G))$ for G, a word-class bigram. This yields a deterministic (modulo homophones), minimal transducer that incorporates all contextual, phonological, lexical, and language model constraints [9]. For the current version of the JUPITER recognizer, U has 84 357 states and 562 361 arcs.

For greater system flexibility, we can compute $U = (C \circ \text{minimize}(\text{determinize}(P \circ L))) \circ G$, performing the composition with G "on the fly" during the search. For example, the use of a dynamic language model that changes during a dialogue would require this approach. However, with on-the-fly composition we have found that the system runs about 40% slower than for the fully composed and optimized U.

B. Language Understanding

TINA, a natural language understanding system developed in our group, is used to transform the words into a meaning representation [10]. TINA is used in JUPITER in two distinct ways. It parses user queries into a semantic frame for interpretation by the system, to perform *query understanding*. It also parses weather reports into a meaning representation for purposes of *content understanding*.

1) Query Understanding: For processing user queries, our TINA system selects the most promising candidate hypothesis from a recognizer word graph. It makes use of a manually constructed grammar that encodes both syntactic and semantic information in the parse tree. The final selection process takes into account both the recognition and parse scores, as well as the prior dialogue context. For example, if the system has just listed a set of cities that it knows in California, it will prefer a hypothesis that contains one of the cities on this list.

The grammar attempts to cover all the legitimate ways people could ask questions about weather, but also supports robust parsing through a mechanism that allows unimportant words to be skipped and that can parse sequences of phrase-level units with full connectivity [11]. Probabilities for both the full parse and robust parse solutions are jointly trained on a large corpus of utterances from our data collection, using a completely automatic procedure. In our experience, the evidence of a complete well-formed sentence is a reliable cue, and therefore we have implemented the algorithm to prefer a full parse solution over a robust parse candidate with a superior score. In some cases TINA is unable to produce a solution, even with robust parsing options. For these utterances, the system backs off to a keyword spotting algorithm, which simply extracts all significant keywords that appear sufficiently often in the top ten recognizer hypotheses derived from the word graph.

2) Content Understanding: Three times daily, TINA automatically parses the *content*, i.e., the weather reports, into semantic frames. Most of our data sources produce outputs in a highly predictable format that are easily covered by a small grammar. However, the National Weather Service reports are prepared manually by expert forecasters. As a consequence, they often contain complex linguistic forms. Some sample sentences are given in Fig. 3. During the first few months of weather harvesting, we invested considerable effort in writing parse rules to cover these constructs. Subsequently, our efforts have dropped down to a maintenance level, with the rule base growing very slowly over time. As of this writing, the parse failure rate has been reduced to a fraction of one percent, and is predominantly due to spelling errors.

The parsing process produces semantic frames, which are then sorted into categories based on the meaning. As illustrated in Fig. 4, each weather report is first converted to an indexed list of semantic frames, one for each sentence. The indices are then entered into the relational database under appropriate topicalized categories. To retrieve the answer to a particular user request, the system first retrieves the indices of the relevant sentences in the weather report via an SQL query, then orders them

Very heavy rain likely where storms occur and there is a chance 1 or 2 storms may bring damaging wind or hail.
Mixing with or changing to rain or sleet before tapering off to patchy light rain or snow this afternoon.
Near record low temperatures with a low from upper 20s north portions to near 40 south sections near del Rio.
The national weather service continues the heat advisory for heat index values 100 to 105 today and around 80 tonight.
Mainly east to southeast winds 10 to 20 mph and gusty this morning becoming west to northwest 5 to 15 mph this afternoon.

Fig. 3. Example sentences obtained from National Weather Service weather reports. These sentences are all covered by our grammar.

Index	Sentences	Categories
1	Wednesday	[date]
2	Becoming very windy and turning colder with a 60 percent chance of snow.	[weather] [snow] [wind]
3	Near blizzard conditions and dangerous wind chills developing.	[weather] [snow] [temperature]
4	High around 20 with temperature falling into the single digits by late afternoon.	[temperature]
5	Northwest winds increasing to 25 to 45 mph.	[wind]

Fig. 4. Extract from an online weather report maintained by the National Weather Service, indexed and annotated for categories.

sequentially, and finally paraphrases each of the corresponding semantic frames in turn, to compose a verbal response. Delays are minimal, since the system has preprocessed all current information into semantic frames in a local cache.

The final step of processing occurs when the user requests information. At this time, the selected parse frames are converted into natural English using our GENESIS language generation system (next section). The response is then a paraphrase of selected portions of the original weather report.

An example of selective database retrieval is given in Fig. 5. The user first asked whether it would snow that day in North Dakota. The SQL query returned all the cities in North Dakota for which some mention of snow was made. A follow-up query asks for the particular conditions in Fargo, and the system replies with all parts of that day's weather report that mention snow, which turns out to include a sentence mainly about the wind speed.

C. Language Generation

Language generation in JUPITER makes use of GENESIS, MIT's language generation module [12]. JUPITER uses GENESIS for three very different types of output. In each, the input to GENESIS is a semantic frame, produced by either TINA, the JUPITER turn manager (next section), or a combination of both. The outputs are either a response to be spoken to a user, an SQL query for accessing the weather database, or a set of keyword-value pairs to aid in interpretation of user queries.

