Robust Parsing of Noise Contaminated and Extra-grammatical Input: A Grammar Focused Approach

Alon Lavie

Thesis Proposal

Abstract

This thesis tackles the problem of parsing noise contaminated input by identifying and parsing the maximal subset of the input string that is found to be grammatical. I develop a parser that is based on the Generalized LR Parsing paradigm and performs this task efficiently. Since the parser uses the grammar to identify the meaningful words of the input, it can be viewed as a focusing tool.

The parser accommodates the skipping of words by allowing shift operations to be performed from previous states of the Graph Structured Stack. To overcome the potentially exponential number of parsable subsets of the original input string, I intend to develop several approximation heuristics. Additional heuristics will be developed to rank the final parses that are found by the parser and select the most desirable one.

I will demonstrate how the robust parser can be used to parse spontaneous speech provided by the output of existing speech recognition systems. I will also investigate how the parser can be used in the area of text extraction. In this application the grammar can be used to specify a language of meaningful concepts. Fast coarse text extraction techniques can be used as a preliminary filter when scanning large volumes of text. The parser can then be used to analyze and interpret the selected sentences.

1 Introduction

In the last few years, speech recognition systems have reached a fairly high level of performance. It is now conceivable to construct systems where speech is the main medium of input. It is likely that spoken input will become the primary source of input in sophisticated tasks such as dictation, database information retrieval, and speech to speech translation.

One example of such a system is *JANUS*, a speech-to-speech translation system, developed here at CMU in collaboration with Siemens and the University of Kalrsruhe (Germany), and ATR (Japan) [Waibel *et al.*, 1991a, Waibel *et al.*, 1991b]. The system performs recognition of spoken input from the source language (English, German or Japanese), translates it into one of the other languages, and produces speech synthesized output. System performance on a restricted domain of conference registration dialogues has been encouraging.

Any useful computer task that is designed to process spoken input beyond the recognition phase requires some level of analysis to extract meaning from the utterance. Syntactic parsing is a common first step in this process. As a form of input, spontaneous speech has characteristics that distinguish it from written text. It is full of noise and irrelevances, such as repeated words, stalls and false beginnings. Some of these can be filtered out by the speech recognizer, but some degree of noise will remain, and must be dealt with by the language understanding component of the system.

When we humans communicate with one another, we usually have no trouble filtering the utterance spoken to us and extracting the meaningful words from the noise. Unfortunately, today's parsers are to a large extent still fragile to noise and contamination in the input stream, and break down when presented with input that is not "clean". This is not surprising, since parsers for both computer and natural languages were originally designed to recognize grammatical input by rejecting any input that is not strictly in the language described by the grammar.

Grammar coverage is another problematic issue in parsing natural languages. Designing a natural language grammar with reasonable coverage is a difficult task even in restricted domains. Most researchers today agree that it is hopeless to expect a predesigned grammar to cover the full richness of language likely to be encountered by spontaneous spoken input. On the other hand, designing a grammar that covers the most frequent and important language constructs of a particular domain, although not trivial, is considered a feasible task.

Conventional parsers are incapable of handling any input that is not covered by the grammar in even the slightest way. For example, if the grammar does not allow the word "please" to appear at the beginning of a sentence, any input in which the speaker started his sentence with "please" would be rejected by the parser, even if the rest of the sentence is completely grammatical. What is needed in this case is a parser that can extract the meaningful grammatical parts of the utterance and ignore the extra grammatical parts.

I suggest that parsers can be made robust to input contaminated by noise, by detecting and parsing the maximal subset of words of the input that is parsable according to the grammar. In this setting, the grammar serves as a tool for focusing on the meaningful parts of the input. Naively, this could be done by attempting to parse all possible subsets of the input string, and choosing the largest such subset that was successfully parsed. However, since the number of subsets is exponential in the length of the input, such an approach is clearly computationally infeasible.

In this thesis, I will develop a parser that performs this task efficiently, in terms of both practical and theoretical (asymptotic) run time. The parser I develop is a modified version of the Generalized LR Parser [Tomita, 1986] that has proven to be a popular and efficient parsing framework for natural language, in practice as well as in theory. I will demonstrate the parsing capabilities of the

new parser on spontaneous speech input, obtained from existing speech recognition systems in the community.

Another potential area of application of this robust parsing method is in the area of text extraction. I intend to investigate how the parser, coupled with a strict grammar that defines the language of interest, can be used for extracting meaningful sentences from large volumes of text.

2 The Problem of Parsing Spontaneous Speech

As a form of input, spontaneous speech is full of noise and irrelevances that surround the meaningful words of the utterance. A parser that is designed to successfully process this form of input must be robust to these forms of noise, and be able to weed out the meaningful words from the rest of the utterance.

When parsing spontaneous spoken input, the parser must deal with three major types of noise:

1. Noise due to the spontaneity of the speaker.
2. Ungrammaticality that is due to the language of the speaker, or to the coverage of the grammar.
3. Noise due to errors of the speech recognizer.

For a better understanding of the problems facing the parser, let us look in more detail into these different types of noise.

