Management and Evaluation of Interactive Dialog in the Air Travel Domain

Lewis M. Norton, Deborah A. Dahl, Donald P. McKay, Lynette Hirschman, Marcia C. Linebarger, David Magerman, and Catherine N. Ball

Unisys Defense Systems
Center for Advanced Information Technology
PO Box 517
Paoli, PA 19301

Introduction

This paper presents the Unisys Spoken Language System, as applied to the Air Travel Planning (ATIS) domain.[1] This domain provides a rich source of interactive dialog, and has been chosen as a common application task for the development and evaluation of spoken language understanding systems. The Unisys approach to developing a spoken language system combines SUMMIT (the MIT speech recognition system [6]), PUNDIT (the Unisys language understanding system [3]) and an Ingres database of air travel information for eleven cities and nine airports (the ATIS database). Access to the database is mediated via a general knowledge-base/database interface (the Intelligent Database Server [4]). To date, we have concentrated on the language understanding and database interface components.

A Dialog Manager integrates the overall user-system interaction. The Dialog Manager accepts user requests in the form of strings of words and calls PUNDIT to interpret the input; it then calls the database indirectly, via the IDS database interface. An important function of the Dialog Manager is to maintain a record of the discourse context, so that the system can successfully process connected dialog.

We first describe our architecture in more detail, then give a short discussion of dialog management, a topic we feel will be crucial to successful systems interacting with users via natural language. We conclude with the presentation and analysis of results from the ATIS common evaluation task and from data gathered at Unisys using our system.

Architecture

System Level Architecture

Our system architecture (see Figure 1) is based on a Dialog Manager, and is taken from a previous application for navigating around Cambridge, Massachusetts [1]. The major difference is that the module providing answers for direction assistance was an expert system, while here it is a database. The Dialog Manager, upon receiving an input from the user, calls PUNDIT for an interpretation of that input. If no interpretation is forthcoming, the user is notified; otherwise the interpretation is passed to a module called QTIP (Query Translation and Interface Program), which attempts to create a database query corresponding to the request. QTIP does not produce SQL code directly. instead, communication with Ingres is done via an Intelligent Database Server [4], developed on another DARPA contract, which we describe in the next section.

Intelligent Database Server

The ATIS Intelligent Database Server (see Figure 2) consists of the Intelligent Database Interface (IDI) and a server supporting the interaction beteen QTIP and a relational database. The IDI provides a logic-based language for queries and transparent support for the actual interaction with an Ingres DBMS. The server supports the interaction with a logic-based query generator (for PUNDIT, QTIP; for the MIT system, TINA [5]). It provides input/output conversion between Ingres and Prolog or Lisp (the languages of choice for language understanding systems), commands for selecting databases, informative

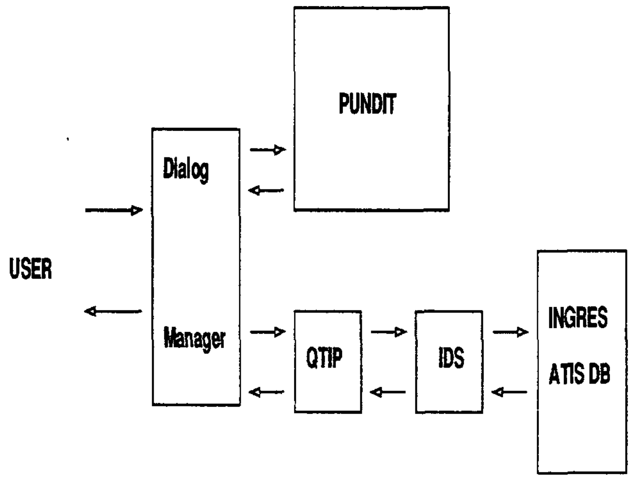

Figure 1: Overall System Architecture

[1]This work was supported by DARPA contract N000014-89-C0171, administered by the Office of Naval Research.