Control for GENESIS is configured into three distinct components: a lexicon, a set of message templates, and a set of rewrite rules. The lexicon's main role is to specify the surface form of a semantic frame entry, including the construction of inflectional endings (gender, case, number, etc.). Message templates

User:	WILL IT SNOW TODAY IN NORTH DAKOTA
SQL:	select distinct city from geo, event where event.apt_code = geo.apt_code and event.snowspk is not null
JUPITER:	The forecast indicates snow for the following places in North Dakota: Bismarck, Dickinson, Devil's Lake, Fargo, Grand Forks, Williston, James-town, and Minot. Please select one.
User:	HOW ABOUT FARGO
SQL:	select distinct dayspk, snowspk from event, geo where geo.city = 'Fargo' and event.apt_code = geo.apt_code and event.day = 0 and event.snowspk is not null
JUPITER:	In Fargo in North Dakota today: becoming windy with light snow developing. Wind east increasing to 20 to 30 miles per hour producing drifting snow. Can I help you with something else?

Fig. 5. Example of a short dialogue between a user and JUPITER, showing the database query and the filtered responses.

are used to recursively construct phrases describing the topics, predicates, and clauses of a semantic frame. Finally, the rewrite rules are intended to capture surface phonotactic constraints and contractions. In English, we use rewrite rules to generate the proper form of the indefinite articles "a" or "an," or to merge "a other" into "another." JUPITER utilizes a separate set of control files for each of its three languages (i.e., English, SQL, and keyword-value).

The system response in JUPITER is typically composed from a list of frames, with each frame in the list corresponding to a part of the weather forecast that answers the specific user query. If the user asked about rain, for example, what would then follow are the clause frames from the weather database dealing with precipitation (including references to accumulation, rain mixed with snow, etc.). A phrase containing the reference city and date is inserted prior to the list of frames, to provide contextual grounding for the user. Thus, for example, the response to a query regarding New York City might start with "In New York City, tomorrow." An example of a frame used to construct a user response can be found in Fig. 6.

JUPITER's parsed weather forecast data are stored in a relational database. When the JUPITER turn manager is ready to access this database, it first sends a request to GENESIS for a well-formed SQL query. The semantic frame representing the user input is included along with this request, as well as a key designating that the output language is SQL. GENESIS treats SQL as it does any other language, returning a paraphrase of the semantic frame in SQL, as it would in English or Chinese.

The turn manager makes use of one other language for processing user queries, a flattened representation of the keys and values from the input semantic frame. Paraphrases in this "keyword-value" language are used by the dialogue control module in the turn manager, as well as by the evaluation module to assess understanding accuracy. Fig. 7 shows an example of the semantic frame constructed from user input and the corresponding English, SQL, and keyword-value representations for that query.

D. Turn Management

By monitoring log files from our user interactions with JUPITER, we have become increasingly aware of the benefits of letting real users influence the design of the interaction.

```
{c weather_response
 :continuant {c something_else4 }
 :db_tlist ({c weather_event
             :input "saturday"
             :city {p in
                    :topic {q city :name "new york city" }}
             :pred {p month_date
                    :topic {q date :day "saturday" }}}
            {c weather_event
             :topic {q weather_act
                     :conditional "mostly"
                     :name "sunny"
                     :and {q iwind :name "brisk" }}
             :input "mostly sunny and brisk" }
            ...
            {c weather_event
             :topic {q weather_act
                     :pred {p temp_qual :topic "chilly" }}
             :and {c weather_event
                   :conjn "with"
                   :topic {q lows
                           :pred {p from_value
                                  :qualifier "around"
                                  :topic {q value
          :name 40 }}}}
             :input "chilly with a low around 40" })
 :domain "Jupiter" }
```

Fig. 6. Excerpts from a response frame for the query "What is the weather going to be like tomorrow in New York?" The response by the system was "In New York City Saturday, mostly sunny and brisk,. . ., chilly with lows around 40. What other information can I give you?" Note: "c . . . =" clause, "p . . . =" predicate, and "q . . . =" quantified noun phrase.

```
SEMANTIC FRAME:
{c wh_query
 :topic {q weather
         :quantifier "which_def"
         :pred {p month_date
                :topic {q date
                        :name "tomorrow" }}
         :pred {p in
                :topic {q city
                        :name "new york city" }}}
 :domain "Jupiter" }

ENGLISH: what is the weather in New York tomorrow?

SQL: select distinct geo.apt_code, source, day, dayspk,
city, state, country, region, weathrspk, tempspk from
weather, geo where geo.city = 'New York City' and weathrspk
is not null and day = 1 and weather.apt_code = geo.apt_code

KEYWORD-VALUE:
TOPIC: weather CITY: New York City DATE: tomorrow
```

Fig. 7. Example semantic frame and various paraphrases for the query "What is the weather going to be like tomorrow in New York?"

We have discovered several interesting issues with regard to appropriate response planning to accommodate users' requests. One of the critical aspects of any conversational interface is the need to inform the user of the scope of the system's knowledge. For example, JUPITER has information about a small subset (approximately 500) of the cities in the world, and users need to be directed to select relevant available data when their explicit request yields no information. Even for the cities it knows, JUPITER does not necessarily have the same knowledge for all cities.

JUPITER has a separate geography table organized hierarchically, enabling users to ask questions such as "What cities do you know about in the Caribbean?" This table is also used to provide a means of summarizing a result that is too lengthy to present fully. For example, if the user asks where it will be snowing in the United States, there may be a long list of cities expecting snow. The system then climbs a geographical hierarchy until the list is reduced to a readable size. For example, JUPITER might list the states where it is snowing, or it might be required to reduce the set even further to broad regions such as "New England," and "Northwest." We try to restrict the size of an enumerated list to under ten items, if possible.

