A PORTABLE APPROACH TO LAST RESORT PARSING AND INTERPRETATION

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

Paramax Systems Corporation
(a Unisys Company)
70 East Swedesford Road
Paoli, PA 19301

ABSTRACT

This paper describes an approach to robust processing which is domain-independent in its design, yet which can easily take advantage of domain-specific information. Robust processing is well-integrated into standard processing in this approach, requiring essentially only a single new BNF rule in the grammar. We describe the results of implementing this approach in two different domains.

1. Introduction

For best performance, natural language processing systems must be able to extract as much information as possible from their inputs, even inputs which cannot be fully processed. In order to do this, systems must be equipped with robust processing mechanisms. In addition, cases also occur in which the system has the ability to process an input, given sufficient time, but it is not desirable to allow unlimited amounts of processing time. In this paper we describe an approach to robust processing which is domain-independent in its general architecture, but which can be easily customized to particular domains by simply listing key words and/or key concepts. The approach uses the extensive grammar already available to the system for standard processing but augments it with a special BNF rule, called "backup", which is able to prune the wordstream while it searches for key concepts. Backup can be triggered either by a failure of normal parsing or by timing out. This approach has been implemented in two distinct domains. In one of these domains, when sufficient time is allotted to attain maximal performance, backup results in an 18% improvement in score. We describe the general approach, discuss how differences in the data in each domain lead to slightly different implementations, and discuss our results.

2. Approach

The approach to robust processing which is described in this paper is implemented in the PUNDIT natural language processing system developed at Paramax Systems Corporation [6, 1]. PUNDIT includes a domain-independent, top-down parser [7] which is the primary component involved in robust processing. The key feature of robust processing in PUNDIT is that the parser is allowed to skip over words when it is unable to find a parse using every word. Skipping is an appropriate strategy for the data in the two domains we are working with, because parsing failures tend to be due to extraneous material such as interpolated irrelevant comments and false starts. Another possible strategy, relaxation of constraints as suggested by [19], is less appropriate for the data we have examined, since few parsing failures are due to violation of grammatical constraints. Skipping over words has also been implemented in the robust parsing strategies of Seneff [15] and Strzalkowski [18]; our approach differs from these in that in addition to skipping, it provides a simple way of taking domain-specific knowledge into account in the skipping process. That is, when an analysis is not possible using every word, the system begins searching through the wordstream for keywords (or words denoting key concepts), which are simply listed in a file. The use of keywords permits the system to make use of the domain-specific knowledge that certain words or concepts are important in the domain. In fact, in a mature domain, the list of keywords and concepts can be automatically generated from the system's semantic interpretation rules.

Because the backup mechanism is implemented by adding a single new BNF rule into the normal grammar, robust processing has been implemented in PUNDIT without losing the advantages of the broad-coverage syntactic grammar already in the system. This is in contrast to approaches like the template matcher discussed in [8] or the frame combiner discussed in [16] which are completely separate mechanisms from the standard linguistic processing components.

In addition to inputs for which the system cannot find a parse using the standard algorithm, there are also cases where a complete analysis would be too costly in terms of time. The system can also invoke backup in these cases, using a variation of the timeout mechanism described in [17]. The timeout mechanism in [17] allocates an absolute amount of time per sentence; in contrast, PUNDIT's timeout allocates time as a function of the number of words in the input sentence so as not to penalize relatively longer sentences.

Previous approaches to robust processing have typically either focused solely on data from one domain [8, 16, 15, 4] or have implemented a domain-independent approach [17]. Both of these alternatives have disadvantages. Approaches which have been tested on only a single domain cannot be guaranteed to be extensible to other domains. Entirely new approaches may be required when the system is ported to another domain. On the other hand, the performance of domain-independent approaches may suffer in domain-specific applications because they are not able to use domain-specific knowledge to constrain the processing. Our approach differs from previous approaches in that, while the basic architecture is domain-independent, the approach also allows domain-specific knowledge to assist in the processing. We demonstrate the general applicability of the architecture by describing implementations in two distinct domains. Although the basic mechanism is the same in each domain, we also discuss differences in the implementation which follow from basic differences in the kind of data which must be processed.

3. Domains

We now briefly describe our two application domains, with emphasis on those properties of the domains which affect the details of implementing backup "last resort" processing.