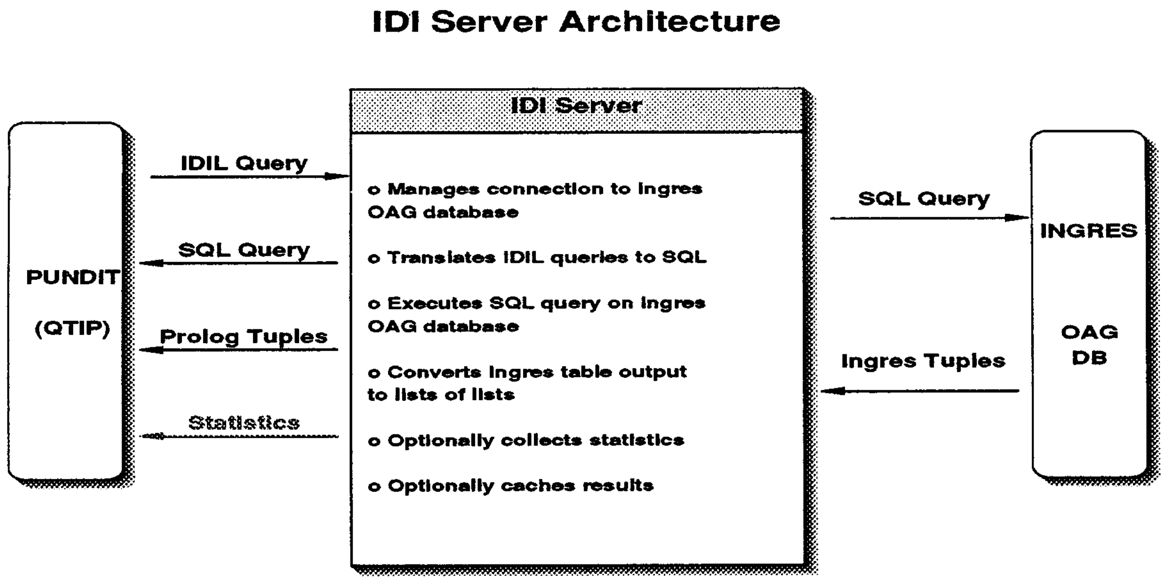
IDI Server Architecture
IDI Server
PUNDIT
(QTIP)
IDIL Query
SQL Query
Prolog Tuples
Statistics
o Manages connection to Ingres OAG database
o Translates IDIL queries to SQL
o Executes SQL query on Ingres OAG database
o Converts Ingres table output to lists of lists
o Optionally collects statistics
o Optionally caches results
SQL Query
Ingres Tuples
INGRES
OAG DB

Figure 2: Intelligent Database Server Architecture

statistics and generation of comparator output.

The Intelligent Database Interface is a portable, cache-based interface providing transparent, efficient access to one or more databases on one or more remote database management systems (DBMS) which support SQL. The IDI was designed to be compatible with a logic-based knowledge representation scheme and has been used to implement a query server supporting the ATIS database access for PUNDIT. The query language of the IDI is the Intelligent Database Interface Language (IDIL) and is based on a restricted subset of function-free Horn clauses where the head of a clause represents the target list (i.e., the form of the result relation) and the body is a conjunction of literals which denote database relations or operations on the relations and/or their attributes (e.g., negation, aggregation, and arithmetic operations).

The Intelligent Database Server supports QTIP's interaction with the ATIS database, accepting IDIL queries, using the IDI to translate and execute the queries and returning results as straightforward lists of lists. The IDI Server supports QTIP's interaction with a relational database in several specific ways; the IDI Server

- accepts an IDIL query as input and returns Prolog tuples or Lisp tuples, the translated SQL query and statistics;
- translates IDIL queries to SQL and executes them on the database;
- manages connections transparently to the Ingres database;
- converts Ingres tuple output to Prolog or Lisp tuples, and
- produces CAS evaluation output.

The IDI architecture also contains a cache; the IDI currently caches results of database queries. Our caching concept also includes the notion of advice provided to the cache. While we have not used it in this application to date, we believe that there are many useful heuristics in the travel planning domain that can lead to optimized DB retrieval strategies, using the cache for query generalizations, pre-fetching and replacement. We plan to collect statistics on ATIS transactions which could then be used to define an effective advice language and strategy.