During our data collection sessions, we noticed considerable frustration among users seeking information about sunrise and sunset times, when such information did not exist for the cities they requested. We realized that the system needs to distinguish between the general set of cities it knows, and the particular knowledge associated with each of those cities. Based on these observations, we decided to augment the system with the capability of suggesting a list of alternative cities in the same geographic region for which the requested data *are* available. This even applies for cities that are completely unknown to JUPITER, as long as the user has given additional information that can be used to infer a neighborhood. Thus, if the user asks for the weather in an incompatible ⟨city⟩ ⟨state⟩ pair (e.g., due to an out-of-vocabulary word or misrecognition), JUPITER will respond with a list of the cities that it *does* know for that state.

In addition to these general considerations, several phenomena required special attention. For example, we had calls after midnight when users, asking for "tomorrow's" weather, really wanted "today's" weather, defined from midnight to midnight. We also had foreign callers who wanted temperature information presented in degrees Celsius rather than Fahrenheit. We have augmented the system to take these issues into account. Converting temperature to Celsius turned out to be a fairly complex process for the frequent cases where temperature was expressed in qualitative terms such as "highs mid to upper 80's." Finally, to encourage the user to continue the dialogue after each exchange, we implemented a simple mechanism to alternate among a set of continuation phrases, such as "Can I help you with something else?" Fig. 1 shows an actual dialogue between a user and JUPITER, illustrating this behavior.

At the highest level, JUPITER's dialogue module is controlled by a "dialogue control table," which is external to the code. This mechanism is used by all of our GALAXY domains, and we have found it to be very effective in helping system developers to visualize the program flow in the turn manager. The strategy first evolved out of our experience in developing the PEGASUS flight reservation system [2], where it quickly became apparent that complex nestings of subroutine calls led to intractable systems. The mechanism is intended to accomplish two major goals: 1) to transform the hierarchical, organization of nested subroutine calls into a linear sequence of operation calls and 2) to provide a mechanism to succinctly outline the entire system's activities in one or two pages of text.

Each of our turn managers is controlled by a dialogue control table, which specifies a sequence of operations that will fire whenever the specified conditions are met. The conditions consist of arithmetic, string, and logical tests on variables. Upon firing, each operation typically alters the state of one or more

variables, and can return one of three possible outcomes: "continue," "stop," and "restart." Typically, the early rules in the table concern verification that the query is complete and well-formed. Once a query is prepared, a database call produces a result table. The latter half of the dialogue control table is then concerned with interpreting the table and preparing a user response frame.

JUPITER invokes the "restart" action whenever it determines that the query may be over-specified. For example, when the user asks, "Are there any advisories?" the discourse component assumes any region specified in a preceding query. Once the query is evaluated and no advisories are found, JUPITER drops the region constraint and reissues the request, by returning program control to the top of the dialogue control table. JUPITER would then summarize advisories found anywhere in the United States. Similarly, if the user asks for sunrise time in a particular city, and JUPITER discovers that it does not have that information, it restarts with a request for sunrise time in the state associated with that city, after adding to the response frame a comment about the missing information. The resulting response string would be "I have no sunrise information for ⟨city⟩. I have sunrise information for the following cities in ⟨state⟩: ⟨list of cities with sunrise information⟩."

E. Confidence Scoring

A deployed system can produce many unanticipated consequences. In examining the JUPITER corpus, we were surprised to find that users sometimes asked questions that were completely outside of the weather domain, such as "What is today's lottery number?" and "Are there any restaurants in Cambridge?" We decided to augment the vocabulary with support for the most frequently asked out-of-domain queries, replying with a specific apology. It became clear, however, that we also needed a sophisticated form of rejection of misunderstood or unanticipated out-of-domain utterances. This would be far preferable to providing a possibly lengthy, incorrect response. To this end, we developed a confidence scoring algorithm, with the goal of providing a mechanism to eliminate incorrectly understood sentences as much as possible, while continuing to accept as many as possible utterances which were correctly understood.

Different system components can reject a user utterance. The speech recognition component can make use of the likelihood of the acoustic models for a hypothesized word sequence. Phenomena such as out-of-vocabulary or partial words, extraneous noise, and poor signal-to-noise ratio are often mismatched with the acoustic models and can be a source of recognition error. A poor acoustic score can therefore potentially signal an unreliable recognizer hypotheseis. Another indicator of an unreliable hypothesis can be provided by the language model score. Often, when confronted by out-of-vocabulary items, the recognizer will hypothesize an unlikely sequence of words in an attempt to match at the acoustic-phonetic level. Finally, when N-best outputs are computed, the relative scores of successive hypotheses can be an indication of recognizer confidence. In addition to the speech recognizer, the natural language component can also provide valuable information. For example, it is extremely useful to know if the utterance can be parsed.

To carry out this research [13], we first developed a procedure that automatically tags an utterance as either "*Accept*" or "*Reject*," based on a semantic-frame comparison between the recognized and transcribed orthographies. On an evaluation set of more than 2000 utterances, our automatic algorithm achieved a better than 90% agreement with manual annotation.

Once a sufficient number of utterances has been correctly tagged, we can investigate the usefulness of various features for utterance rejection. We have thus far concentrated on utterance-level features, because such features are easily computed and can alleviate the need to combine individual word confidence scores into a meaningful rejection score for the entire utterance. In addition to recognition-based features (e.g., the acoustic and language model scores, the number of words and phones in the hypotheses, and the number of N-best hypotheses), we also investigated the use of linguistic and application-specific features (e.g., parse probability, and the quality of the parse), as well as semantic features (e.g., the relative weights of the word classes).