2.1 Noise due to the spontaneity of the speaker

This class of noise contains the various phenomena in spontaneous speech that interfere in the fluency of the sentence. Typical forms of this kind of noise are repeated words, false beginnings, stutters, and various filled pauses (such as "ah","um" etc.). Such phenomena are speaker dependent, and vary greatly from one speaker to another. However, my hope is that the parser can be made robust by ignoring the occurrences of such noise, regardless of the speaker. Here are some examples, taken from transcriptions of spontaneous conversations in the conference registration domain, that demonstrate these phenomena. Each example is accompanied by a "clean" content version of the same sentence, which is the ideal goal for the parser to actually detect and successfully parse.

- False beginning, filled pauses:
  - Original input: `um okay then yeah I am disappointed.`
  - Content input: `I am disappointed.`
- Repetition, filled pauses:
  - Original input: `uh I I sent a check in on about the first of May.`
  - Content input: `I sent a check in on the first of May.`
- Stutter, filled pauses:
  - Original input: `uh yes I would like to register the for for the conference.`
  - Content input: `yes I would like to register for the conference.`

2.2 Ungrammaticality

The first source of problems of ungrammaticality is the limited coverage of the grammar. Any prewritten grammar is bound to fail in covering the wide range of spoken language used by different speakers, even when restricted to a specific domain. Utterances that are grammatical in the large sense may fail to be covered by the grammar in their entirety. The best we can hope is for the grammar to cover a wide variety of expected grammatical constructs, and thus enable the parser to succeed in parsing the core part of the utterance, and ignore the rest as noise. Here are some examples, taken from a test bed of conference registration sentences of the JANUS system. Each example is a sentence that fails to be completely parsable by the grammar. Each sentence is accompanied by the best parsable subset of the original sentence.

- Unrecognized verb form:
  - Original input: `Are you able to help me make a reservation at a hotel.`
  - Parsable subset : `Help me make a reservation at a hotel.`
- Unrecognized prepositional phrase:
  - Original input: `Will any speakers be participating in the tour of the city.`
  - Parsable subset : `Will any speakers be participating in the tour.`
- Unrecognized relative clause:
  - Original input: `I will send you the form to fill out`
  - Parsable subset : `I will send you the form.`
- Unrecognized verbs and part of a compound noun:
  - Original input: `I'm expecting to attend the AI conference but I don't have a registration form.`
  - Parsable subset : `I'm to the conference but I don't have a registration form.`
- Unrecognized polite manner at end of WH question:
  - Original input: `What is your name please.`
  - Parsable subset : `What is your name.`

The second source of ungrammaticality arises due to the nature of the speaker. People seem to adhere less to rules of the grammar when they speak (as opposed to when they write). The most we can expect of a parser in this case is to extract and parse the parts of the utterance that are grammatical. This, of course, will not be sufficient in many cases, but should still provide some useful additional information that can be used for clarification purposes by a deeper understanding component of the system. Furthermore, if users are made aware to the sensitivity of the system to general grammaticality, they can be expected to speak in a reasonable grammatical manner with only a minor infringement on their spontaneity.

2.3 Noise due to errors of the speech recognizer

Even though the quality of speech recognition systems has improved substantially over the last several years, the best output of the recognizer can be expected to include misrecognized words on a rather frequent basis. Grammar based parsing is very sensitive to such misrecognitions. This can be turned into an advantage when efficiency considerations allow operating in N-best mode, where the recognizer outputs a list of the N best recognitions (for some small N), and the parser parses the recognized outputs in order, searching for one which is grammatical.

Tests on the JANUS system [Waibel *et al.*, 1991a, Jain, 1991] have shown that in N-best mode, an LR parser provides better performance than a connectionist parser by using the grammaticality criteria to detect the actual utterance from the list of outputs provided by the recognizer. In cases where all recognitions are corrupt in one way or another, we may still be able to parse a major subset of the utterance.

3 Context within Previous Work on Robust Parsing

3.1 Previous Approaches

There have been several different approaches to robust parsing in the last decade. Most of these approaches to some extent abandon syntax and grammar as the major tools for parsing the input robustly when faced with various extra-grammaticalities.

Hayes and Mouradian [Hayes and Mouradian, 1981] were among the first to try to tackle these kind of problems. They developed a pattern matching parser that parsed the input in a bottom-up fashion. Robustness is achieved by allowing parses to be suspended when faced with unexpected input, and then reactivated with later input words.

Carbonell and Hayes [Carbonell and Hayes, 1984], in an early work that attempts to deal with a wide range of extra-grammatical phenomena, suggest that a semantic case-frame approach to parsing is the suitable way to handle the majority of these problems. Semantic interpretation of the input is achieved by detecting the main verb of the sentence, and then searching the sentence for components that instantiate the semantic frames that are associated with the particular verb. A similar approach has been adapted in more recent systems designed to parse and understand spoken input, such as the PHOENIX system [Ward, 1991, Ward *et al.*, 1992].

Other systems such as BBN's DELPHI system [Stallard and Bobrow, 1992] and the MIT ATIS system [Seneff, 1992] construct fallback components that are designed to handle extra-grammatical input that causes the main parser to fail. The DELPHI system incorporates both syntax and frame-based fragment combination components. In the MIT system, grammatical constraints are relaxed when an input fails to parse in full, and the system attempts to combine partial parsed fragments, in search of the parse that consumes the most words of the original sentence.

One notable attempt to handle ill-formed input by a general purely syntactic method is presented by Mellish [Mellish, 1989]. Mellish describes how to augment a chart parser with a procedure that can focus on an error in the input once the parser has failed. The error is detected by performing a top-down search through the chart left behind by the parser. Although the method can potentially handle missing words as well as misspelled and extra words, it is demonstrated to work reasonably well only in the case of a single error, and is not expected to generalize well to multiple errors.