Overview of the Dialog Manager

The Dialog Manager oversees the interaction of the system with the user. It also oversees the communication between the language understanding subsystem and the database, via QTIP. QTIP reports back to the Dialog Manager, returning both the IDIL query and the zero or more tuples which it received from the IDS (or a diagnostic message containing information explaining why it didn't make a database call). The Dialog Manager then does two things: it presents the answer to the user, using information from the IDIL query to format the tuples in the answer and to generate column headings for them; and it retains the answer in a data structure representing the discourse context. The latter action is what makes it possible for our system to handle certain types of reference by the user to material in answers. The basic idea of having the Dialog Manager store responses was developed for the direction-assistance application. In that task, the expert system responded in English sentences, and the responses were processed by PUNDIT and their interpretations stored in the discourse context. For ATIS, the DB answers are kept, and mechanisms for referencing data in that form have been added to PUNDIT, to enable it to interpret subsequent inputs in the context of both previous user queries and previous system responses. This is illustrated in the next section.

Dialog Management

Reference Resolution

PUNDIT has for some time had the ability to resolve anaphoric references of various kinds, including pronominal reference ("that one") and definite noun anaphors ("the Delta flight", "those flights"). This capability was used in the direction-assistance application, and it is present also for ATIS. We have made no major extensions to this capability for ATIS.

Deixis

Presenting tabular responses encourages the user to refer to entries in those responses. This introduces the need for a different kind of contextual reference. For instance, the code "LH" is used in the ATIS DB both for a (relatively rare) fare class and for Lufthansa German Airlines. If a user asks what "LH" means, the cooperative response is not to give both interpretations, but to give the interpretation corresponding to its use in a previous response, which will be in an "airline" column or a "fare class" column, rarely both. A more interesting example involves expressions like "Delta flight 123". In the general case, that flight has several legs, say from City1 to City2 to City3. If a user has asked for flights from City1 to City2, and then asks for fares for one of them, it is not cooperative to respond with fares not only from City1 to City2, but also from City2 to City3 and from City1 to City3, just because the flight in question goes on to City3. It is quite clear from the context that the user wants only fares from City1 to City2; specifically, Delta flight 123 will have been found in a previous answer tuple for just the City1 to City2 leg. Both of the previous examples involve examination of entries in previous responses. Our system has been extended to handle the "Delta flight 123" example as a demonstration of this kind of capability; we plan to add the ability to handle other such contextual references in the near future.

The mechanisms involved in handling such contextual references can be illustrated by how the "Delta flight 123" example proceeds. When the table with flights from City1 to City2 is returned, a discourse entity representing it is added to the discourse context, but without representing any individual flights as discourse entities. A semantics for the table is also provided, which corresponds to something like "table of Delta flights from City1 to City2". When "Delta flight 123" is subsequently referred to, PUNDIT tries to find a flight in the context with that flight number and airline (we can also handle just "flight 123"), by searching the tables in the discourse context whose semantics are consistent with those of the flight referred to.

Diagnoses

The Dialog Manager is also responsible for enhancing the cooperativeness of the system. It has a primitive generation capability (based on sentence templates and keywords) to provide English diagnoses of failures. For instance, if a user asks for flights leaving after "eight o'clock", QTIP doesn't know if the DB query should specify 8 a.m. or 8 p.m. The optimum action is to request the user to resolve the ambiguity. Our system at least tells the user it couldn't decide between a.m. or p.m. Another example involves queries "just outside" or "on the fringe of" the database. The DB contains information on ground transportation between cities and the airports serving them, but not on ground transportation between two cities or between two airports. If a query requesting information of the latter kind is properly understood, it is preferable for the system to tell the user as specifically as possible why the query cannot be answered, rather than to go through the motions of making a call to the database and returning a "table" with no tuples. Our system issues a message stating that ground transportation is between an airport and a city, not between two cities or airports.

Evaluation

Common Evaluation

The first experiment using our system that we report on is the common evaluation task. Most of the discourse features we discussed in the previous section do not come into play for this task, because it involves testing the system on requests which are entirely self-contained, within the bounds of the domain, and unambiguous–so-called "class A" sentences. Thus there is no need for resolution of anaphoric expressions, reference to previous answer tables, or diagnoses of out-of-domain requests for the June 1990 common evaluation. Furthermore, the answers for evaluation of sentences in context were not uniformly available for the June evaluation. For these reasons, we will not report on an evaluation of sentences in context at this time. In the following section, we present our results for the common ATIS evaluation task, along with an analysis of the data.

The common task data consisted of 90 queries, of which our system obtained correct ("True") answers for 48, or 53%. Of the remainder, 10 resulted in DB calls which obtained inappropriate ("False") information, and 32 resulted in no DB call at all ("NA"). We consider incorrect answers to be even worse than no answers, and the greater than 10% "false alarm" rate experienced on this task to be beyond the acceptable rate for such errors.

