Preliminary ATIS Development at MIT

Victor Zue, James Glass, David Goodine, Hong Leung, Michael Phillips, Joseph Polifroni, and Stephanie Seneff

Room NE43-601
Spoken Language Systems Group
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

Introduction

DARPA has recently initiated a plan for a common spoken language task, to be developed independently by all members of the DARPA community, with the hope that it will provide a mechanism leading to appropriate formal evaluation procedures at the level of spoken language. The task that was selected for this purpose is the Air Travel Information System (ATIS) task, based on selected tables from the Official Airline Guide (OAG). It was decided that the first evaluation would be limited in scope to deal with text input only, and to cover only sentences that could be understood unambiguously out of context. Data have been recorded over the past several months at Texas Instruments, using an interface that involves a "wizard" who fully interprets the meaning of the subject's sentences, and generates database responses using a menu driven data access system.

We have been actively engaged in the last few months in developing the natural language and back end portions of the MIT version of the ATIS domain. This paper describes our progress to date on this effort, including an evaluation of the performance of the system on the recently released designated DARPA test set. The remainder of this paper is organized as follows. First we will give a general description of the system we are developing, emphasizing those aspects that differ from the current general conception of the common task. Next we will describe in greater detail certain aspects of the back end, including knowledge representation, control strategy, the user interface, and our preliminary treatment of discourse history. This is followed by a section describing changes made in the parser, in the areas of semantics, the interface with the back-end, and a preliminary new-word treatment. This section also includes a brief discussion of some interesting phenomena that occurred in the training sentences. An evaluation section follows, discussing our system's performance on both training and test data, as well as a preliminary assessment of the perplexity of the system. We conclude with a summary of our results and our position on the nature of the common task.

General Description

Our conception of an ATIS system is somewhat different from the one defined by the common task. First, we would like a domain that is sufficiently restrictive that we could hope to cover most sentences spoken with high probability. That is, it should be easy for the user to understand the limits of the domain. Secondly, we wanted a task that would be of practical interest to a large number of people with minimal prior training. The core task of the OAG, booking flights, meets that requirement. We believe that the task makes sense if it is the traveler, not the agent, who is seeking the information. The traveler would be willing to take the extra time to find out what meals are served and whether he/she could save money with a stopover in Denver. The agent, on the other hand, will typically have no knowledge of the prior preferences of the client in these regards.

We took several steps to limit the types of queries the user would ask. First of all, we omit the display of all code numbers, flight code, fare code and connection code. Connecting flights are displayed directly as pairs of the two legs for each flight, thus obviating the need for questions such as "Show the flights under the connection code 456767." Secondly, in general we display only items of key interest to the traveler. For instance, the subject does not get a meal-code or duration column unless they explicitly ask for them with a sentence such as, "What meals are served on flight twenty-nine." We omitted most of the columns in the aircraft table, presuming that the majority of travelers would consider the information irrelevant. The subject can ask general questions about the aircraft type for the flights of interest, but cannot ask specific questions about range, compression, etc.

We designed the system to be user-friendly as much as possible. We placed heavy emphasis on discourse, which we found to be a very interesting research problem within this domain. Since the system presumes that certain restrictions from the discourse history still hold, we found that it was important for the system to give the user feedback on its understanding of the sentence in context. To that end, the system answers in complete sentences of the form, "There are 4 direct flights from Boston to San Francisco leaving before 3:00 p.m. serving lunch," prior to displaying the table. We are hoping that the user will be less inclined to speak "computerese" if the computer behaves a little more like a person.

Knowledge Representation

We made a major paradigm shift in moving from VOYAGER [5] to ATIS in terms of back-end function operations, one that was necessitated by the fact that the database should be accessed only once, after all restrictions are in place. Within VOYAGER, low-level functions would access the database di-

rectly, passing on sets of objects to be filtered by later low-level functions. Thus, for example the (restaurant) function returns the set of all restaurants, which might later be subsetted by the (serve) function to include, for example, only restaurants serving Chinese food.