3.2 Advantages of the Robust GLR Approach

The major drawback of the previous approaches to robust parsing outlined above is their lack of generality. The semantic based case-frame approach is particularly domain dependent, and at times can be somewhat shallow and inaccurate. In contrast, the robust GLR approach is a general one. Because the parsing algorithm is a modification of the standard GLR context-free parsing algorithm, all of the techniques and grammars developed for the standard parser can be applied as they are. In particular, the standard LR parsing tables are compiled in advance from the grammar and used "as is" by the parser in runtime. The robust parser inherits the benefits of the original parser in terms of ease of grammar development, and, to a large extent, efficiency properties of the parser itself. In the case that the input sentence is by itself grammatical, the parser behaves exactly as the standard GLR parser.

In the context of parsing spoken input, previous researchers have noted correctly that while syntax driven formulations provide strong linguistic constraints and useful structures for further linguistic analysis, they tend to be very fragile to extra-grammaticalities. On the other hand, semantic driven approaches can potentially provide better coverage and handle ill formed sentences, but provide less constraint for the speech recognizer, and may fail to adequately interpret more complex linguistic structures. The robust GLR approach therefore tries to achieve the "best of both worlds". It attempts to maximize the ability of the parser to handle various forms of extra-grammaticality while remaining strictly within a formal syntactic setting. This allows the system to enjoy the full benefits that the syntax based formalism has to offer. At the same time, this does not exclude the employment of further system and domain specific methods to enhance the parse result by processing fragments of the input that were rejected by the parser.

The techniques that I develop for enhancing the GLR parser to overcome extra-grammatical input can be generally applied to other chart-based parsers. In particular, heuristics developed for choosing the best parse and for limiting the parser to a beam search should prove to be directly applicable to other similar parsers. It is worth noting however that the efficiency of the developed robust parser is due in part to several particular properties of the GLR parser, and may thus not be easily transferred to other syntactic parsing formalisms.

4 LR and GLR Parsing

The LR parsing techniques are efficient and powerful methods for parsing context free languages. LR parsers parse the input bottom-up, scanning the input left to right, producing a right-most derivation. They are deterministic and efficient, being driven by a table of parsing actions pre-compiled from the grammar.

Unfortunately, it is not possible to construct deterministic LR parsing tables for all context free grammars. In particular, ambiguous grammars, such as those typically used to describe the syntax of natural language, are not LR and thus result in tables that contain entries with multiple parsing actions.

Tomita's Generalized LR parsing algorithm [Tomita, 1986] extended the original LR parsing algorithm to the case of nondeterministic parsing tables while preserving the efficiency of the parser. Tomita's algorithm uses a Graph Structured Stack (GSS) in order to efficiently pursue in parallel the different parsing possibilities that arise as a result of the multiple entries in the parsing tables. A second data structure uses pointers to keep track of all possible parse trees throughout the parsing of the input, while sharing common subtrees of these different parses.

(1) S → NP VP
(2) NP → det n
(3) NP → n
(4) NP → NP PP
(5) VP → v NP
(6) PP → p NP

Figure 1: A Simple Natural Language Grammar

5 The Modified Robust Parser

The modified parsing algorithm is an extension of the Generalized LR Parser, as implemented in the Universal Parser Architecture developed at CMU [Tomita and Carbonell, 1987]. This implementation incorporates an SLR(0) parsing table.

The parser accommodates skipping words of the input string by allowing shift operations to be performed from previous states in the GSS. Shifting an input symbol from a back state is equivalent to skipping the words of the input that were encountered after the parser reached the back state and prior to the current word being shifted. Since the parser is LR(0), reduce operations need not be repeated for skipped words (the reductions do not depend on any lookahead). Information about skipped words is maintained in the symbol nodes that represent parse sub-trees.

An initial version of the robust parser has been implemented in Lucid Common Lisp, in the integrated environment of the Universal Parser Architecture. Some preliminary test results are discussed in later sections.

5.1 An Example

To clarify how the proposed word skipping parser actually works, in lieu of a more formal description of the algorithm itself, I will present a step by step runtime example. For the purpose of the example, we use a simple natural language grammar that is shown in Figure 1. The terminal symbols of the grammar are depicted in lower-case, while the non-terminals are in upper-case. The grammar is compiled into an SLR(0) parsing table, which is displayed in Table 1. Note that since the table is SLR(0), the reduce actions are independent of any lookahead. The actions on states 10 and 11 include both a shift and a reduce.

To understand the operation of the parser, we now follow some steps of the robust parsing algorithm on the input `x = det n v n det p n`. This input is ungrammatical due to the second `det` token. The maximal parsable subset of the input in this case is the string that includes all words other than the above mentioned `det`.

In the figures ahead, which graphically display the GSS of the parser in various stages of the parsing process, we use the following notation:

- An *active* (top level) state node is represented by the symbol "@", with the state number indicated above it. Actions that are attached to the node are indicated above the state number.

- An *inactive* state node is represented by the symbol "*". The state number is indicated above the node and the actions to the right of the node.

