Recent Improvements and Benchmark Results for the Paramax ATIS System

Lewis M. Norton, Deborah A. Dahl, and Marcia C. Linebarger *

Paramax Systems Corporation, a Unisys Company
P.O. Box 517
Paoli, PA, 19301

ABSTRACT

This paper describes three relatively domain-independent capabilities recently added to the Paramax spoken language understanding system: non-monotonic reasoning, implicit reference resolution, and database query paraphrase. In addition, we discuss the results of the February 1992 ATIS benchmark tests. We describe a variation on the standard evaluation metric which provides a more tightly controlled measure of progress. Finally, we briefly describe an experiment which we have done in extending the n-best speech/language integration architecture to improving OCR accuracy.

1. INTRODUCTION

In recent work on the Paramax spoken language understanding system we have focused on domain-independent capabilities whose relevance extends beyond the ATIS application. These include non-monotonic reasoning, implicit reference resolution, and the ability to create a simple paraphrase of a query for feedback to the user. Although these capabilities were motivated by ATIS data, they are likely to be required for processing data from other domains. We describe these improvements in the following section. In addition, we discuss our results on the ATIS benchmark tests. Because test data vary from test to test, it can be difficult to know whether fluctuations in performance are real, or are due to idiosyncracies in the particular test data selected. We have experimented with an evaluation paradigm in which systems used in earlier tests run the test data from the current test, thus controlling for the effect of variations in the test data. We suggest that this paradigm could provide a valuable supplement to the official tests.

2. SYSTEM IMPROVEMENTS

2.1. Non-Monotonic Reasoning

We previously described [1] a feature of the PUNDIT natural language processing system whereby the system makes inferences involving more than one decomposition.[1] For example, the instantiated decompositions produced for "flights leaving Boston" are:

```
flight_C(flight1, source(_), ...)
leaveP(leave1, flight(flight1),
                source(boston), ...)
```

Application of a rule relating the leaveP and the flight_C decompositions results in the source slot of the flight_C decomposition being instantiated to "boston".

We have extended this feature to make it possible to retract such inferences. This extension allows PUNDIT to do non-monotonic reasoning ([2]). That is, the system can make and reason from tentative inferences, which can be removed from the context if they are not supported by the developing dialog. The facility has been implemented in a fully general way, so that any test that can be coded in PROLOG can be the trigger for retraction.

Currently we use this capability to retract certain inferences which result in a database call with no answers. This facilitates better dialog processing. If the query *Do any flights from Boston to Denver serve dinner?* is answered by a list of one or more such dinner flights, the preferred antecedent of a subsequent reference to "those flights" is the set of those dinner flights. In contrast, if the answer to the query is "no", a subsequent reference to "those flights" refers to all flights from Boston to Denver. As explained in detail below, the ability to conditionally retract the inference enables our system to correctly identify the preferred antecedent in both cases.

In addition, this capability simplifies our system's processing of yes/no questions. The same inference rule applies to both *flights serving dinner* and *Do any flights serve dinner*; i.e., the rule makes no provision for distinguishing between these two contexts. Yet when they are embedded in a dialog, there are some differences. If a query such as *Show me flights from Boston to Denver serving dinner* revealed that there were no such flights,

*This paper was supported by DARPA contract N000014-89-C0171, administered by the Office of Naval Research, and by internal funding from Paramax Systems Corporation (formerly Unisys Defense Systems). We wish to thank Suzanne Taylor for helpful discussions on applying natural language constraints to OCR enhancement.

[1] A *decomposition* in PUNDIT is a frame-like structure created for most nominal concepts and for events, processes and states.

a subsequent query about "those flights" would seem rather odd. In contrast, as shown in the preceding paragraph, such a subsequent query can follow the yes/no question quite naturally.

The detailed processing of the example query

Do any flights from Boston to Denver serve dinner?

proceeds as follows. First a rule relating the decompositions for "flights" and "serve" causes the meals slot of the flight_C decomposition to be instantiated:

```
flight_C(flight2, source(boston),
                  goal(denver),
                  meals(dinner1), ...)
```

Then a database request is made for all dinner flights from Boston to Denver. If there are any, the flight_C decomposition as modified by the inference is retained; i.e., the context is left with a concept of *the dinner flights from Boston to Denver.* If there are no such flights, the inference is retracted, leaving in the context a concept of *(all) the flights from Boston to Denver.* In both cases, the correct concept is made available for subsequent reference resolution.

2.2. Implicit Reference Resolution

The pragmatics component in PUNDIT takes care of explicit reference resolution ([3]), as in *What is the cost of those flights?* But there are many cases where the reference to be resolved is implicit. A second extension to our system handles implicit reference resolution as in the following pair of queries:

Show me morning flights from Boston to Washington.