3.1. Air Traffic Control

Air traffic control (ATC) involves oral communication, as controllers interact with pilots via radio, issuing commands which govern the movements of planes both on the ground and in the air [3]. Since the controllers are already speaking into microphones, their half of this dialogue is easy to capture in a high-quality signal. If this input can be understood, possible applications will range from intelligent indexing for archival purposes to real-time monitoring for safety and planning purposes.

Utterances in the ATC domain tend to be short sequences of relatively independent commands. The range of possible commands is well-bounded, and controllers are trained to avoid expressing these commands in different phrasings. As a consequence, it is possible to separate utterances into their constituent commands with high reliability, and similarly, to resume processing at the next command if processing of the present command fails for any reason. Also, some commands may be irrelevant for a given application. For example, wind advisories could be ignored by an application only concerned with ground operations.

A sample well-formed utterance follows:

Delta seven forty six turn right heading two seven zero cleared to land runway two nine left.

3.2. Air Travel Information System

Our second domain is called ATIS (Air Travel Information System) [12, 13, 11]. This is basically a database query application. The input utterances are retrieval requests addressed to a database of information about flight schedules, fares, etc. This application has been set up by DARPA as an infrastructure for research in spoken language understanding.

DARPA has arranged for the collection of data in this domain [5]. This data is spontaneous speech from naive users, who have no idea what phrasings will work and which will not. Thus, they use an extremely wide set of variations for each request, so that the system is expected to process inputs ranging from a vanilla *Show me flights from Boston to Denver* to *I am going to have to go to Denver; I will be leaving from Boston,* etc. Disfluencies are more prevalent in this domain, since the speakers are not trained users. Another feature distinguishing ATIS from ATC is that ATIS utterances, no matter how discursive they appear, normally constitute a single request. Therefore parse fragments created by the backup mechanism seldom correspond to individual commands as they do in the ATC domain; instead, a single request may give rise to several fragments which must be integrated during semantic and pragmatic processing[1].

In both domains, since the input is spoken, there is the additional possibility of errors introduced by the speech recognition component. While the techniques discussed in this paper have obvious applicability to recovery from such errors, in what follows we will assume perfection on the part of the recognizer, and that all errors and disfluencies originate with the speaker. Note, however, that current recognizers do not include punctuation in their output, either within sentences or at the end of them. We therefore have included no punctuation in our data.

4. Implementation

Grammars used with PUNDIT have at the top level a BNF rule for the "center" node. This rule is always a disjunction of possibilities; for example, in a toy grammar, the center rule might expand to either assertion or question. In typical application domains this rule is more complex, including perhaps compounds and/or fragments. One important fact about the disjuncts for the present discussion is that they are required to consume the whole

[1] A detailed discussion of PUNDIT's general approach to fragments can be found in [9].

input word string in order to succeed.

In any grammar, our approach to robust parsing is implemented by adding one additional disjunct at the end of the center rule. We call this disjunct "backup". The BNF rule for backup has the following form:

- If positioned at a keyword, reset the time allotment if necessary, then retry the other center options, *relaxing the requirement to consume the entire word string.* If a parse is found, call the center rule on the remainder of the word string.
- If not positioned at a keyword, or if a parse is not found in the previous step, *skip to the next keyword* if any, reset the time allotment if necessary, and call the center rule on the word string starting with the keyword. If no keyword is found, fail.

The backup rule is entered either if normal parsing fails (i.e., none of the other disjuncts of the center rule produce a parse consuming the whole word string), or if timeout occurs. Users specify an amount of time in the form of a number (possibly fractional) of seconds per word, so that longer inputs are given more time. Once time has expired, no rule will execute except the backup rule, which will reallot time based on the length of the remaining word string, and then proceed as described above.

The opportunity for introducing domain knowledge to influence the behavior of the backup rule comes in the specification of the keywords. To discuss what we have done in the two domains we experimented with, we first need to introduce the PUNDIT *knowledge base.* This is simply a mapping of word tokens to a hierarchical set of concepts [10]. Synonyms usually denote the same concept. The "is-a" relation is defined over the hierarchy, so that a concorde is-a jet is-a plane, a propeller_plane also is-a plane, etc.