	Reduce	Shift					Goto			
State		det	n	v	p	$	NP	VP	PP	S
0		sh3	sh4				2			1
1						acc				
2				sh7	sh8			5	6	
3			sh9							
4	r3									
5	r1									
6	r4									
7		sh3	sh4				10			
8		sh3	sh4				11			
9	r2									
10	r5				sh8				6	
11	r6				sh8				6	

Table 1: SLR(0) Parsing Table for Grammar in Figure 1

```
0                                   after initialization
@ sh3                               (and empty reduce phase)
```

Figure 2: Initial GSS

- Grammar symbol nodes are represented by the symbol "#", with the grammar symbol itself displayed above it.

The parser operates in phases of shifts and reductions. We follow the GSS of the parser following each of these phases, while processing the input string. Reduce actions are distributed to the active nodes during initialization and after each shift phase. Shift actions are distributed after each reduce phase. Note that the robust parsing algorithm distributes shift actions to *all* state nodes (both active and inactive), whereas the original parser distributed shift actions only to active nodes. Figure 2 is the initial GSS, with an active state node of state 0. Since there are no reduce actions from state 0, the first reduce phase is empty. With the first input token being `det`, the shift action attached to state node 0 is `sh3`.

Figure 3 shows the GSS after the first shift phase. The symbol node labeled `det` has been shifted and connected to the initial state node and to the new active state node of state 3. Since there are no reduce actions from state 3, the next reduce phase is empty. The next input token is `n`. Shift actions are distributed by the algorithm to both the active node of state 3 and the inactive node of state 0, as can be seen in Figure 3.

Figure 4 shows the GSS after the next shift phase. The input token `n` was shifted from both state nodes, creating active state nodes of states 9 and 4. The shifting of the input token `n` from state 0 corresponds to a parsing possibility in which the first input token `det` is skipped. Reduce actions are distributed to both of the active nodes.

The following reduce phase reduces both branches into noun phrases. The two `NP`s are packed together by a local ambiguity packing procedure. Using information on skipped words that is maintained within the symbol nodes, the ambiguity packing can detect that one of the noun

Figure 3: GSS after first shift phase

Figure 4: GSS after second shift phase

phrases (the one that was reduced from `det n`) is more complete, and the other noun phrase is discarded. The resulting GSS is displayed in Figure 5. Shift actions with the next input token `v` are then distributed to all the state nodes. However, in this case, only state 2 allows a shift of `v` into state 7.

Figure 6 shows the GSS after the third shift phase. The state 7 node is the only active node at this point. Since no reduce actions are specified for this state, the fourth reduce phase is empty. Shift actions with the next input token `n` are distributed to all state nodes, as can be seen in the figure.

Figure 7 shows the GSS after the fourth shift phase and Figure 8 after the fifth reduce phase. Note that there are no active state nodes after the fifth reduce phase. This is due to the fact that none of the state nodes produced by the reduce phase allow the shifting of the next input token `det`. The original parser would have thus failed as this point. However, the robust parser succeeds in distributing shift actions to two inactive state nodes in this case.

For the sake of brevity we do not continue to further follow the parsing step by step. The final GSS is displayed in Figure 9. Several different parses, with different subsets of skipped words are actually packed into the single `S` node seen at the bottom of the figure. The parse that corresponds