Show me afternoon flights.

We have implemented an ATIS-specific heuristic which addresses this need. It is invoked when our system is attempting to produce a database request for flights or fares but cannot find either the origin or destination (or both) in the current utterance. The heuristic allows the system to broaden its search for this information to earlier inputs. We have circumscribed this search in order to limit incorrect inferences; currently the heuristic works only in the following restricted manner:

The system finds the most recent flight entity in the discourse context other than the one explicitly involved in the current request, and checks that this entity satisfies two conditions:

a. If any origin or destination information is known about the current flights, the candidate entity must have no conflicting origin or destination information. So, for example, if a dialog proceeds

Show me flights from Boston to Philadelphia.

Show me the earliest flight from Boston.

the condition will be satisfied and the heuristic will apply, whereas for

Show me flights from Boston to Philadelphia.

Show me the earliest flight leaving Philadelphia.

it will not apply. Unfortunately, the heuristic currently will not apply in the following case, either:

Show me flights from Boston to Philadelphia.

Show me flights to Pittsburgh.

It would not be hard to refine the heuristic to apply to the above sequence, but we have not done so.

b. The query giving rise to the candidate entity must have been successfully processed, and must have received a non-null response. By successfully processed, we mean that a database request was made for the query. The system cannot tell, of course, if it was the correct request. But if no request was made, that is evidence that the earlier query either was not properly understood or that it was flawed in some way, and that it would be dangerous to use the candidate entity as a referent. Given the fact that our system currently fails to create database requests for over one-third of its inputs, taking this conservative approach turns out to be well-justified.

The requirement that the database request produce a non-null response is needed for cases such as:

Show me afternoon flights from Boston to San Francisco.

[there aren't any]

Show me flights on wide-body aircraft.

If the heuristic applied, it would create a request for afternoon flights from Boston to San Francisco on wide-body aircraft, and obviously none would be found.

If the candidate entity satisfies both of the above conditions, the non-conflicting properties of the current (origin or destination-deficient) entity and the candidate entity are merged. Thus for the pair of queries

Show me morning flights from Boston to San Francisco.

Show me flights on wide-body aircraft.

our system generates a request for morning flights from Boston to San Francisco on wide-body aircraft. However, for

Show me flights from Boston to San Francisco that serve lunch.

Show me dinner flights.

it asks for flights from Boston to San Francisco that serve dinner, not flights that serve both lunch and dinner.

If the candidate entity fails to satisfy both conditions, the heuristic simply fails; no other candidate entities are considered.

On the basis of training data, we predicted that the implicit reference resolution heuristic would apply to about 15 percent of discourse-dependent utterances. The recent benchmark test showed that our heuristic was more relevant than we expected, although it also turned out to be somewhat more error-prone. On the February 1992 natural language test, it resulted in the production of a database request for 48 class D utterances that otherwise would have been unanswered. 38 of these requests obtained the correct answer, 10 did not, so that the heuristic produced a net improvement in our overall score in spite of its unfortunately high level of errors. (It also was invoked on 4 class A queries, presumably inappropriately, although one of the four ended up with the correct answer!) There were 285 class D utterances in the test, so the heuristic was invoked for 16.8 percent of them. In addition, there was an undetermined number of other utterances for which it could have been invoked if our system had successfully processed the appropriate antecedent utterances.

2.3. Database Query Paraphrases

We have added a database query paraphrase capability to our ATIS system, which is used as follows.

When a database query is created by the system and the response received from the database, the query as well as the response is passed to an output formatting routine. At first this routine merely formatted the tabular response, but it turned out to be difficult for users to notice if the table displayed to them contained the desired information or not. Some of the time the system would mis-interpret the user's query, but the misinterpretation would go undetected. For example, a request for flights from Boston to Pittsburgh on Monday may have resulted in a table of all flights from Boston to Pittsburgh, and of a large number of flights, perhaps only one did not operate on Mondays. To address this shortcoming, we implemented a query paraphraser. The database query, after all, encodes what is actually retrieved from the database, so if we label the output table with a description of what it contains, any discrepancy between what the user requested and what the system provided can be spotted more easily. And in the majority of cases, when the system provided the desired output, the paraphrase served as a useful header to the table. For example, for the input sentence

Show me round-trip fares for flights from Boston to Denver leaving on Sunday.

the following paraphrase is produced.

```
Displaying:

Fare(s):
   - round-trip
   - on Sunday
   - for Flights:
      - from Boston
      - to Denver
```