The 42 queries that were not successfully processed can be further subclassified as follows. 5 contained items that were not in our lexicon. 9 either did not parse, or did not obtain a usable parse. 10 either obtained no semantic/pragmatic analysis, or an incomplete one. QTIP, though given a complete interpretation of the input, could not create a call to the DB for 12 more, and created an incorrect call 6 times. Our results are summarized in Table 1; there we show the error source for the incorrect ("False") answers and the unanswered ("NA") questions separately. From this table we note that PUNDIT performed quite well. QTIP was directly responsible

	Outcome	True	False	NA
Pundit	lexicon		1	4
	parsing			9
	semantics		3	7
QTIP	QTIP–no call			12
	QTIP–call	48	6	
Totals		48	10	32

Table 1: Common Evaluation Task Results

for nearly half (18 of 42) of the cases where an input query was not processed correctly.

Even among the queries for which our system obtained the correct answer, there were 5 cases where the input was not processed entirely correctly. These cases can be subdivided into two groups, those where the unprocessed material was irrelevant to the handling of the request, and those where the unprocessed material could well have resulted in an error. An example of the former is the query *Under the category ground transportation what is transport A?* Our system ignored the redundant "under the category ground transportation". An example of the latter type is the query *What is the fare on flight eleven forty nine from continental airlines?* Our system failed to process "from continental airlines", but since no other airline has a flight number 1149 in the ATIS DB, the correct answer was obtained anyway.

Speaker	Success Rate
bd	50%
bf	71%
bm	26%
bp	46%
bw	87%

Table 2: Success Rate by Speaker

It is interesting how much variance there is between speakers. There were 5 different speakers in the common task data, and our system's success rate for them ranged from 26% to 87%, as shown in Table 2.

Common Evaluation Analysis

Successful handling of only slightly more than half of the input queries, and class A queries at that, indicates that this system is a long way from being an operational system. However, these data were gathered with maximal co-operativeness (and therefore permissiveness). From our experiences with the direction-assistance application, we suspect that that the ATIS method of gathering the input data decreased the success rate of our (and anybody else's) system, since the Wizard coped with nearly all inputs, giving a user no reason to change mode of expression. By contrast, if user bm had been using our system, his or her lack of success in getting answers might well have led to exploration with alternative ways of phrasing queries, with the result that a larger percentage of inputs would have been processed correctly over the entire session. In the next section, however, we will see that the diagnoses, etc. from our system are not yet good enough to enable such an adaptation to take place for all users.

The bottom line is that our system has not yet achieved a satisfactory level of performance, and it is not hard to understand why. First and foremost, it is an incomplete system, in the middle of its development, and the results of the common task are simply a measure of our progress so far, and in no sense a measure of the level of achievement that our system will attain when fully developed. In fact, in the few weeks before the test, we confined our attention to a subset of about 550 class A queries from development data available to us, and had achieved a success rate of 65% on those. So we were pleasantly surprised to succeed on as many as 53% of new, previously unseen utterances. We believe that this is evidence that our development work is indeed of general applicability for this domain, as opposed to consisting of a collection of ad hoc tricks to make specific inputs get through the system.

On the other hand, why can our system correctly process only 65% of the training input? Why could we not have achieved a greater success rate by now? We suspect that the answer involves the wide range of expressions different people use to make essentially the same requests in this domain. Indeed, in a later section we quantify this observation, comparing vocabulary growth and grammar growth in this domain with that in the direction-assistance domain. To return to a point touched on earlier, it may be significant that the data in the ATIS domain was collected using a Wizard arrangement which bypassed not only the speech recognition component but ALSO the automated natural language understanding component; such was not the case for the direction-assistance experiment.

Our widely different rates of success for the different speakers in the common task data supports the observation that there are a large number of different ways to ask essentially the same questions. And if this is really the case, it means that a natural language understanding system will have to be trained on much larger volumes of data for the ATIS domain. In the direction-assistance domain, we reported having to train on 1000 sentences to a success rate of over 80% before our system could achieve 70% new sentence coverage. It is an open question how many more than 1000 sentences will be necessary for the ATIS domain; currently we have worked with less than 1000 sentences and have achieved (with comparable effort) only a 65% coverage on the class A subset of the training corpus (and about 50% coverage of the entire corpus). It would be informative to train to an 80% coverage and reassess coverage on test data.