The keywords used by backup can be specified as word tokens or as concepts. In the latter case, the concept is taken to refer to any word token that maps to the concept or any descendant of the concept in the knowledge base. Keywords may also be specified by syntactic category, e.g., determiners or tensed verbs may function as keywords.

4.1. Air Traffic Control

In the ATC domain, we designated only word tokens as keywords. Furthermore, the list of keywords was chosen manually with great care, and is not very extensive. The choices were dictated by the semantics of the possible commands which controllers may issue, and the normal phraseology (defined by the FAA) for expressing those commands. The intent, which we were able to achieve to a large degree, was to have skipping to the next keyword be equivalent to skipping to the start of the next command. Most of the keywords are verbs, corresponding to the imperative form most often used to express commands.

4.2. Air Travel Information System

In contrast, the list of keywords for the ATIS domain is much larger, and consists mostly of concepts, which in effect makes it even larger in terms of words. The basic idea is not to skip over any word which might be useful. Thus we included prepositions, wh-introducers, and such word tokens, plus all the concepts known to the PUNDIT semantic interpreter for that domain. This list of concepts was obtained mechanically from the files driving the interpreter, followed by the removal of concepts which were descendants of other concepts in the list, for these would be redundant for the purposes of the backup procedure. As a consequence, the only words skipped are meaningless (to the semantic interpreter), including unknown words.

An ATIS utterance normally constitutes a single database retrieval request. Therefore an additional step in this domain is to integrate the parse fragments obtained by the robust parsing procedure. We delegate this responsibility to the semantic and pragmatic interpreter [14, 2]. For those fragments which are complete sentences, no extensions are necessary. The interpreter merely treats them as distinct sentences coming in sequentially in the context of the ongoing dialogue.

For true fragments we did need to add some new capability. We assume that the overall content of the utterance is either a request for some flights or some fares. For noun phrase fragments, either the head is a flight or a fare, or it is not. If it is, our normal reference resolution capabilities are sufficient to resolve the flight or fare with any other flight or fare in the context[2]. If the head is not a flight or fare, flight and fare entities are explicitly generated into the context space maintained by the semantic interpreter, and the fragment is interpreted as a modifier of either the flight or the fare. Then normal reference resolution takes over. For example, the fragment *afternoon* ends up with the same semantic representation as does *afternoon flight*, and the system proceeds as before.

[2]This is because dialogues often proceed like the following: *Show me flights from Boston to Denver.* [answer] *Show me just afternoon flights.* So in effect, *afternoon flights* is treated as *show afternoon flights* [2, 13, 11].

Prepositional phrase fragments are treated in a manner completely analogous to noun phrase fragments whose heads are not flights or fares. For example, *in the afternoon* becomes *flight in the afternoon*, and the system proceeds as before.

The data for this domain has not warranted treatment of any other fragment types.

5. Results

5.1. Air Traffic Control

We performed experiments on a set of 233 utterances in the ATC domain, incorporating utterances from two different controllers. One was guiding planes which had just landed; the other was guiding planes as they taxied in preparation for takeoff.

Substantial benefits are gained from using backup, or "last-resort" processing, after normal parsing fails or a timeout occurs. Figure 1 shows that application accuracy is improved by the use of such processing, at two different settings of the timeout parameter. In fact, performance with backup at the lower timeout setting clearly exceeds performance without backup at the higher timeout setting. The improvement comes at a cost of increased cpu time, as can be seen in Figure 2; the increase is less for the higher value of the timeout parameter, even though the benefit to accuracy remains high.

Figure 1: Effect of backup on score

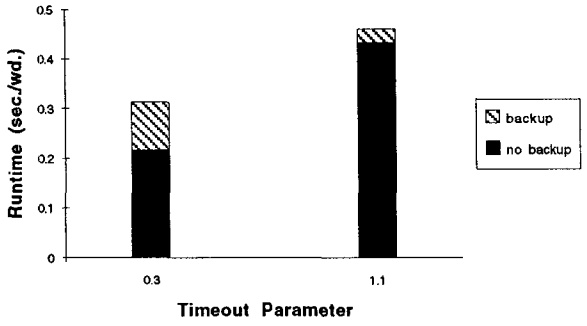

Figure 2: Effect of backup on runtime

We investigated the effects of varying the timeout parameter when backup processing is in use. Recall that this parameter is the amount of cpu time allotted for each word of an utterance before timeout. Backup processing resets this allotment, adjusted for the current position in the utterance, so that the amount of time spent processing an utterance can increase by a factor of two to four over the initial allotment, depending on the number of keywords in the utterance.