As can be seen from the above example, the paraphrase is not in sentence form, but in a stylized form that is easy to read and understand. Sometimes it gives useful feedback concerning the system's interpretation of imprecise queries by the user, as in:

I need a flight from Boston to Pittsburgh that leaves early in the morning.

```
Displaying:

Flight(s):
   - departing before 8 am
   - from Boston
   - to Pittsburgh
```

And when an error is made, the user can notice it easily, particularly if told to check the easy-to-read paraphrases for missing conditions, as in this example from the October 1991 dry run test:

I want to travel from Atlanta to Baltimore early in the morning first flight.

```
Displaying:

Flight(s):
   - from Atlanta
   - to Baltimore
```

As in the above example, even though the system misinterprets the user's query, sometimes the desired answer can be gotten from the response produced, particularly with the guidance provided by the paraphrase of the response. In this way the system as a whole becomes more

capable of assisting the user in reaching a successful conclusion to the travel planning task.

3. BENCHMARK EVALUATION

Paramax undertook the February 1992 ATIS benchmark evaluation tests, with the cooperation of BBN, who provided us with speech recognizer output as described below. Our results were, in a word, disappointing. One component of the set of factors leading to this level of performance was our concentration on improvements to our system which were of applicability to spoken language understanding systems in general, as opposed to specific features that would only be applicable to the ATIS domain. But it is clear from the experience of this test that domain-specific features will have to be included if we are to perform in the ATIS domain. While we knew we were underemphasizing such features, we were somewhat taken by surprise since as recently as three months ago our level of performance had been comparable with other sites.

This latest test has not only gotten us thinking about the performance level of our system, but also about the subject of evaluation in general. The DARPA Spoken Language Understanding program, along with the MUC effort, has made significant advances in the state of the art of evaluation of language understanding systems ([4]). Particularly for the natural language community, these advances have given a new objectivity to the assessment of the level of achievement of their systems. Tests such as the ATIS benchmarks allow participants to find out how well their systems perform on suites of hundreds of utterances, not only in the simple absolute sense, but in relation to other similar systems.

The benchmark tests as administered, however, fall short of quantifying progress in any satisfactory sense. One might be tempted to claim that a history of scores on a series of similar tests gives an indication of progress or the lack thereof. It appears, however, that the only sense in which this might be meaningful is if one compares relative performance of different systems from one test to another. For example, our system, as remarked above, performed comparably to a number of other systems in past tests, yet did poorly in relation to those systems on this test. Yet that tells us nothing about whether our system improved, stagnated, or degraded.

The reason that the current common evaluation paradigm fails to quantify progress over time is that the test data varies from test to test. This variability tends to lessen the reliability of comparisons between system performance on different evaluations. That is, it is difficult to interpret variations in a single system's performance over time because we cannot quantify the effect of accidental differences in the particular test data that happens to be selected for a particular evaluation. Furthermore, the test paradigm has undergone a number of changes which make it difficult to compare results from evaluation to evaluation. For example, in June 1990, the test data included only Class A utterances. In February 1991, Class D1 utterances were included, but in a separate test. In the February 1992 test, Class A, D, and X utterances were all included together. In addition, in the current test scoring is being conducted under the min/max rules ([5]). All of these differences contribute to lessening the reliability of comparisons.

We have experimented with a more tightly controlled variation of the common evaluation metric in which the same test data is processed by several different versions of our system – the current version, and two older versions. These older versions correspond generally to the system which was reported on in February 1991 ([1]) and the system which we used to participate in an informal evaluation in October 1991.[2] By holding the data constant and varying the system, we eliminate the effect of data variations on scores. Furthermore, by comparing scores produced this way with the scores our system has received on the standard evaluations, we demonstrate that variations in the test data are a real concern, because we see a much less consistent pattern of development over time with the standard evaluation. Figure 1 shows how the scores on the February 1992 natural language test for the three different versions of the Paramax system varied.

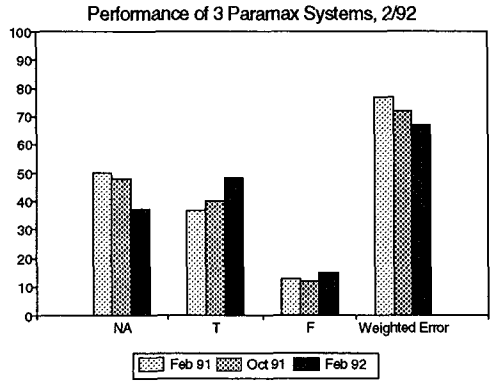

Figure 1: Comparative performance of systems.