In ATIS, low level functions typically fill slots in an event frame with appropriate semantic information. Once the entire sentence has been processed and the history frames have been merged, an IDIL[2][1] query is then constructed from the completely specified frame. We had initially constructed the query "on the fly," but we found that this led to much complexity at the end because of discourse effects that would require last minute alterations on the restrictions in the query.

An Example

As in the VOYAGER domain, we have taken the viewpoint that the parse tree contains semantically loaded nodes at the lower levels, and these are in fact the only semantic information that is used by the interface between TINA[3] and the back-end. The presence of certain nodes within specified positions in the hierarchy triggers the execution of particular functions whose action is typically to update slots in a semantic representation or event frame. The event frame is passed as the first argument to each new function that is called, and the final event frame then has in place all of the appropriate information to be extracted from the sentence.

We will illustrate how the ATIS system converts a sentence into an event frame by walking through a single example. Consider the sentence, "Show me the flights on September twenty ninth from San Francisco to Denver that leave after twelve and cost less than two hundred dollars." The set of nested commands generated by an analysis of the parse tree is shown in Figure 1. The innermost function is the (flight) command which was triggered by the presence of the semantic node A-FLIGHT within the event node FLIGHT-EVENT in the parse tree[2]. It generates an event frame with appropriate slots for all of the types of information that might need to be recorded for a flight.

The event frame is then passed in turn as the first argument to a sequence of functions, each generated by a particular post-modifier to the noun "flights." The presence of a FROM-PLACE node triggers a call to the function (from) with the arguments [flight event][3] and the city [San Francisco]. The (from) function fills the [departure-place] slot in the event frame with the entry [San Francisco], and then returns the event frame. The (to) function is then called with the other city, [Denver] as the second argument, and it fills an [arrival-place] slot with the appropriate information. The (date) function then processes the number "twenty ninth" and the month "September" to update the [date] slot with an entry that includes Saturday as the day of the week, assuming 1990 as the year. The function (clock) is called on the

```
(price
  (leave
    (hour
      (date
        (to
          (from
            (flight)
            (city [San Francisco]))
          (city [Denver])
        (number [twenty ninth])
        (month [September]))
      [after]
      (clock [twelve] nil nil))
  )
  [less]
  (number [two hundred])
  [dollars])
```

Figure 1: Nested functions resulting from analysis of the parse tree for the sentence, "Show me the flights on September 29th from San Francisco to Denver that leave after twelve and cost less than two hundred dollars."

clock-time twelve[4], and it produces a time event-frame which is provided as an argument, along with the keyword [after] to an (hour) function. The (hour) function in turn sets a generic [time] slot in the event frame to a [range] frame representing "after twelve." The verb "leave" triggers a (leave) function which simply moves the entry in the [time] slot to a more specific place, the [departure-time] slot. Since the temporal modifier occurred as an argument of the verb "leave," it is always guaranteed that the appropriate time specifications will be available in the generic [time] slot when the (leave) function is called. Finally, the verb phrase, "costing less than two hundred dollars" generates a call to the (price) function, with the arguments [flight-event], a key-word "less," the number "200"[5] and the unit "dollars." The price function generates another [range] frame specifying a price range, and puts it into the [fare] slot of the [flight-event].

The resulting [flight-event] then contains a number of filled slots, which permit it to generate a self description phrase of the form, "flights from San Francisco to Denver departing after 12:00 on Saturday, September 29 costing less than 200 dollars." The final processing of this [flight-event] fills in some additional slots that were left unspecified by the subject. It presumes, through a set of default conditions, that this must be *direct* flights, the time should be 12:00 *P.M.*, and the price restriction should be on *one-way* fares. Another option would be to query the user for clarification at this time. Finally, an IDIL query is constructed from all of the information in the [flight-event], with the appropriate restrictions, and the database is accessed.