Individual examples of successes and failures of our system on the common task data seem not to be of sufficient general interest to report in this paper, given the

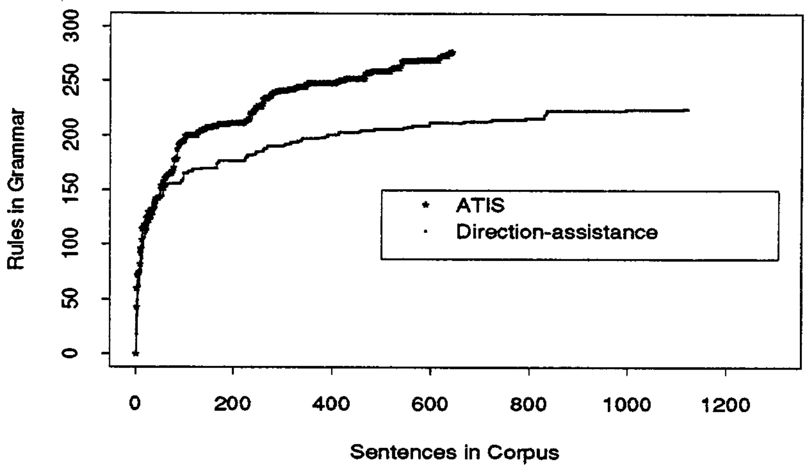

Figure 3: Incremental growth of grammar in ATIS and direction-assistance domains

current limits of our system. We do, however, feel that even these experiments with this partially developed system point to a need to work in the ATIS domain at a task level as well as at a sentence level. So, even with the deficiencies of our system in its present state of development, we have begun experiments along those lines, which we discuss in the next section.

Unisys Data and Evaluation

In order to explore issues in evaluation particularly from the user's perspective, we designed a data collection/evaluation task using the system as a tool to collect data from users. Seven subjects were asked to use the Unisys ATIS system to solve travel planning scenarios. They were given the same instructions as the ATIS subjects at TI, the same scenarios, and the same follow up questionaire. In addition, in order to measure user satisfaction, after the session was over, the subjects were also asked to score each response from the system on a zero to five scale of satisfactoriness. A total of 206 typed inputs were collected,[2] of which 38% were processed correctly. The mean user satisfaction was 2.4 on the 0 to 5 scale. Although we had planned to collect other data, such as time to complete task, very few of the subjects actually completed the task. This was because of the incomplete development of the system and the difficulty of the scenarios. Consequently we were unable to collect this data.

One question which we wished to address was what factors affect user satisfaction in a spoken language system. For example, we were interested in how coverage affects user satisfaction. Coverage is clearly the most important component of user satisfaction, although it does not completely determine it. In comparing user satisfaction on the queries that were handled to those which were not handled, we found a mean rating of 4.8 (on a 0-5 scale) for the queries that were handled and a mean rating of .98 for the queries that were not handled. Some inputs which were not handled received a relatively high score (4) from the users because the error messages were perceived to be useful. For example, the query *From Oakland to Boston, what is the fare?* was answered with the error message *Sorry, could you rephrase that?* which indicates that it wasn't parsed, but it nevertheless got a rating of 4 from the user. On the other hand, sometimes a query which was completely understood got a relatively low rating because the user didn't like how the information was presented. For example, the query *How much does a flight from San Francisco cost?* was answered correctly, but received a score of 3 because the fares presented were not associated with specific flights.

Anecdotally, we noted that response time, which is independent of coverage, is also an important component of user satisfaction. Nearly all the Unisys subjects said that the system was too slow, and 28/53 or 53% of the TI subjects also said that the system they were using was too slow.[3] This data lead us to believe that there may be important trade-offs in coverage and informative error messages vs. speed that can lead to increased user satisfaction and usability of the system.

System Growth as a Function of Training Data

One of our most interesting findings was our ability to quantify the lack of convergence of the ATIS data, both in terms of grammar rules and in terms of lexicon. Starting with the direction-assistance application, we developed techniques for quantifying the growth of the system as

[2]This data is available from the authors upon request.