From Figure 1 we can see that our system has made mod-

[2]The database revision released in May 1991 complicated matters somewhat. As we have reported elsewhere, our system has separate modules for natural language processing (PUNDIT) and database query generation (QTIP). We were forced to use the October 1991 QTIP with the February 1991 PUNDIT. Thus the performance labelled "February 1991" is really an overestimate, to whatever degree QTIP improved in that time.

est improvements over time, as shown by the decrease in weighted error rate. Additionally, the percentage of correctly answered (T) inputs has increased, while the percentage of unanswered (NA) inputs has decreased and the percentage of incorrectly answered (F) inputs has remained nearly constant. In contrast, if we compare our scores on the October dry run with the February 1992 benchmark test, we find that our system obtained a weighted error score of 64.1% on the October dry run when all utterances in classes A, D1, and D were considered. For the February benchmark, the corresponding figure was 66.7%. Breaking this down more finely, our class D error decreased from 97.9% to 83.9%, while our class A error increased from 47.4% to 54.5%. From this we might have concluded that our class D performance improved at the expense of our class A performance and overall system performance.

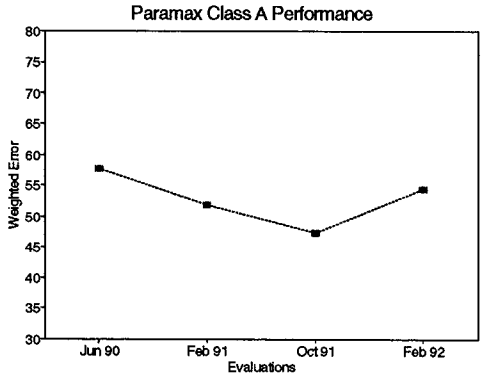

Figure 2: Comparative performance on class A tests.

Focusing only on class A utterances, we can go back farther in time, finding (Figure 2) that our system's error consistently decreased until the most recent test, which would reinforce the hypothesis that our class A performance degraded recently. Yet we see from Figure 3, which elaborates upon the information in Figure 1, that such a conclusion is unwarranted.

By running the same test with different versions of the understanding system, as described above, we obtain important information on the changes over time in system performance. This simple extension of the evaluation methodology already in place supplements comparisons between systems from different sites and comparisons between different tests with clear-cut documentation of the progress made by an individual site on its system. As such, it is a valuable tool to add to the ever-increasing arsenal of objective system evaluation techniques.

Table 1 summarizes our scores on the February 1992 natural language and spoken language benchmark tests. The SLS results were obtained by filtering nbest out-

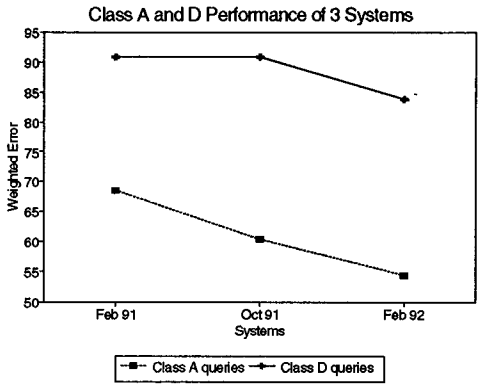

Figure 3: System performance on classes A and D.

put from BBN's speech recognition system through the Paramax natural language processing system, using an N of 6. This SLS score is only five percentage points inferior to our NL score. The corresponding difference for other sites ranged from ten to thirty-six points, with no correlation between actual scores and this difference. At this time we have no explanation for this intriguing phenomenon.[3]

Test	T	F	NA	Weighted Error
NL	331	102	254	66.7%
SLS	322	128	237	71.8%

Table 1: Paramax 1992 ATIS benchmark results.

3.1. Speech Recognition

The speech recognition scores which Paramax submitted for this evaluation were produced by using the Paramax natural language processing system to filter the n-best output from BBN's speech recognition system. The n-best output used in this evaluation had a word error

[3]The Paramax scores reported in this paper for the February 1992 benchmark tests are not the same as those disseminated by NIST. The NIST comparator is not guaranteed to score all utterances correctly. It first computes an estimate of the difficulty it will have in scoring an utterance, and if this estimate is too high, a score of F is automatically assigned, even though the answer may be correct according to the rules of min/max scoring. The difficulty parameter is R!/(R-H)!, where R is the number of columns in the maximal answer, and H is the number of columns in the system's answer. This figure must be less than $3 * 10^6$. Thus, for example, if the maximal answer has 15 columns, no more than 5 of them can appear in the system's answer. 20 of our answers on both the natural language test and the spoken language test were subject to this phenomenon. It is NIST's belief that no other site was similarly affected on more than 2 utterances, and they have given us permission to present scores which have been adjusted to account for comparator errors.