Next, a Fisher linear discriminant analysis (LDA) classifier was used iteratively to select the best feature set for the classification task. On each iteration, the top N feature sets from the previous iteration were each augmented with one additional feature from the set of M unused features. The $N*M$ new feature sets were scored using LDA classification on a development set, and the top N feature sets were retained for the next iteration. The LDA threshold for each classifier was set to maintain a false rejection rate of 2% on a development set. The procedure terminated when no additional improvement was found. Using this method, we selected a set of 14 features for utterance rejection, which had a correct rejection rate of 60%.

IV. THE JUPITER CORPUS

A. Data Collection

Several different methods have been employed to collect data for JUPITER. We created an initial corpus of approximately 3500 read utterances collected from a variety of local telephone handsets and recording environments. This data set was augmented with over 1000 utterances collected in a wizard environment [14]. These data were used to create an initial version of JUPITER, which naive users could then call via a toll-free number to ask for weather information. The benefit of this eventual setup is that it provides us with a continuous source of data from interested users. Over the past two years, we have collected over 180 000 utterances from over 30 000 calls, all without widely advertising the availability of the system. At this writing, we average over 100 calls per day. Fig. 8 shows the amount of data collected each month over a two year period starting from May 1997.

Tools have been developed so that incoming data can be transcribed on a daily basis [15]. The transcriber starts with the orthography hypothesized by the recognizer during the call, and makes corrections by listening to the waveform file. The transcribed data are also marked for obvious nonspeech sounds, spontaneous speech artifacts, speaker type (male, female, child), and other characteristics as appropriate (e.g., speaker phone,

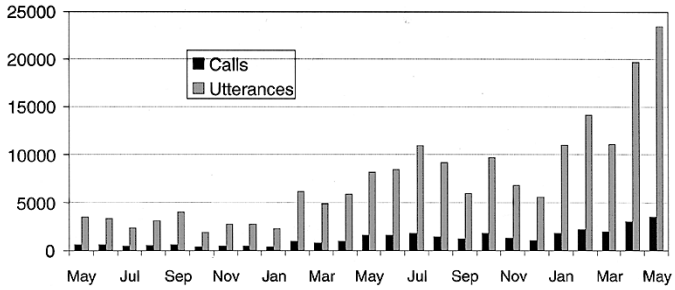


Fig. 8. Plot of the amount of JUPITER data collected from naive users via a toll-free number each month over a two year period starting from May 1997.

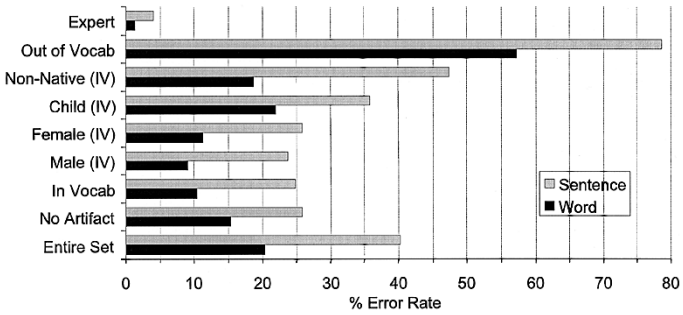


Fig. 9. Detailed analysis of the speech recognition results. (IV = in vocabulary).

heavily accented speech). The transcribed calls are then bundled into sets containing approximately 500 utterances and are added to the training corpus as they become available (with sets periodically set aside for testing).

B. Data Analysis

Data analysis is based on approximately 59 000 utterances from over 10 000 calls. A breakdown of the live data shows that just over 70% of callers are males and approximately 21% females. The remainder of the utterances in the corpus were spoken by children. A portion of the utterances was from nonnative speakers, although the system performs adequately on speakers whose dialect or accent does not differ too much from general American English. Callers with strong accents constituted approximately 7% of the calls and 14% of the utterances. A small fraction (0.1%) of the utterances included talkers speaking in a foreign language (e.g., Spanish, French, German, or Chinese).

The signal quality of the data varied substantially depending on the handset, line conditions, and background noise. It is clear that speaker phones were used in approximately 5% of the calls due to the presence of multiple talkers in an utterance. Less than 0.5% of the calls was estimated to be from cellular or car phones.

Over 11% of the utterances contained significant noises. About half of this noise was due to cross-talk from other speakers, while the other half was due to nonspeech noises. The most common identifiable nonspeech noise was caused by the user hanging up the phone at the end of a recording (e.g., after saying goodbye). Other distinguishable sources of noise included (in descending order of occurrence) television, music, phone rings, touch tones, etc.

There were a number of spontaneous speech effects present in the recorded data. Over 6% of the utterances included filled pauses ("uh," "um," etc.) which were explicitly modeled as words in the recognizer, since they had consistent pronunciations, and occurred in predictable places in utterances. Utterances contained partial words another 6% of the time, although approximately two thirds of these were due to clipping at the beginning or end of an utterance. The remaining artifacts were contained in less than 2% of the utterances and included phenomena such as (in descending order of occurrence) laughter, throat clearing, mumbling, shouting, coughing, breathing, sighing, sneezing, etc.

V. PERFORMANCE EVALUATION

JUPITER is a system that is under constant development. From time to time, we evaluate its performance on unseen test sets. In this section, we will report a snapshot of JUPITER's performance both at the component and system level.