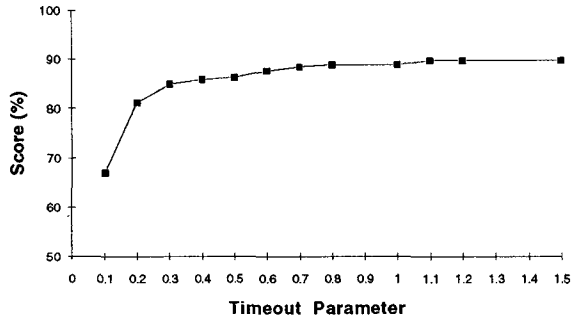

Figure 3: Effect of varying timeout

Figure 3 shows the results of our investigation. A setting of the timeout parameter below 0.3 is clearly undesirable. A setting of 0.3 enables the system to process correctly all but a handful of the utterances it could handle at a higher setting; that is, the curve changes at this point to a nearly horizontal orientation. At a setting of 1.1, the system achieves maximal performance accuracy. Somewhat surprisingly, if the utterances of each controller are considered separately, these findings remain the same, even though the content and phrasing of the utterances vary noticeably.

The optimal setting of the timeout parameter depends on the relative costs of processing time and application errors. The 0.3 setting might be optimal for archival purposes or high volume processing. The 1.1 setting might be necessary for applications which demand maximal accuracy at any cost.

5.2. Air Travel Information System

As examples of data which our system handles properly, we list some inputs which are successfully processed. All of them were previously unseen test data from the November 1992 ATIS test. None of them would result in a parse from normal parsing. These inputs include false starts, corrections, constructions not covered by our grammar, and breaks in parsing due to unknown words. Our technique contributes for all these phenomena.

I would like to do you have any flights between Philadelphia and Atlanta (false start)

Okay shoot I would have to choose the Delta flight nine seventy seven departing at twelve pm and arriving in San Francisco at two ten pm shoot and *choose* were unknown words, but the system recovers and understands the chosen flight.

Okay American Airlines does it leave Philadelphia for Dallas in the mornings Left dislocation, not in our grammar; the airline is parsed as a fragment separate from the main body of the question, and semantic processing integrates the two parses correctly.

Yes could you please give me a list of all American Airline first class flights to from Philly to Dallas Fort Worth please The correction of the preposition at *to from Philly* is successfully handled by our technique; *to* is dropped, and parse fragments are produced for the rest of the input starting at *from*.

Quantitative Results Because the semantic integration of fragmentary information is still in progress, the robust processing mechanism did not affect our final score on the ATIS evaluation. However, we did look closely at the effect of robust processing on parsing accuracy, in order to answer the following two questions:

- How much does the backup mechanism improve parsing accuracy?
- How often does the backup mechanism do the right thing?

In order to answer the first question, we compared the proportion of usable or potentially usable non-X parses which the system produced with and without backup on the subset of the 1992 ATIS test collected at BBN. Without backup, 77% of the parses were usable; with backup, 88% were usable. Thus, backup resulted in an 11% increase in the number of usable parses.

In order to answer the second question, we looked at the parses produced by backup. We found that 45% of them were usable or potentially usable by semantics. Of the parses that were not usable, we found that most of the time they were unusable because the system did not have information about some semantically important word in the sentence. Because of this missing information, the system ended up ignoring the word, and consequently the parse did not contain this important word. The fact that many of the unusable parses were due to lexical gaps was encouraging, because it means that the backup mechanism will continue to improve in this respect simply as new words are added to the system in the normal course of development.

6. Conclusion and Future Directions

We have described a domain-independent approach to robust processing which can be customized to particular domains through the simple mechanism of building a list of keywords, where keywords may correspond to specific word tokens, syntactic categories, or semantic concepts. The approach was tested in two different domains, ATC and ATIS. Differences in the way that information is conveyed in the two domains necessitated slight differences in the implementations across the two domains; in particular, additional semantic processing was required in the ATIS domain to put together information from the fragmentary outputs of the parser. Future plans include development of keyword selection techniques, both domain-independent and domain-specific; and improvements to the semantic integration process.