Discourse History

The MIT ATIS system maintains a discourse history of the set of recently mentioned flight restrictions. These restrictions are merged into the current flight event object after

[1]Intelligent Database Interface Language, provided to us by researchers at Unisys as an intermediary to SQL.

[2]These were found through a path that included the nodes [request], [verb-phrase-show] and [dir-object].

[3]returned by the function (flight).

[4]The null arguments are for A.M./P.M. and time zone.

[5]returned by the function (number).

the new sentence has been completely analyzed. The merge procedure allows the flight event to inherit from the history all the features that have not been explicitly mentioned in the current sentence. Thus if the user says, "Show all Delta flights from Boston to Dallas," followed by, "Show only the morning flights," the system will return all Delta flights from Boston to Dallas leaving in the morning. If the new sentence mentions both a from-place and a to-place, the discourse history is completely erased, under the assumption that the user is moving on to a new topic. Likewise, whenever the user successfully books a flight or set of flights, the history is erased. The system refuses to book a set of flights upon request if the number of flights that meet the requirements is not consistent with the number of flights the user wanted to book. In this failure case the system does not erase the history. The user can digress to inquire about definitions of entries or headings in a table, etc., in which case the history is ignored but not erased. If the subject refers anaphorically to the flight set, as in "Which of *these* flights serve breakfast," the system filters the database directly on the set of flight codes available from the previous sentence.

Our discourse history mechanism was actually used on a number of sentences among the type A training set. These were either compound sentences or two-sentence units spoken without a break. A good example is the sentence pair, "I would like to take Delta flight number eighty–three to Atlanta. What is the cheapest fare I can get?" The translation of the parse tree to function calls produced a list of two sets of nested functions, which, when evaluated, generate two flight event frames, named [Delta flight 83 to Atlanta] and [Direct flights costing minimum]. The system first completes the action on the first flight event, providing the user with a time table on Delta flight 83 to Atlanta. It then uses this information as discourse history to answer the second question, producing a table containing the Delta flight along with the price and restrictions for the least expensive fare on that flight.

Natural Language Issues

In this section we will discuss changes that were introduced within the natural language component as a direct consequence of issues that came up within the ATIS task. These include a significant modification to the way semantic filtering is done, the introduction of a preliminary attempt to handle new words in certain restricted contexts, and modifications in the interface between TINA and the back-end. We will also discuss some interesting linguistic phenomena that occurred in the spoken sentences of the subjects.

Semantic Filtering

The ATIS domain led to a significant modification in the way TINA handles semantic filtering. The main change is that, when a new node in the parse tree carries semantic information, it *or's in* its semantic bits with the pre–existing bits, rather than simply replacing them. In conjunction with this mechanism, all pre–existing semantic bits are erased within the domain of each new clause. This has the effect of isolating the semantics of each clause[6] This change was inspired by the inherent free order in the language of the arguments of verbs such as *leave* and *arrive*. These verbs typically take a FROM-PLACE, a TO-PLACE, and an AT-TIME as arguments, but they can occur in any order. Due to the cross-pollenization of rules in TINA, the context-free rules will end up licensing an infinite sequence of multiple arguments. With the semantics *or'd in*, however, a proposed sibling can simply fail if its semantic bit has already been set. This corresponds to saying that, within a clause, only a single FROM-PLACE is permissible. We found this mechanism to be quite powerful, and we intend to modify VOYAGER to also make use of it.

New Word Problem

We have developed a very preliminary capability for handling new words that show up in non-content positions in the sentence. This is done by mapping any word not in the lexicon to a category called LOST, and then allowing certain selected terminals to accept this LOST category as a suitable solution. These words are restricted to positions that either don't affect the meaning of the sentence, or affect the meaning in such a way that a parent node can infer the intended action. That is, the system should be able to provide a "guess" at the correct answer without actually knowing the explicit spelling of the unknown word. Thus, if the user says, "Swell, now show me the flights again," and the system doesn't know the word "swell," it will allow it to map to the same terminal node as "okay," and then proceed to ignore it. A semi-content word would also be acceptable as in, "Please explain the letter Q." The system allowed the unknown word "letter" to come under the same terminal node as the word "symbol." A few verbs, such as "show" are also allowed to map to unknown words. This mapping saved a sentence containing the unknown contraction, "what're," but it also gave a parse with a grossly incorrect meaning in another sentence by substituting "show" for the verb "cancel!" Although we have not done a formal study, there were a number of sentences that were "saved" by this simple strategy.