A. Speech Recognition

The recognition test data, consisting of 2507 utterances, represents a collection of calls randomly selected over our data collection period. Of these, 2003 were free of artifacts such as partial words, cross-talk, etc., and a further subset of 1793 were considered to be "in vocabulary" in that they contained no out-of-vocabulary words. The 504 utterances containing an artifact and the other 210 utterances containing an out-of-vocabulary word were combined to create a 714 utterance "out of vocab" set. About 72% of the in-vocabulary utterances (1298) were from male speakers, about 21% (380) were from females, and 7% (115) from children.

Fig. 9 details JUPITER's recognition performance on the test set. The word and sentence error rates (WER and SER) for the entire test set are 20.4% and 40.2%, respectively. The error rates decreased by approximately 25% when utterances containing crosstalk or other nonspeech artifacts were removed. For the in-vocabulary subset, WER and SER are reduced to 10.4 and 24.8%, respectively. Closer examination of the in-vocabulary utterances reveals that error rates for females are somewhat higher than those for males, and the error rates for children are significantly worse. This is probably a reflection of the lack of training material for females and children, although there may be other factors.

Performance on speakers judged to have a strong foreign accent is more than twice as bad as that for male speakers. Finally, the system has considerable trouble (57.3% WER) with utterances containing out-of-vocabulary words and artifacts. This rate may be artificially high, however, due to the nature of the alignment procedure with reference orthographies.[4] It is reassuring, however, to observe that the system performs extremely well on "expert" callers (i.e., mainly staff in our group) who have considerable experience using the JUPITER

[4]Partial words always caused an error due to the nature of our mark-up scheme. Noise artifacts may or may not have caused an error since they were excluded from the alignment transcription (making perfect sentence recognition possible). They typically produced one or more insertion errors however (e.g., during cross-talk).

system, but were not used for training or testing. This behavior is typical of users who become familiar with the system capabilities (a case of users adapting to the computer!).

B. Language Understanding

Language understanding was evaluated using two measures analogous to word and sentence error rates, which we have called *keyword* and *understanding error rates*. Keyword error rate (KER) was designed to be similar to word error rate and uses the same metrics of substitution, insertion, and deletion. Keyword error rate is computed on a set of keyword-value pairs that are automatically generated for each utterance with a parsable orthographic transcription. We have developed a special language within GENESIS for evaluation, one that captures the salient semantic concepts from an utterance in a flattened representation using the format KEYWORD: VALUE. For example, the utterance, "Will it rain tomorrow in Boston?" would be paraphrased into the string "TOPIC: rain; DATE: tomorrow; CITY: Boston" for evaluation. The recognition hypothesis "Will it rain tomorrow in Austin?" would produce a similar paraphrase, with one substitution, on the CITY key. The KER reported in this paper is computed by summing insertions, deletions, and substitutions, and dividing that number by the total number of keys generated from the parsed utterances. An utterance is considered to be understood if all the keyword-value pairs between the hypothesis and reference agree. This is measured by the understanding error rate (UER).

Numbers for understanding error are divided into three categories, based on how the utterance was treated at data collection time and the parse status of the transcription string. Utterances that were answered at run-time and whose transcription strings parse are scored fully for understanding. These utterances may include out-of-domain words or other nonspeech artifacts, but they are included here if their orthographic transcription parses. Utterances that were rejected by the confidence-scoring module are scored separately. The understanding score of these utterances is irrelevant for overall system performance, since the JUPITER turn manager did not generate an answer, but it is useful as a way of evaluating the confidence-scoring module. Finally, utterances whose transcriptions do not parse cannot be automatically evaluated for understanding since there is no way to automatically create a reference meaning representation.

Table I shows the speech understanding evaluation performed on the same evaluation test set that was used for speech recognition. Out of a total of 2507 recorded utterances, 269 contained no speech, and were eliminated from further consideration for this evaluation. JUPITER was able to answer nearly 80% (1755 out of 2238) of the remaining utterances. The word and sentence recognition error rates for this subset were 13.1% and 33.9%, respectively. Keyword error rate for this subset is 14.5%, and the corresponding utterance understanding error rate is 21.2%. Note that many utterances containing recognition errors were correctly understood. Approximately 5% (105 out of 2238) of the utterances were rejected, and these utterances have much higher error rates. Had these utterances not been rejected, the understanding error rate for them would have been 41.9%. The remaining utterances (17%, or 378 out of 2238) did not have a reference parse, and thus are not "evaluable" using the automatic procedure that we developed. Some of these utterances may have been answered correctly, but we have no automatic way of evaluating them. The word and sentence error rates were very high on these data however.

TABLE I
PERFORMANCE SUMMARY FOR WORD (WER), SENTENCE (SER), KEYWORD (KER), AND UNDERSTANDING (UER) ERROR RATES (IN PERCENT) FOR THE 2507 UTTERANCE TEST SET. WER AND SER ARE FOR RECOGNITION ONLY. KER IS BASED ON THE KEYWORD-VALUE EVALUATION, WHILE UER MEASURES UNDERSTANDING ERROR AT THE UTTERANCE LEVEL

	# Utts	WER	SER	KER	UER
Accept	1,755	13.1	33.9	14.5	21.2
Reject	105	27.1	72.4	30.0	41.9
No Ref.	378	56.0	93.4	N/A	N/A

C. Content Understanding

On a typical day, our natural language system, TINA, parses 20 000 sentences from the National Weather Service. Shortly after we began parsing these weather reports, we decided to maintain a careful record of parse coverage over time so that we could determine if the system was reaching convergence in its ability to process content. From an initial parse coverage of 89% during the first week, the system rapidly achieved a parse coverage of over 99% by the eleventh week. Due to the seasonal nature of weather events (e.g., summer hurricanes, winter snow storms), we occasionally encountered previously unseen weather forecasts during the first year of JUPITER development, requiring new grammar rules to accommodate them. In recent months, however, the number of sentences that cannot be parsed hovers around 40 per day, or 0.2%. These sentences are typically set aside and dealt with by the system developers on a monthly basis.