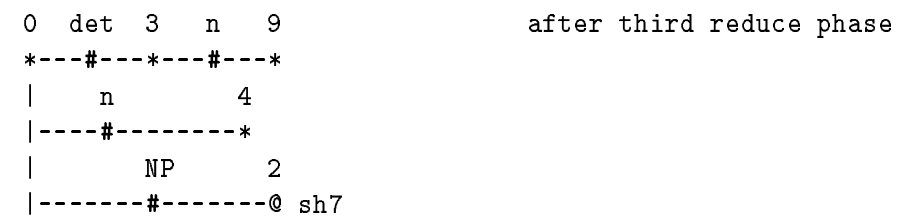

Figure 5: GSS after third reduce phase

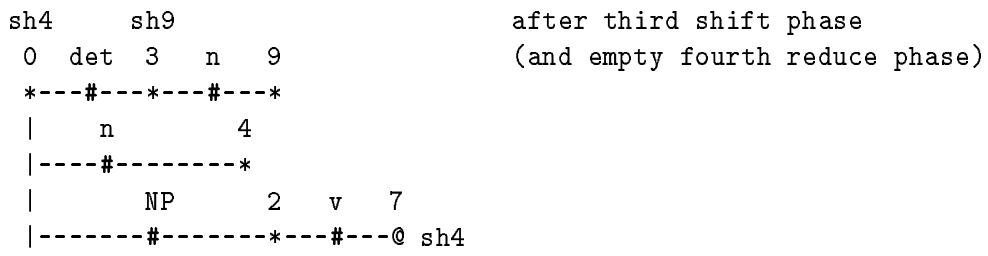

Figure 6: GSS after third shift phase

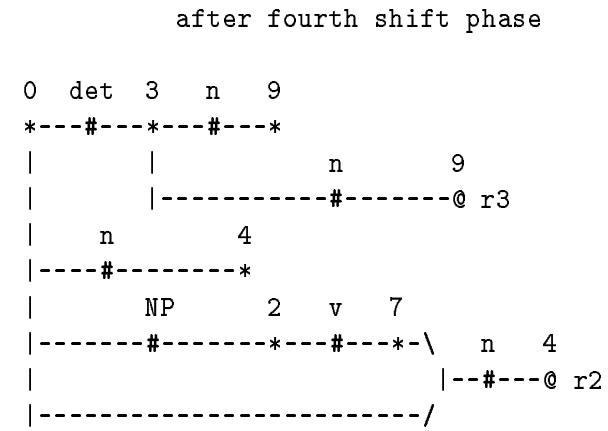

Figure 7: GSS after fourth shift phase

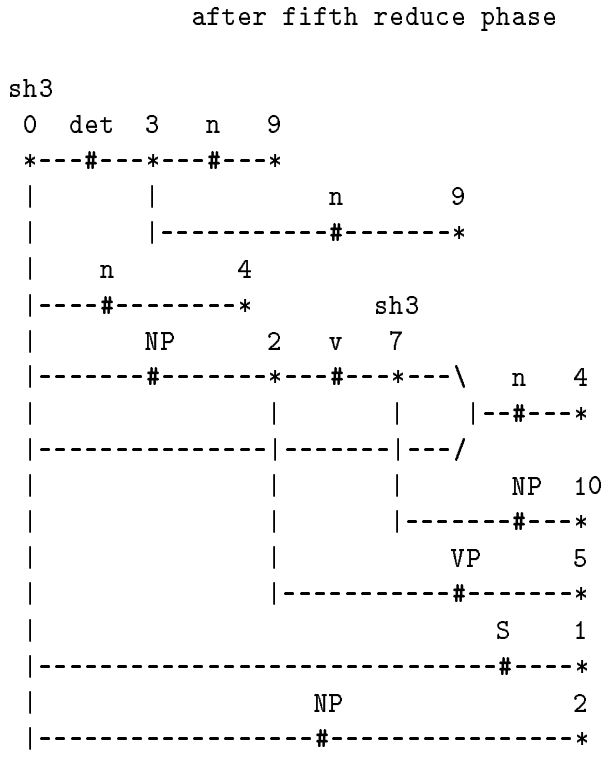

Figure 8: GSS after fifth reduce phase

to the maximal subset of the input is the one in which the second `det` is the only word skipped.

5.2 Efficiency of the Parser

Efficiency of the parser is achieved by a number of different techniques. The most important of these is a sophisticated process of local ambiguity packing and pruning. A local ambiguity is a part of the input sentence that corresponds to a phrase (thus, reducible to some non-terminal symbol of the grammar), and is parsable in more than one way. The process of skipping words creates a large number of local ambiguities. For example, the grammar in Figure 1 allows both determined and undetermined noun phrases (rules 2 and 3). As seen in the example presented earlier, this results in the creation of two different noun phrase symbol nodes for the definite noun phrases. The first node is created for the full phrase after a reduction according to the first rule. A second symbol node is created when the determiner is skipped and a reduction by the second rule takes place.

Locally ambiguous symbol nodes are detected as nodes that are surrounded by common state nodes in the GSS. The original GLR parser detects such local ambiguities and packs them into a single symbol node. This procedure was extended in the modified parser. Locally ambiguous symbol nodes are compared in terms of the words skipped within them. In cases such as the example described above, where one phrase has more skipped words than the other, the phrase with more skipped words is discarded in favor of the more complete parsed phrase. This subsumption operation drastically reduces the number of parses being pursued by the parser.

Another technique employed to increase the efficiency of the parser is the merging of state-vertex nodes of the same state and level after the reduce-actions phase and after the shift-actions phase. This allows the parsing through the GSS to continue with fewer state-vertex nodes.

6 Ranking the Parse Results and Selecting the Best Parse

In principle, we are interested in finding only the maximal parsable subset of the input string (and its parse). To accomplish this, we use a filtering procedure at the end of the parsing process to select the maximal parse from the set of parses returned by the parser. Currently only a simple filtering procedure has been implemented. It prunes the set of parsed input string subsets, and discards all but those that are minimal in terms of the total number of words skipped. The remaining parses are filtered through an additional procedure that prunes and selects among ambiguous parses. Presently, the sole criteria used in this ambiguity filtering is minimality in the number of sentences that are indicated by the parse. I intend to explore various methods and heuristics to enhance the process of selecting the most desired parse from the parser's set of outputs.

One approach to this problem is to use probabilistic information to rank the parse results. This can be done by constructing a probabilistic version of the parser. Wright and others [Wright and Wrigley, 1989, Wright *et al.*, 1991, Wright, 1990] have already developed the theoretical framework for the construction of a probabilistic LR parser. First, probabilities are attached to the rules of the grammar. A modified algorithm for constructing the parsing tables is then used to pre-compile the parsing tables, where rule probabilities induce probabilities on actions in the parsing tables. These probabilistic parsing tables are at times significantly larger in size than the original tables. The probabilistic parser then uses the probabilistic information in the parsing tables to attach a score to the parse trees and partial parse trees constructed in the parsing process. The scores of the final parses can then be used to rank them. Hopefully, there exists a direct correlation between the score of a parse and its desirability, so selecting the parse with the best score will produce the most desirable parse.