This approach would only be effective for spoken input in conjunction with a new-word capability within the recognizer. One possibility has been proposed by researchers at BBN [1], where an all-phone general-word model is permitted to succeed only on the condition that it overcomes a stiff word-score penalty.

Interface between TINA and the Back-end

We found that the methods for converting TINA's parse tree to a set of back-end functions that had been developed for VOYAGER worked very well within the ATIS domain. We made only one significant improvement, which was easy to implement but had far-reaching effects. This was to allow certain node types to block access to their children by their ancestors, at the time of analysis of the parse tree to produce back end function calls. This mechanism serves to isolate embedded clauses and noun phrases. For example, a FARE-EVENT expects to process a postmodifier such as PRED-ADJUNCT for a sentence such as "Show all fares *from Boston*

[6]The current-focus and float-object retain semantic information from outside the clause that nodes within the clause may make use of.

to Denver." However, it should not directly process PRED-ADJUNCT in the sentence, "Show all fares *for flights from Boston to Denver,*" where the PRED-ADJUNCT is embedded in the object of the preposition *for* in a following FOR-PHRASE. With the blocking mechanism in place, the OBJECT-PREP appears as a terminal to the FARE-EVENT noun phrase. To fetch the information in the PRED-ADJUNCT it must go through a path that finds the series FOR-PHRASE, OBJECT-PREP, and FLIGHT-EVENT. It is then the job of the FLIGHT-EVENT node to pick up the postmodifying PRED-ADJUNCT. If the blocking is not done, then the post-modifier will be picked up multiple times.

There are several examples of conjunction within the ATIS domain, both at the clause level and at the noun-phrase level. We handled conjunction in a case-specific way, based on whether the functions in the back-end would prefer to take multiple arguments or be called multiple times. For example, a sentence such as "Show all flights from Boston to Denver and show the fares," produces a list of two event-objects, one for each main clause. It behaves just like a sequence of two sentences, with the semantics of the first clause acting as discourse history when the second clause is processed. A sentence such as "Show all flights from San Francisco or Oakland," processes the function *from-place* on multiple arguments. A sentence such as, "What do L, R, and T mean," on the other hand, calls the *code* function three times, once for each code name. The interface is flexible enough to hand tailor the conjunction mechanism to match the functionality of the conjoined elements.

Interesting Sentence Constructs

There were a surprising number of examples in this database of people violating the rules of syntax and semantics. One common error was to use a verb in the singular form when the subject contained multiple singular nouns, as in "What does A, L, R and T mean?" We decided to allow singular form optionally in the grammar in such a case. However, a sentence containing the phrase "all the arrival time" failed to parse on account of number violation between the plural "all" and the singular "time." Similarly, the type-A training sentence, "What does the abbreviations under transport stand for," failed on number agreement between "does" and "abbreviations." We decided to retain these number constraints in the system, thus sacrificing these sentences.