D. Utterance Rejection

Utterance rejection was evaluated using 25 000 utterances collected from naive users during the first part of 1998. Our utterance rejection algorithm incorrectly rejected 2.8% and correctly rejected 63.3% of all utterances, for a total of 82.7% correct *Accept/Reject* decision. Table II shows the classification results in greater detail. Closer examination of incorrectly accepted utterances reveals that there were often misrecognized city names contained in the recognizer hypothesis, or the utterance contained out-of-vocabulary city names, nonspeech events, or out-of-domain requests.

The system responses to rejection are conditioned on the preceding dialogue's rejection pattern. The response to a first rejection is simply, "I'm sorry I didn't understand you." Subsequent rejections elicit increasingly detailed "help" messages, intended to encourage the user to speak sentences within the domain. We have analyzed a corpus of over 6500 queries to see what effect the prior rejection pattern has on the likelihood of rejection of subsequent utterances. As might be anticipated, the system is significantly more likely to accept an utterance subsequent to a previously correctly accepted utterance (80%) rather than subsequent to a single correctly rejected utterance (56%). After

TABLE II
EVALUATION RESULTS FOR JUPITER'S UTTERANCE REJECTION ALGORITHM. CONFIDENCE SCORING RESULTS: THE CORRECT DECISION WAS MADE IN 82.7% OF THE CASES ((14 075 + 6879)/25 346)

	Reference		
System	Accept	Reject	Total
Accept	14,075 (97.2%)	3,980 (36.7%)	18,055
Reject	412 (2.8%)	6,879 (63.3%)	7,291

the second contiguous rejection, the system recovers somewhat, but to only a 64% acceptance rate, still far short of the performance after a correct acceptance. This is in spite of the detailed "help" message that has been provided at this point. In general, system performance is the worst (in terms of recognition and understanding) after multiple failures—if the system is having trouble understanding the user, it continues to have trouble (an example of users *not* adapting to the system!).

VI. DISCUSSION

JUPITER is an example of a new generation of speech-based interfaces that combines several human language technologies to help users access information using a conversational paradigm. Many speech-based interfaces can be considered conversational (e.g., [16]–[19]), and they differ primarily in the degree with which the system maintains an active role in the conversation. For most of the conversational interfaces deployed commercially today, the computer takes control of the interaction by requiring that the user answer a set of prescribed questions, much like the DTMF implementation of interactive voice responses (IVR) systems. In contrast, systems like JUPITER can deal with *mixed-initiative*, goal-oriented dialogue, in which both the user and the computer participate to solve a problem interactively.

JUPITER's content is more complex than data stored in regularized tables. The original weather reports are linguistically diverse, and therefore natural language processing is an integral part of content creation. Besides, the information is dynamic, requiring frequency updates. Finally, the knowledge base of the weather domain (e.g., humidity, temperature, weather advisories, etc.) is conceptually rich and can potentially lead to a wide variety of ways users can query the system.

JUPITER is a manifestation of our ongoing research strategy of developing human language technologies within *real applications*, rather than relying on mock-ups, however realistic they might be. We believe that this strategy will force us to confront critical technical issues that may otherwise not require our attention, such as dialogue modeling, new word detection/learning, confidence scoring, robust recognition of accented speech, and portability across domains and languages. We also believe that working on real applications has the potential benefit of shortening the interval between technology demonstration and its deployment. Above all, real applications that can help people solve problems will be used by real users, thus providing us with a rich and continuing source of useful data. These data are far more useful than anything we could collect in a laboratory environment.

Fig. 10 shows, over a two-year period, the cumulative amount of data collected from real users and the corresponding WER's

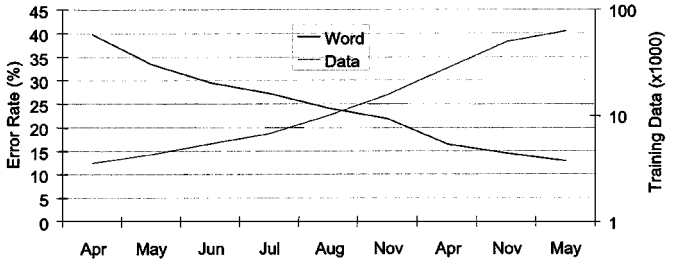


Fig. 10. Comparison of recognition performance and the number of utterances collected from real users over time. Note that the x-axis has a nonlinear time scale, reflecting the time when new versions of the recognizer were released.

of our recognizer. Before we made the system accessible through a toll-free number, the WER was about 10% for laboratory collected data. The WER more than tripled during the first week of data collection. As more data were collected, we were able to build better lexical, language, and acoustic models. As a result, the WER continued to decrease over time. This negative correlation suggests that making the system available to real users is a crucial aspect of system development. If the system can provide real and useful information to users, they will continue to call, thus providing us with a constant supply of useful data. These data have provided a fertile playground for our staff and students to explore different aspects of spoken language research [20]–[23].