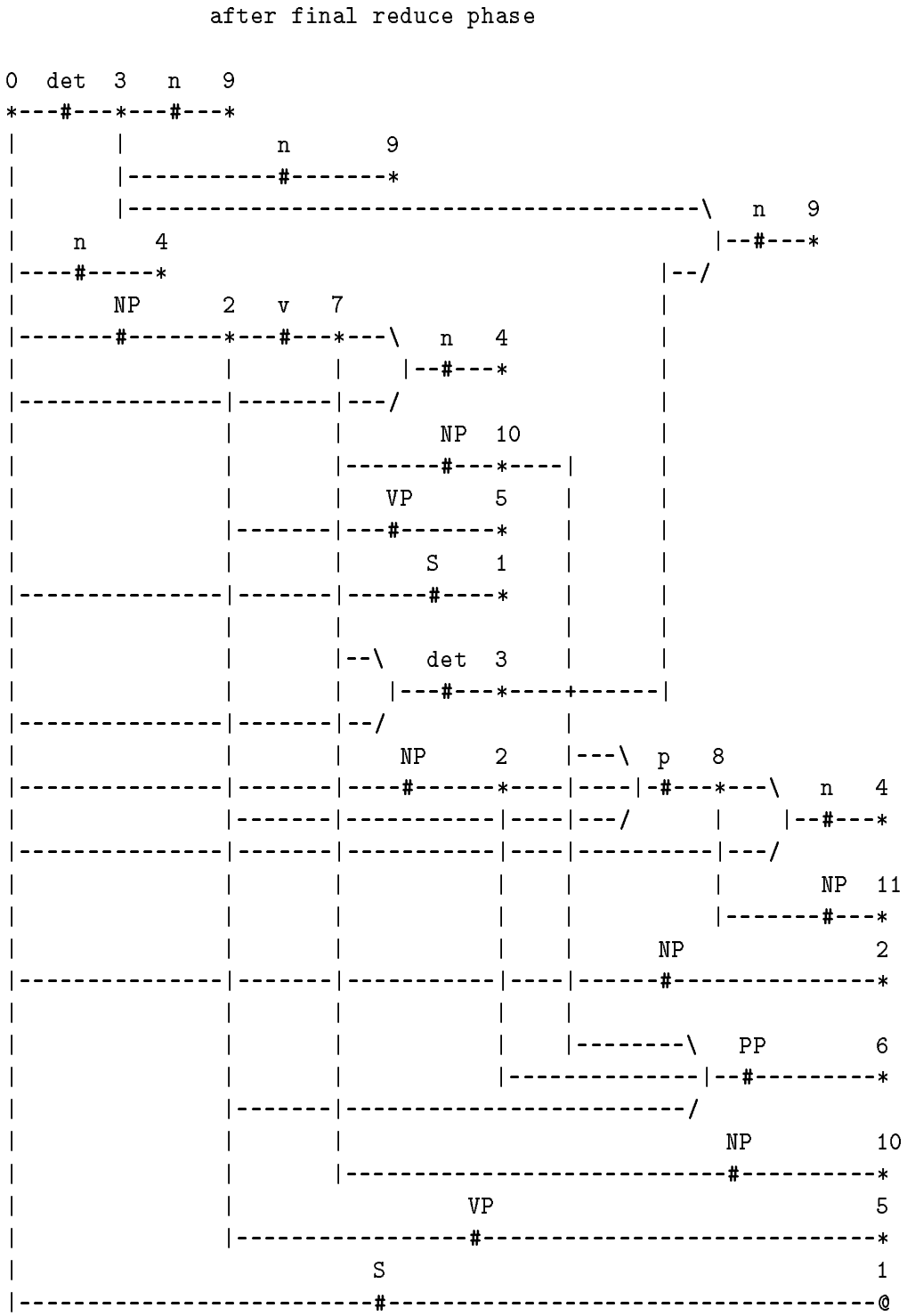

Figure 9: GSS after final reduce phase

I would like to develop a more sophisticated heuristic scheme that combines the probabilistic parse ranking with various other ranking heuristics for selecting the most desired parse. In this particular setting, there is little value to precise notions of parse tree probabilities, and we are primarily interested in relative probabilistic information. Since there are overwhelming advantages to retaining the basic framework of the original GLR parser "as is", I will devise a method to augment the constructed parse trees with probabilistic information without modifying this basic framework. I will then explore other heuristic measures (such as minimality in the total number of sentences that the input string was parsed into), and combine the different measures into a single ranking scheme.

A good scheme for ranking the parse results is needed for the additional purpose of discriminating between good and bad parse results. The original GLR parser, although fragile, was highly likely to return correct parse results when it succeeded in parsing the input. The robust parser, on the other hand, returns some parse result in almost all cases, thus concealing an explicit indication of failure. I will attempt to develop some additional heuristic methods that can discriminate between good and bad parse results. The ranking of the parses and other characteristics will be used.

7 Making Things Run Faster - Adding some Heuristics

Although implemented efficiently, the basic parser described earlier is still not guaranteed to have an acceptable running time. The above algorithm computes parses of all parsable subsets of the original input string, the number of which is potentially exponential in the length of the input string. Our goal is, in principle, to find the maximal parsable subset of the original input string (or a subset that is close to maximal, but is preferable due to some other criteria).

I intend to explore several different approximation heuristics that offer major savings in the amount of computation, while enabling the parser, in most cases, to still find the maximal parsable subset (and other large parsable subsets).

One such simple heuristic is to fix the number of consecutive words allowed to be skipped by the parser to some constant k. This limit may be fixed by the user at runtime.