rate of 10.7%. N was set to 6 for this test. The natural language system selected the first candidate in the n-best which passed its syntactic, semantic, and application constraints, and output this candidate as the recognized utterance. If no candidate passed the natural language constraints, the first candidate of the n-best was output as the recognized utterance. After natural language filtering, the official speech recognition score was 10.6%. Although intuitively, natural language constraints should be able to reduce speech recognition error, they do not appear to do so in this case. There are at least three possible reasons for this outcome. One, the speech recognition is already quite good, consequently there is less room for improvement. Two, the natural language processing is not very good. Three, there is always going to be some residue of speech recognizer errors which result in perfectly reasonable recognized utterances, and no amount of natural language knowledge will be able to correct these. We have not explored these hypotheses in detail; however, we have done a related experiment with n-best data of a totally different kind, which shows a remarkably similar pattern. We briefly describe this experiment in the following section.

3.2. Optical Character Recognition

Although the nbest architecture was developed in the context of spoken language understanding, it is in fact applicable to any kind of input where indeterminacies in the input result in misrecognitions. In addition to speech recognizers, optical character recognition systems (OCR's) also have this property. Although current OCR technology is quite accurate for relatively clean data, accuracy is greatly reduced as the data becomes less clean. For example, faxed documents tend to have a high OCR error rate. Many OCR errors result in output which is meaningless, either because the output words are not legitimate words of the language, or because the output sentences do not make sense. We have applied linguistic constraints to the OCR problem using a variation of the N-best interface with a natural language processing system. Because the OCR produces only one alternative, an "alternative generator" was developed which uses spelling correction, a lexicon, and various scoring metrics to generate a list of alternatives from raw OCR output. Just as in the case of speech recognizer output, alternatives are sent to the natural language processing system, which selects the first candidate which passes its constraints.

The system was tested with 120 sentences of ATIS data on which the natural language system had previously been trained. The text data was faxed and then scanned. NIST speech recognition scoring software was used to compute word error rates. The word error rate for output directly from the OCR was 13.1%. After the output was sent through the spelling corrector, it was scored on the basis of the first candidate of the N-best set of alternatives. The error rate was reduced to 4.6%. Finally, the output from the natural language system was scored, resulting in a final average error rate of 4.2%.

Sending the uncorrected OCR output directly into the natural language system for processing without correction led to a 73% average weighted error rate. Spelling correction improved the error rates to 33%. Finally, with natural language correction, the weighted error rate improved to 28%. Thus, although improvements in word accuracy were minimal, application accuracy was greatly improved. This is consistent with previous experiments we have done on speech recognizer output [6]. Additional detail on this experiment can be found in [7].

Interestingly, training on the OCR data led to a processing time improvement. Comparing the performance of the October system and the February system, we found that total cpu time for processing the February test data was reduced by one-third. This improvement was due to improving processing inefficiencies which were noted during analysis of the processing of the OCR data. It is encouraging that the system is sufficiently general that training on OCR data can improve the processing of natural language and spoken language data.

References

1. L. M. Norton, M. C. Linebarger, D. A. Dahl, and N. Nguyen, "Augmented role filling capabilities for semantic interpretation of natural language," in *Proceedings of the DARPA Speech and Language Workshop*, (Pacific Grove, CA), February 1991.
2. M. L. Ginsberg, ed., *Readings in Nonmonotonic Reasoning*. Los Altos, California: Morgan Kaufmann, 1987.
3. D. A. Dahl and C. N. Ball, "Reference resolution in PUNDIT," in P. Saint-Dizier and S. Szpakowicz, eds., (*Logic and logic grammars for language processing*). Ellis Horwood Limited, 1990.
4. D. A. Dahl, D. Appelt, and C. Weir, "Multi-site natural language processing evaluation: MUC and ATIS," in *Proceedings of the Workshop on Natural Language Processing Evaluation*, (Berkeley, CA), June 1991.
5. MADCOW, "Multi-site data collection for a spoken language corpus," in *Proceedings of the DARPA Speech and Natural Language Workshop*, Morgan Kaufmann, Feb. 1992.
6. D. A. Dahl, L. Hirschman, L. M. Norton, M. C. Linebarger, D. Magerman, and C. N. Ball, "Training and evaluation of a spoken language understanding system," in *Proceedings of the DARPA Speech and Language Workshop*, (Hidden Valley, PA), June 1990.
7. D. A. Dahl, S. L. Taylor, L. M. Norton, and M. C. Linebarger, "Using natural language processing to improve OCR output," Tech. report, Paramax Systems Corporation. In preparation.