Fig. 10 suggests that domain-specific data are crucial for good recognition performance. This is also the case for other components, such as language understanding. Until we can make language technology components domain-independent, or the knowledge acquisition process automatic, building conversational systems for real-world applications will continue to be labor intensive. The development of tools that facilitate knowledge acquisition is an important aspect of the research infrastructure. Even if we can solve these problems, there are a myriad of issues that needs attention, including content processing (e.g., dealing with changes in the format of an HTML document), audio capture (e.g., enabling multiple audio streams), and keeping the system constantly available. Many of these issues have little to do with the development of human language technologies. Nonetheless, they represent a significant part of the system development overhead.

Over the past year, we have begun to utilize JUPITER as the domain in which to conduct research on multi-lingual conversational interfaces, including German, Japanese, Mandarin Chinese, and Spanish. Our approach is predicated on the assumption that the users' queries in different languages can be represented using a common semantic frame [24]. In the case of JUPITER, this appears to be the case. We have begun an effort to paraphrase the weather responses in English into these other languages. For each of these languages, a native speaker who is also fluent in English is preparing the corresponding GENESIS generation rules. In addition, we are also incorporating weather reports in foreign languages, so that region-specific information (e.g., typhoons) can be made available. There were a few instances in which the same word in English had to be given a different translation depending on the context. For example, the word "light," translates differently into Mandarin for the two

phrases, "light wind" ("qinq1 feng1") and "light rain" ("xiao3 yu3"). GENESIS handles this situation using a semantic grammar that can categorize the two cases into different adjective types.

To address the issue of portability, we are in the process of developing other, similar online services as natural extensions to JUPITER. There are a number of similar domains for which the information is dynamic and the vocabulary is sufficiently limited to support practical conversational interfaces. These include flight status information, traffic information, and navigation information. Having multiple application domains will also provide us with the opportunity to explore strategies to navigate seamlessly from one domain to another. We have had some success in building recognizers in these domains using JUPITER's acoustic models.

Finally, JUPITER represents our first attempt at building conversational interfaces to serve real-world users. While it addressed several important research issues such as telephone-based speech recognition/understanding, virtual browsing, and information on demand, the weather information domain simply does not require extensive dialogue management.[5] To support dialogue research, we have recently started the development of MERCURY, a conversational interface for travel planning, which requires tens of turns to accomplish a typical task of making a round-trip flight reservation [25].

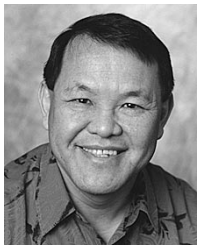
ACKNOWLEDGMENT

The research described in this paper has benefited from the effort of many past and current members of the Spoken Language Systems Group at the MIT Laboratory for Computer Science, including E. Hurley, R. Lau, S. Lee, H. Meng, P. Schmid, N. Ström, and R. Schloming. Their contributions are gratefully acknowledged.

REFERENCES

[1] J. Glass *et al.*, "Multilingual spoken-language understanding in the MIT Voyager system," *Speech Commun.*, vol. 17, pp. 1–18, 1995.
[2] V. Zue *et al.*, "Pegasus: A spoken language interface for on-line air travel planning,," *Speech Commun.*, vol. 15, pp. 331–340, 1994.
[3] D. Goddeau *et al.*, "Galaxy: A human-language interface to on-line travel information," in *Proc. ICSLP*, Yokohama, Japan, 1994, pp. 707–710.
[4] R. Lau, G. Flammia, C. Pao, and V. Zue, "WebGalaxy—Integrating spoken language and hypertext navigation," in *Proc. Eurospeech*, Rhodes, Greece, 1997, pp. 883–886.
[5] S. Seneff *et al.*, "Galaxy-II: A reference architecture for conversational system development," in *Proc. ICSLP*, Sydney, Australia, 1998, pp. 931–934.
[6] J. Glass, J. Chang, and M. McCandless, "A probabilistic framework for feature-based speech recognition," in *Proc. ICSLP*, Philadelphia, PA, Oct. 1996, pp. 2277–2280.
[7] J. Glass, T. Hazen, and L. Hetherington, "Real-time telephone-based speech recognition in the Jupiter domain," in *Proc. ICASSP*, Phoenix, AZ, 1999, pp. 61–64.
[8] F. Pereira and M. Riley, "Speech recognition by composition of weighted finite automata," in *Finite-State Language Processing*, E. Roche and Y. Schabes, Eds. Cambridge, MA: MIT Press, 1997, pp. 431–453.
[9] M. Mohri *et al.*, "Full expansion of context-dependent networks in large vocabulary speech recognition," in *Proc. ICASSP*, Seattle, WA, 1998, pp. 665–668.
[10] S. Seneff, "TINA: A natural language system for spoken language applications," *Comput. Linguist.*, vol. 18, pp. 61–86, 1992.
[11] S. Seneff, "Robust parsing for spoken language systems," in *Proc. ICASSP*, San Francisco, CA, 1992, pp. 189–192.
[12] J. Glass, J. Polifroni, and S. Seneff, "Multilingual language generation across multiple domains," in *Proc. ICSLP*, Yokohama, Japan, 1994, pp. 983–986.
[13] C. Pao, P. Schmid, and J. Glass, "Confidence scoring for speech understanding systems," in *Proc. ICSLP*, Sydney, Australia, 1998, pp. 815–818.
[14] V. Zue *et al.*, "From interface to content: Translingual access and delivery of on-line information," in *Proc. Eurospeech*, Rhodes, Greece, 1997, pp. 2047–2050.
[15] J. Polifroni, S. Seneff, J. Glass, and T. Hazen, "Evaluation methodology for a telephone-based conversational system," in *Proc. 1st Int. Conf. Language Resources and Evaluation*, Granada, Spain, 1998, pp. 43–49.
[16] H. Aust, M. Oerder, F. Seide, and V. Steinbiss, "The Philips automatic train time table information system," *Speech Commun.*, vol. 17, pp. 249–262, 1995.
[17] R. Billi, F. Canavesio, A. Ciaramella, and L. Nebbia, "Interactive voice technology at work: The CSELT experience," *Speech Commun.*, vol. 17, pp. 263–271, 1995.
[18] A. Gorin, G. Riccardi, and J. Wright, "How may I help you?," *Speech Commun.*, vol. 23, pp. 113–127, 1997.
[19] L. Lamel *et al.*, "The LIMSI RailTel system: Field trial of a telephone service for rail travel information," *Speech Commun.*, vol. 23, pp. 67–82, 1997.
[20] A. Halberstadt, "Heterogeneous measurements and multiple classifiers for speech recognition," Ph.D. dissertation, Mass. Inst. Technol., Cambridge, MA, Nov. 1998.
[21] G. Chung, S. Seneff, and L. Hetherington, "Toward multi-domain speech understanding using a two-stage recognizer," in *Proc. Eurospeech*, Budapest, Hungary, 1999.
[22] K. Livescu, "Analysis and modeling of nonnative speech for automatic speech recognition," M.S. thesis, Mass. Inst. Technol., Cambridge, MA, Sept. 1999.
[23] S. Kamppari, "Word and phone level acoustic confidence scoring," M.Eng. thesis, Mass. Inst. Technol., Cambridge, MA, Sept. 1999.
[24] V. Zue, S. Seneff, J. Polifroni, H. Meng, and J. Glass, "Multilingual human-computer interactions: From information access to language learning," in *Proc. ICSLP*, Philadelphia, PA, 1996, pp. 2207–2210.
[25] V. Zue, "Conversational interfaces: Advances and challenges," in *Proc. Eurospeech*, Rhodes, Greece, 1997, pp. KN9–KN18.