A more powerful heuristic is to add some kind of "beam search" to the parser. The idea in this case is to limit the number of active-vertex nodes pursued by the parser at each stage. Only nodes that correspond to parses with a minimal number of skipped words (or those that are close enough) are pursued, while the others are discarded.

Heuristics such as these reduce the runtime of the robust parser to within a constant factor of the original GLR parser. Although they are not guaranteed to find the desired maximal parsable subset of the input string, they are designed to function acceptably well in most practical cases.

I intend to evaluate the performance of these and other heuristics in terms of the tradeoff that they offer between parser robustness and runtime.

8 Preliminary Results on Parsing Spontaneous Speech

The primary and most direct application of the robust parser is in parsing the output produced by speech recognizers in general, and spontaneous speech input in particular. I have conducted a number of preliminary tests to evaluate the performance of the robust parser on these types of input.

	Original Parser		Robust Parser	
	number	percent	number	percent
Parsable	71	61%	114	97%
Unparsable	46	39%	3	3%
Good/Close Parses	61	52%	82	70%
Bad Parses	10	9%	32	27%

Table 2: Performance of the Robust Parser vs. the Original Parser

8.1 Parsing Coverage

The first test expanded a comparative evaluation conducted by Ajay Jain as part of his thesis [Jain, 1991]. In the original comparative study, Jain compared the performance of a GLR parser with that of PARSEC, a connectionist parser he developed in his thesis. The GLR parser used in this evaluation was the Universal Parser Architecture [Tomita and Carbonell, 1987], the same parser that is the basis of the robust parser.

Jain compared the performance of PARSEC with the performance of the GLR parser with three different grammars. The purpose of the test was to compare PARSEC parsing coverage capabilities with those of the GLR parser. The grammars that were used were developed to cover the domain of a conference registration task. The common test set was a set of 117 sentences from the conference registration task. These sentences contained no spontaneous speech noise. Jain reports that PARSEC succeeded in parsing 67% of the test set sentences (78% including near misses), whereas the GLR parser with the best grammar returned correct parses in only 38% of the sentences (39% including near misses). The GLR parser managed to actually parse 54 of the 117 sentences (46%), but these included 8 sentences with bad parses. Fully 63 out of the 117 sentences (54%) were unparsable by the GLR parser.

I tested my robust parser on the same set of 117 sentences. Unfortunately, the exact grammar that Jain used in the original test was unavailable. However, the grammar I used for the test is an upgraded version of the best grammar Jain used in his comparison. I thus compared the performance of the original GLR parser using this grammar with that of the robust parser. The results are shown in Table 2.

The results indicate that using the robust parser results in a significant improvement in performance. The percentage of sentences, for which the parser returned good or close parses increased from 52% to 70%, an increase of 18%. Fully 97% of the test sentences (all but 3) are parsable by the robust parser, an increase of 36% over the original parser. However, this figure includes a significant increase (from 9% to 27%) in the number of bad parses found. Thus fully half of the additional parsable sentences of the set return with bad parses. Compared with the robust parser, the original GLR parser, although fragile, returned results of relatively good quality, when it succeeded in parsing the input. This indicates a strong need in the development of methods for discriminating between good and bad parse results. I will try and develop some effective heuristics to deal with this problem.

8.2 Parsing Noisy Spontaneous Speech

The second test I conducted was to evaluate the performance of the robust parser on noisy sentences typical of spontaneous speech. The robust parser was tested on a set of 100 sentences of spontaneous

	Robust Parser
	number (and percent)
Parsable	99
Unparsable	1
Good/Close Parses	77
Bad Parses	22

Table 3: Performance of the Robust Parser on Spontaneous Speech

speech transcriptions. These transcriptions include a multitude of noise in the input. The following example is one of the sentences from this test set:

```
``fsma2_7 /h#/ okay {comma} I have found {comma} *pause* your {comma} *pause*
records here {comma}''
```

The grammar used in the previous test was used for this test as well. Since the test sentences were drawn from actual speech transcriptions, they were not guaranteed to be covered by the grammar. The test thus challenges the parser with problems of both noisy input and grammar coverage. However, to ensure meaningful results, sentences with verbs and major nouns that were beyond the vocabulary of the grammar were avoided. Also pruned out of the test set were short opening and closing sentences (such as "goodbye"). The performance results are presented in Table 3. Note that due to the noise contaminating the input, the original parser is unable to parse a single one of the sentences in this test set.

The results of both preliminary performance tests are very encouraging. They suggest that the robust parser being developed will be very effective as a means for parsing speech recognized output in general, and spontaneous speech in particular. I intend to conduct experiments of a larger scale to determine the effectiveness of the robust parser in such settings. I will use an environment and data provided by the JANUS speech-to-speech translation system as primary tools for this evaluation. Thorough performance tests of the final parser will be conducted in this and other settings.

9 Application : Text Extraction

Another potential application of the robust parser is in the area of text extraction. Text extraction is the process in which large volumes of textual data are scanned in search of particular units of text that are of interest. A text extraction system must include the means for specifying the interesting units being searched for, as well as an efficient algorithm for searching the large volumes of text being processed.

The robust parser can be used to enhance the performance and capabilities of a text extraction system. Let us assume that the units of text for which we are searching are sentences that contain interesting keywords and phrases. In this setting, we can construct a context-free grammar to specify the set of interesting phrases. Traditional fast pattern matching algorithms can then be used to extract potentially interesting sentences from the large volume of text being processed. This can be viewed as an initial "coarse" stage. In the second "fine" stage, the robust parser can be applied to the smaller collection of sentences that were selected in the first stage, in order to detect and analyze the sentences that contain interesting phrases (according to the language specified by the grammar).