We were intrigued by the relatively frequent occurrence of the verbs "leaving" and "arriving" in situations where the apparent subject of the verb should not carry a +ANIMATE feature. These sentences initially failed to parse on account of subject/verb semantic agreement failure. Some examples are given in Table 1. We pondered the problem for some time, wondering whether a ticket, a flight-list, and a fare should inherit some properties from an implicit flight-event that would give them mobility. However, we finally came to the realization that these forms are probably acting as a dangling participle with an implied PRO subject, appearing at a higher level in the parse tree and therefore not modifying the main noun of the noun phrase. When dealt with this way, the forms could be kept highly restrictive in that 1) they can only occur as gerunds in the present-participle form, and 2) they cannot be followed by a set of additional modifiers for the main noun. According to these rules, sentences such as "I need a one-way ticket to Denver that leaves in the afternoon," and "I need a one-way ticket leaving in the afternoon and costing less than 200 dollars," would fail to parse. When we implemented this gerund form as a dangling participle, all of the sentences containing this odd construct could then parse, because the CURRENT-FOCUS slot was empty at the time the verb was processed, implying a null subject. This solution also handles a similar phenomenon in sentences such as, "Take me from San Francisco to Denver on Sept. 25th leaving after 7:00 P.M. but arriving before 10:00 A.M."

Table 1: Some examples from the training data of sentences with an unusual semantic phenomenon.

- I need a one-way ticket to Denver from San Francisco on Sept. 29th leaving in the afternoon.
- May I see the airfare leaving from San Francisco to DFW, leaving at 12:00 P.M. and arriving at 1:17 P.M.?
- Cost of a first class ticket Dallas to San Francisco departing August the 6th.
- Give me some flight information Dallas to San Francisco departing August 6th.

Performance Evaluation

Four independent releases of so-called Type A sentences were distributed to the DARPA community over the past few weeks, to be used for system development in preparation for the first official DARPA formal spoken language evaluation. Type A sentences are roughly defined as sentences containing no context-dependencies, and having answers that are unambiguous and available within the database. In this section we will attempt to tabulate our progress on handling these sentences, and we will also discuss our performance on the official test set.

A record of our progress over time was kept by evaluating the developing system on each new set upon its arrival, both in terms of parser coverage and agreement of the back-end responses with the canonical answers. A comparison between performance on the new set and performance of the identical system on the previous training set gives an indication of how well the system is generalizing to unseen data. The results are summarized in Figures 2 and 3. Figure 2 gives the initial and final percent coverage of the parser, for each data set, as well as an initial performance for the test set. Figure 3 gives initial, intermediate, and final performance for the percent agreement between our responses and the canonical ones. The intermediate stage represents the status of each release upon arrival of the next release. The solid lines connect runs that were done on the same system.

We saw very little convergence on the patterns of sentences spoken in each data set. Both figures show a consistent trend *downward* in performance on unseen data. We are quite concerned that rules created to deal with utterances in one release don't seem to generalize well to new releases. We were eventually able to get parser coverage up to about 89%

overall and response performance up to about 85% overall, for the training data.

A detailed analysis of the results for the 93 sentence test set is given in Figure 4. 71% of the sentences in the test set parsed, and 60% gave a correct answer to the comparator. We categorized sentences in one of three bins: "parsed," "word spot" and "failure." The "word spot" category comprises sentences which failed to parse but for which we gave an answer by finding a key abbreviation or table heading and presuming that the user wanted a definition of it. This step was inspired by training sentences of the form, "What is transport stand for?" that we didn't want our parser to be able to handle, but could answer correctly by simply detecting the word "transport." As can be seen from the table, this idea was notably unsuccessful, in that only one of the 21 sentences in this category obtained a correct answer. This is in some sense reassuring, because it argues in favor of the extra effort required to fully parse a sentence.

The most interesting category is the set of sentences that parsed but for which we gave either no answer or an incorrect one. There were a total of 12 sentences in this category, six of which failed due to an inappropriate translation of the parse tree into back-end functions.[7] The remaining six failed due to problems in the back end or inconsistencies between our understanding of the "correct" answer and the true one. Two sentences failed because we had assumed that "San Francisco airport" meant only "SFO," whereas the canonical answer included "OAK" as well. Another two failed due to a table fault in translating from "Dallas Fort Worth" to "DFW." The final two failed because we were not properly informed of the canonical definition of "advance purchase."