[5]An average conversation with JUPITER lasts five to six turns.



Victor Zue (M'79) received the Dr. Sci.degree in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge, in 1976.

He is currently a Senior Research Scientist at MIT, Associate Director of the MIT Laboratory for Computer Science, and Head of its Spoken Language Systems Group. His main research interest is in the development of conversational interfaces to facilitate graceful human/computer interactions. He has taught many courses at MIT and around the world, and he has also written extensively on this subject.

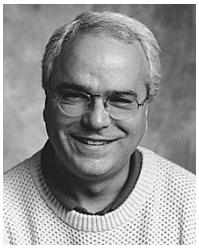
Dr. Zue is a Fellow of the Acoustical Society of America. In 1994, he was elected Distinguished Lecturer by the IEEE Signal Processing Society.

Stephanie Seneff received the S.B. degree in biophysics in 1968, the S.M. and E.E. degrees in electrical engineering in 1980, and the Ph.D. degree in electrical engineering in 1985, all from the Masssachusetts Institute of Technology (MIT), Cambridge.

She is a Principal Research Scientist in the Spoken Language Systems Group, MIT Laboratory for Computer Science. Her current research interests encompass many aspects of the development of computer conversational systems, including speech recognition, probabilistic natural language parsing, discourse and dialogue modeling, speech generation, and integration between speech and natural language.

Dr. Seneff has served on the Speech Technical Committee for the IEEE Society for Acoustics, Speech and Signal Processing. She is currently a member of the Speech Communications Advisory Board and of the Permanent Council for the ICSLP.

James R. Glass (M'78), for a photograph and biography, see this issue, p. 2.

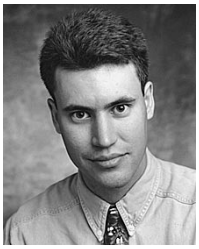


Joseph Polifroni received the B.A. degree in linguistics from Pennsylvania State University, University Park, and the M.S. in linguistics from the University of Pittsburgh, Pittsburgh, PA.

He worked in the Speech Group at Carnegie Mellon University, Pittsburgh, for three years before joining the Spoken Language Systems Group, Massachussetts Institute of Technology, Cambridge. His interests include multilingual systems, dialogue systems, human–computer interaction, and language generation.

Christine Pao received the S.B. degree in physics from the Massachusetts Institute of Technology (MIT), Cambridge.

She has been involved in research and software development in natural language and spoken language systems at the MIT Laboratory for Computer Science. Her most recent work has been developing recognition and understanding confidence measures for utterance rejection and developing and supporting the GALAXY spoken language system architecture.



Timothy J. Hazen received the S.B., S.M., and Ph.D. degrees, all in electrical engineering and computer science, in 1991, 1993, and 1998, respectively, all from the Massachusetts Institute of Technology (MIT).

He joined the Spoken Language Systems Group at the MIT Laboratory for Computer Science as a Research Scientist in 1998. His primary research interests include acoustic modeling, speaker adaptation, automatic language identification, and phonological modeling.

Lee Hetherington received the S.B. and S.M. degrees in 1989 and the Ph.D. degree in 1994, all in electrical engineering and computer science, from the Massachusetts Institute of Technology.

Since then he has been with the MIT Laboratory for Computer Science, where he is currently a Research Scientist in the Spoken Language Systems Group. His research interests include many aspects of speech recognition, including search techniques, acoustic measurement discovery, and recently the use of weighted finite-state transduction for context-dependent phonetic models, phonological rules, lexicons, and language models in an integrated search.