This proposed approach to text extraction is in its preliminary stages, and many questions are still open. In particular, it is not clear whether there are practical advantages in using the grammar as a specification tool. I will investigate this and other issues. I plan to integrate the robust parser into the TIPSTER project, an existing text extraction system being developed at the Center for Machine Translation here at CMU.

10 Research Plan

My plan of research at this time is the following:

- March 1993: Complete initial implementation.
- April 1993: Develop some simple initial heuristics for choosing the best parse, and for determining parse quality, based on the performance evaluations on speech, that I have already conducted.
- May-September 1993: Conduct a first round of experiments with the parser on data in the areas of spontaneous speech and text extraction, using the environments provided by the JANUS and TIPSTER projects. These evaluations will be conducted fairly in parallel, for mutual benefit. Included in this period will be the major development of the software needed for integrating my parser in these environments.
- October-November 1993: Work on developing the ranking hueristics for choosing the best parse. Combine the various heuristics into an integrated ranking scheme.
- December 1993 - January 1994: Develop the heuristics for evaluating the quality of the parse result.
- February 1994: Work on improving the search heuristics.
- March-May 1994: Final round of evaluations on both spontaneous speech and text extraction, and construction of the final version of the parser. In parallel with writing the thesis itself.
- June 1994: Thesis defense.

11 Expected Contributions

I expect this thesis to add the following major contributions:

1. The development of an efficient robust general parsing architecture, capable of handling extra-grammatical and noise contaminated input by detecting and parsing the "best" (usually largest) subset of the input string that is found to be grammatical.
2. Demonstrate the effectiveness of such a parser in handling spontaneous speech input, by substantially enhancing the capabilities of systems that process spoken input.
3. The construction of a new approach to the text extraction problem, which uses context-free grammars as a specification tool and the robust parser as an extraction mechanism.

References

[Carbonell and Hayes, 1984] J. G. Carbonell and P. J. Hayes. Recovery Strategies for Parsing Extragrammatical Language. *Technical Report CMU-CS-84-107*, 1984.

[Hayes and Mouradian, 1981] P. J. Hayes and G. V. Mouradian. Flexible Parsing. *Computational Linguistics*, 7(4):232–242, 1981.

[Jain, 1991] A. N. Jain. PARSEC: A Connectionist Learning Architecture for Parsing Spoken Language. *Technical Report CMU-CS-91-208*, 1991.

[Mellish, 1989] C. S. Mellish. Some Chart Based Techniques for Parsing Ill-formed Input. In *Proceedings of Annual Meeting of the Association of Computational Linguistics (ACL-89)*, pages 102–109, 1989.

[Seneff, 1992] S. Seneff. A relaxation method for understanding spontaneous speech utterances. In *Proceedings of DARPA Speech and Natural Language Workshop*, pages 299–304, February 1992.

[Stallard and Bobrow, 1992] D. Stallard and R. Bobrow. Fragment processing in the DELPHI system. In *Proceedings of DARPA Speech and Natural Language Workshop*, pages 305–310, February 1992.

[Tomita and Carbonell, 1987] M. Tomita and J. G. Carbonell. The Universal Parser Architecture for Knowledge-based Machine Translation. *Technical Report CMU-CMT-87-101*, 1987.

[Tomita, 1986] M. Tomita. *Efficient Parsing for Natural Language*. Kluwer Academic Publishers, Hingham, Ma., 1986.

[Waibel *et al.*, 1991a] A. Waibel, A. N. Jain, A. McNair, J. Tebelskis, L. Osterholtz, H. Saito, O. Schmidbauer, T. Sloboda, and M. Woszczyna. JANUS: Speech-to-Speech Translation Using Connectionist and Non-Connectionist Techniques. *Advances in Neural Information Processing Systems*, 4, 1991.

[Waibel *et al.*, 1991b] A. Waibel, A. N. Jain, A. E. McNair, H. Saito, A. G. Hauptmann, and J. Tebelskis. JANUS: A Speech-to-Speech Translation System Using Connectionist and Symbolic Processing Strategies. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 1991.

[Ward *et al.*, 1992] W. Ward, S. Issar, X. Huang, H. Hon, M. Hwang, S. Young, M. Matessa, F. Liu, and R. Stern. Speech understanding in open tasks. In *Proceedings of DARPA Speech and Natural Language Workshop*, pages 78–83, February 1992.

[Ward, 1991] W. Ward. Understanding spontaneous speech: The Phoenix system. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 365–367, April 1991.

[Wright and Wrigley, 1989] J. H. Wright and E. N. Wrigley. Probabilistic LR parsing for speech recognition. In *Proceedings of International Workshop on Parsing Technologies '89*, pages 105–114, 1989.

[Wright *et al.*, 1991] J. Wright, A. Wrigley, and R. Sharman. Adaptive probabilistic generalized LR parsing. In *Proceedings of International Workshop on Parsing Technologies '91*, pages 100–109, 1991.

[Wright, 1990] J. Wright. LR parsing of probabilistic grammars with input uncertainty for speech recognition. *Computer Speech and Language*, 4:297–323, 1990.