References

1. Deborah A. Dahl. PUNDIT – natural language interfaces. In G. Comyn, N.E. Fuchs, and M.J. Ratcliffe, editors, *Logic Programming in action*, Heidelberg, Germany, September 1992. Springer-Verlag.
2. Deborah A. Dahl and Catherine N. Ball. Reference resolution in PUNDIT. In P. Saint-Dizier and S. Szpakowicz, editors, *Logic and logic grammars for language processing*. Ellis Horwood Limited, 1990.
3. Deborah A. Dahl, Lewis M. Norton, and Nghi N. Nguyen. Air traffic control instruction monitoring using spoken language understanding. In *Proceedings of the 36th Air Traffic Control Association Meeting*, Atlantic City, NJ, November 1992.
4. Philip J. Hayes and George V. Mouradian. Flexible parsing. *American Journal of Computational Linguistics*, 7(4):232–242, 1981.
5. Charles T. Hemphill, John J. Godfrey, and George R. Doddington. The ATIS spoken language systems pilot corpus. In *Proceedings of the DARPA Speech and Language Workshop*, Hidden Valley, PA, June 1990.
6. L. Hirschman, M. Palmer, J. Dowding, D. Dahl, M. Linebarger, R. Passonneau, F.-M. Lang, C. Ball, and C. Weir. The PUNDIT natural-language processing system. In *AI Systems in Government Conf.* Computer Society of the IEEE, March 1989.
7. Lynette Hirschman and John Dowding. Restriction grammar: A logic grammar. In P. Saint-Dizier and S. Szpakowicz, editors, *Logic and Logic Grammars for Language Processing*, pages 141–167. Ellis Horwood, 1990.
8. Eric Jackson, Douglas Appelt, John Bear, Robert Moore, and Ann Podlozny. A template matcher for robust NL interpretation. In *Proceedings of the DARPA Speech and Natural Language Workshop*. Morgan Kaufmann, February 1991.
9. Marcia C. Linebarger, Deborah A. Dahl, Lynette Hirschman, and Rebecca J. Passonneau. Sentence fragments regular structures. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, NY, June 1988.
10. David L. Matuszek. K-Pack: A programmer's interface to KNET. Technical Memo 61, Unisys Corporation, P.O. Box 517, Paoli, PA 19301, October 1987.
11. Lewis M. Norton, Deborah A. Dahl, and Marcia C. Linebarger. Recent improvements and benchmark results for the Paramax ATIS system. In *Proceedings of the DARPA Speech and Language Workshop*, Harriman, New York, February 1992.

12. Lewis M. Norton, Deborah A. Dahl, Donald P. McKay, Lynette Hirschman, Marcia C. Linebarger, David Magerman, and Catherine N. Ball. Management and evaluation of interactive dialog in the air travel domain. In *Proceedings of the DARPA Speech and Language Workshop*, Hidden Valley, PA, June 1990.

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

14. Martha Palmer. *Semantic Processing for Finite Domains.* Cambridge University Press, Cambridge, England, 1990.

15. Stephanie Seneff. A relaxation method for understanding spontaneous utterances. In *Proceedings of the DARPA Speech and Natural Language Workshop.* Morgan Kaufmann, February 1992.

16. David Stallard and Robert Bobrow. Fragment processing in the DELPHI system. In *Proceedings of the Speech and Natural Language Workshop*, San Mateo, California, 1992. Morgan Kaufmann.

17. Tomek Stralkowski. TTP: a fast and robust parser for natural language. Technical report, New York University Department of Computer Science, New York, NY, 1991.

18. Tomek Strzalkowski and Barbara Vauthey. Information retrieval using robust natural language processing. In *Proceedings of the Thirtieth Annual Meeting of the Association for Computational Linguistics*, pages 104–111, 1992.

19. R. M. Weischedel and N. K. Sondheimer. Meta-rules as a basis for processing ill-formed input. *American Journal of Computational Linguistics*, 9(3-4):161–177, 1983.