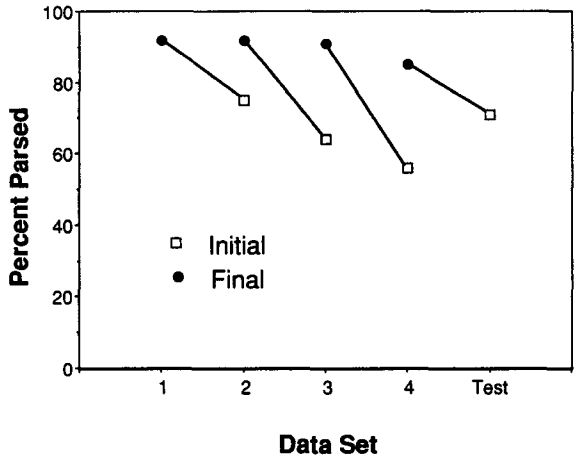

Figure 2: Graph of percent coverage of TINA as a function of release number, for the four sequential releases and the test set.

In order to assess the difficulty of this task for recognition, we computed the test-set perplexity for the condition when probabilities were obtained from the four releases of training data, and the condition when all words were considered equally likely. The vocabulary size was 571 words, and the perplexity, without probabilities, was 166.5, or nearly 1/3 of the full vocabulary.[8] However, when we trained on frequencies of occurrence in the training data, the perplexity was reduced to only 10.8, a factor of 15 improvement. We were very much surprised by this result, as the VOYAGER domain only gives a factor of three improvement, although Resource Management had yielded a factor of eight. We suspect that this is due in part to the fact that the ATIS domain contains terminal categories such as certain abbreviations that have large numbers of words but rarely occur. In addition, there were no sentence fragments and no sentences using indirect speech such as "I want to go...." For instance, the sentence "Show me the flights," gives a perplexity reduction from 300 to 5 with the use of probabilities, reflecting the fact that this is a very common form.

[7]The four "no answer" sentences fell in this bin.

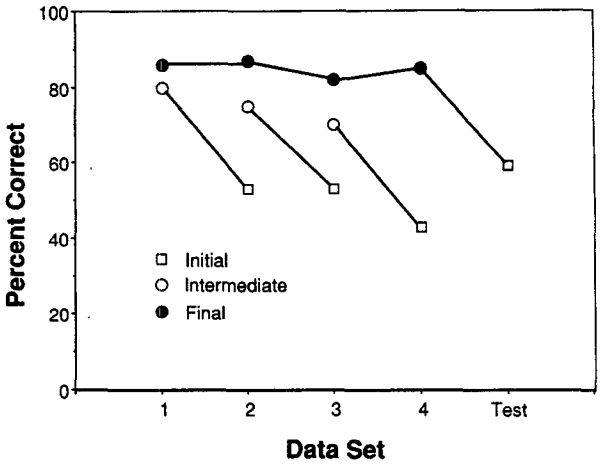

Figure 3: Graph of percent agreement between system responses and canonical answers, as a function of release number, for the four sequential releases and the test set. The intermediate performance, in each case, is the level of performance for that set at the time of release of the next set.

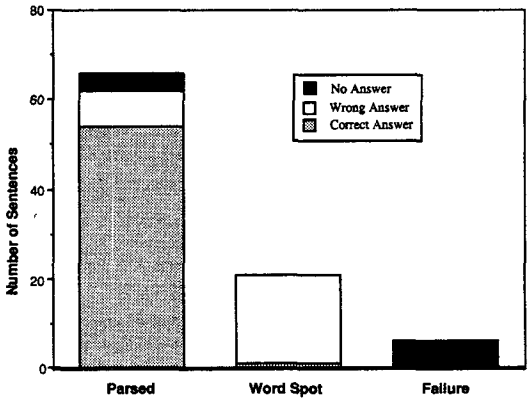

Figure 4: Detailed analysis of the results for the TI test data.

[8]A word-pair grammar would give a further increase in perplexity. For Voyager the perplexity increased from 28 to 73 when long-distance constraints were ignored.