[3]This data was collected from the TI debriefing questionaires. We thank Charles Hemphill of TI for making these questionaires available to us.

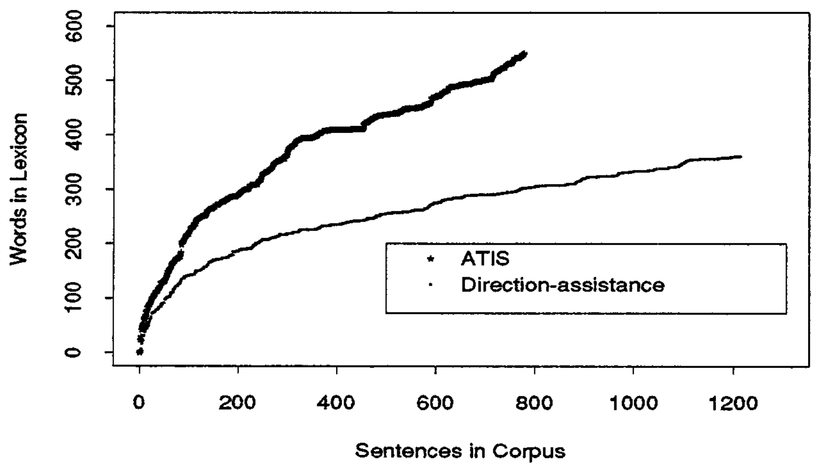

Figure 4: Incremental growth of lexicon in ATIS and direction-assistance domains

a function of training data. We recorded the rate of growth in terms of grammar rules and lexical items as a measure of convergence for both ATIS and direction-assistance ([1],[2]) versions of PUNDIT. Our expectation is that the rate of growth should level off as more and more training is seen. To the extent that it does not, significant gaps in coverage can be expected. Figure 3 shows the incremental growth of the grammar for both domains and Figure 4 shows the incremental growth of the lexicon. It is interesting to note that after 600 sentences from the direction-assistance domain the rate of growth in both grammar and vocabulary is quite slow, indicating that this amount of training data is enough to provide a good sample of the kinds of constructions used in the domain. In contrast, we do not see any leveling off in ATIS growth after 600 sentences. From this we can conclude that a larger set of data will be required to provide a good sample of the constructions needed for ATIS. It is important for future evaluations to develop some better methods for estimating the amount of training data needed for a given application. Since the vocabulary growth curve is similar to the grammar growth curve in both applications it may be that simple measurement of vocabulary convergence would serve as a crude measure of amount of training data needed. We are just beginning to assemble some data points in terms of training data for multiple applications. The direction-assistance vs. ATIS applications illustrate that two seemingly similar kinds of applications can have very different characteristics, perhaps reflecting how the actual data collection was carried out. As we look at more spoken language applications, our ability to make reasonable estimates on training data should improve significantly.

References

[1] Catherine N. Ball, Deborah Dahl, Lewis M. Norton, Lynette Hirschman, Carl Weir, and Marcia Linebarger. Answers and questions: Processing messages and queries. In *Proceedings of the DARPA Speech and Natural Language Workshop*, Cape Cod, MA, October 1989.

[2] Deborah A. Dahl, Lynette Hirschman, Lewis M. Norton, Marcia C. Linebarger, David Magerman, and Catherine N. Ball. Training and evaluation of spoken language understanding system. In *Proceedings of the Darpa Speech and Language Workshop*, Hidden Valley, PA, June 1990.

[3] Lynette Hirschman, Martha Palmer, John Dowding, Deborah Dahl, Marcia Linebarger, Rebecca Passonneau, François-Michel Lang, Catherine Ball, an Carl Weir. The PUNDIT natural-language pr system. In *AI Systems in Government C* Computer Society of the IEEE, March 19

[4] Don McKay, Tim Finin, and Anthony O' intelligent database interface. In *Proceed* 7th *National Conference on Artificial In* 1990.

[5] Stephanie Seneff. Tina: a probabilistic syntactic parser for speech understanding systems. In *Proceedings of the First DARPA Speech and Natural Language Workshop*, Philadelphia, PA, February 1989.

[6] Victor Zue, James Glass, Michael Phillips, and Stephanie Seneff. The MIT SUMMIT speech recognition system: A progress report. In *Proceedings of the First DARPA Speech and Natural Language Workshop*, Philadelphia, PA, February 1989.