Data Collection

We have performed a preliminary data collection session, using a procedure that mimics closely the one we used for VOYAGER[4]. We told subjects to pretend they were a traveler planning a trip. Their task was to work together with the system to select particular flights that would meet their requirements. They could book the selected flights, with the understanding that booking in real use means recording the relevant information so that a travel agent could complete the task. We reminded them of the types of information the system contains (meals, fares, times, etc.). The wizard's task was a simple one of typing in verbatim (minus false starts) what the user said. The system answered as well as it could, and identified unknown words or points in the sentence where it got stuck. We found that over 100 sentences could be collected in an hour. Preliminary examination of the sentences shows that they are quite different from the ones collected at TI, in particular with regard to indirect speech acts and anaphora. We are encouraged by the results of our data collection procedure, and are hopeful that we can contribute to collecting more spoken sentences in the future.

Summary

We have now completed an initial port of our natural language system to the new ATIS domain, and we have developed our first interface with a standard database using SQL. For the most part, methods used for the VOYAGER domain could be effectively applied. The most significant change was to maintain an intermediate representation of the results of processing a sentence in the form of a hierarchical tree of frames recording the appropriate semantic information. A final postprocessing function develops an SQL query from the semantic representations, and fetches the appropriate database entries.

We spent considerable effort on expanding the coverage of our system to handle forms that showed up in the four releases of Type A sentences from Texas Instruments. As a consequence of this effort, we were able to achieve an 89% coverage overall of the parser on these sentences, with 85% of them yielding an answer that conformed to that required by the comparator. For the test release, 71% of the sentences could parse, and 60% gave a correct response.

We believe that the idea of a common task involving booking flights is a good one. It would be useful to a great deal of people with no prior experience, and hence it could potentially be a profitable and practical enterprise. However, we feel that it is essential for the success of such a system that it behave in a graceful way, such that the information available to the user is clear, concise, and relevant. With these ideas in mind, we have tried to emphasize those aspects of the system that make it effective to the user, namely responses in the form of fully descriptive sentences along with the displays, minimal information in the tables so as not to confuse the subject, and effective discourse capabilities. We would like to produce a system that can be a useful aid for providing the subject with the relevant information to select flights to be booked. It is our opinion that other aspects of the database, such as specific aircraft capabilities and ground transportation, should be de–emphasized at first, pending the successful closing of the loop to include spoken input within a narrower domain of booking flights. Overall however, we have found the process of developing an ATIS system to be rewarding and challenging. We look forward to the time when we will be able to work with spoken input, thus integrating the natural language component with a recognizer.

Acknowledgements

We would like to thank Lynette Hirschman, Don McKay, and Andy Glick of Unisys for their help in installing our database management system and providing us with IDIL software to interface with it.

References

[1] Asadi, A., R. Schwartz, and J. Makhoul, "Automatic Detection of New Words in a Large-Vocabulary Continuous Speech Recognition System," Proceedings, ICASSP 90, Albuquerque, NM, 3-6 April 1990, pp. 125-129.

[2] O'Hare, A. B., "The Intelligent Database Interface," Technical Report, Unisys Paoli Research Center, 1989.

[3] Seneff, S. "Probabilistic Parsing for Spoken Language Applications," paper presented at the International Workshop in Parsing Technologies, Pittsburgh, PA, 28-31 August 1989.

[4] Zue, V., N. Daly, J. Glass, D. Goodine, H. Leung, M. Phillips, J. Polifroni, S. Seneff, M. Soclof, "The Collection and Preliminary Analysis of a Spontaneous Speech Database," paper presented at the Second DARPA Speech and Natural Language Workshop, Harwichport, MA, 15-18 October 1989.

[5] Zue, V., J. Glass, D. Goodine, H. Leung, M. Phillips, J. Polifroni, and S. Seneff, "The VOYAGER Speech Understanding System: A Progress Report," paper presented at the Second DARPA Speech and Natural Language Workshop, Harwichport, MA, 15-18 October 1989.