GLR* :
A Robust Grammar Focused Parser for Spontaneously Spoken Language

Thesis Summary

Alon Lavie

August 15, 1995

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Thesis Committee:
Masaru Tomita, Chair
Jaime Carbonell
Alex Waibel
Edward Gibson, MIT

Abstract

The analysis of spoken language is widely considered to be a more challenging task than the analysis of written text. All of the difficulties of written language can generally be found in spoken language as well. Parsing spontaneous speech must, however, also deal with problems such as speech disfluencies, the looser notion of grammaticality, and the lack of clearly marked sentence boundaries. The contamination of the input with errors of a speech recognizer can further exacerbate these problems. Most natural language parsing algorithms are designed to analyze "clean" grammatical input. Because they reject any input which is found to be ungrammatical in even the slightest way, such parsers are unsuitable for parsing spontaneous speech, where completely grammatical input is the exception more than the rule.

This thesis describes GLR*, a parsing system based on Tomita's Generalized LR parsing algorithm, that was designed to be robust to two particular types of extra-grammaticality: noise in the input, and limited grammar coverage. GLR* attempts to overcome these forms of extra-grammaticality by ignoring the unparsable words and fragments and conducting a search for the maximal subset of the original input that is covered by the grammar. The parser is coupled with a beam search heuristic, that limits the combinations of skipped words considered by the parser, and ensures that the parser will operate within feasible time and space bounds.

The developed parsing system includes several tools designed to address the difficulties of parsing spontaneous speech. To cope with high levels of ambiguity, we developed a statistical disambiguation module, in which probabilities are attached directly to the actions in the LR parsing table. The parser must also determine the "best" parse from among the different parsable subsets of an input. We thus designed a general framework for combining a collection of parse evaluation measures into an integrated heuristic for evaluating and ranking the parses produced by the GLR* parser. This framework was applied to a set of four parse scoring measures developed for the JANUS scheduling domain and the ATIS domain. We added a parse quality heuristic, that allows the parser to self-judge the quality of the parse chosen as best, and to detect cases in which important information is likely to have been skipped.

To demonstrate its suitability to parsing spontaneous speech, the GLR* parser was integrated into the JANUS speech translation system. Our evaluations on both transcribed and speech recognized input have indicated that the version of the system that uses GLR* produces between 15% and 30% more acceptable translations, than a corresponding version that uses the original non-robust GLR parser. We also developed a version of GLR* that is suitable to parsing word lattices produced by the speech recognizer, and investigated how lattice parsing can potentially overcome errors of the speech recognizer and further improve end-to-end performance of the speech translation system.

1. Introduction

The analysis of spoken language is widely considered to be a more challenging task than the analysis of written text. All of the difficulties of written language (such as ambiguity) can generally be found in spoken language as well. However beyond these difficulties, there are additional problems that are specific to the nature of spoken language in general, and spontaneous speech in particular. The major additional issues that come to play in parsing spontaneous speech are speech disfluencies, the looser notion of grammaticality that is characteristic of spoken language, and the lack of clearly marked sentence boundaries. The contamination of the input with errors of a speech recognizer can further exacerbate these problems.

Most natural language parsing algorithms are designed to analyze "clean" grammatical input. By the definition of their recognition process, these algorithms are designed to detect ungrammatical input at the earliest possible opportunity, and to reject any input that is found to be ungrammatical in even the slightest way. This property, which requires the parser to make a complete and absolute distinction between grammatical and ungrammatical input, makes such parsers fragile and of little value in many practical applications. Such parsers are thus unsuitable for parsing spontaneous speech, where completely grammatical input is the exception more than the rule.

This thesis describes GLR*, a parsing system based on Tomita's Generalized LR parsing algorithm, that was designed to be robust to two particular types of extra-grammaticality: noise in the input, and limited grammar coverage. GLR* attempts to overcome these forms of extra-grammaticality by ignoring the unparsable words and fragments and conducting a search for the maximal subset of the original input that is covered by the grammar.

Because the GLR* parsing algorithm is an enhancement to the standard GLR context-free parsing algorithm, grammars, lexicons and other tools developed for the standard GLR parser can be used without modification. GLR* uses the standard LR parsing tables that are compiled in advance from the grammar. It inherits the benefits of GLR in terms of ease of grammar development, and, to a large extent, efficiency properties of the parser itself. In the case that an input sentence is completely grammatical, GLR* will normally return the exact same parse as the standard GLR parser.

Although it should prove to be useful for other practical applications as well, GLR* was designed to be particularly suitable for parsing spontaneous speech. Grammars developed for spontaneous speech can concentrate on describing the structure of the meaningful clauses and sentences that are embedded in the spoken utterance. The GLR* parsing architecture can facilitate the extraction of these meaningful clauses from the utterance, while disregarding the surrounding disfluencies.

1.1. Extra-grammaticalities in Spontaneous Speech

1.1.1. Disfluencies in Spontaneous Speech

Because spontaneous speech is an "on-line" form of communication, it contains many kinds of disfluencies. The typical forms of disfluencies that can be found in spontaneous speech are false beginnings, repeated words, stutters, and various filled pauses (such as "ah","um" etc.). Disfluencies are speaker dependent, and vary greatly from one speaker to another. However, the parser should be able to disregard most such disfluencies, regardless of the speaker. Here are some examples,

taken from transcriptions of spontaneous conversations in the scheduling domain, that demonstrate these phenomena. For each example we show the original utterance along with a "clean" version of the utterance, that contains just the meaningful content and omits the disfluencies. The "clean" versions correspond to what we would like the parser to detect and successfully parse.

- False beginning, filled pauses:
  - Original input: `um okay then yeah Monday at two is fine.`
  - Content input: `Monday at two is fine.`
- Repetition, filled pauses:
  - Original input: `uh I I need to meet with you next week.`
  - Content input: `I need to meet with you next week.`
- Stutter, false start, filled pauses:
  - Original input: `October the first I have a meeting on the uh I have a meeting at ten to eleven in the morning.`
  - Content input: `October first I have a meeting at ten to eleven in the morning.`

1.1.2. Ungrammaticality

One source of ungrammaticality is the limited coverage of the grammar. Any pre-written 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 one can hope for 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, once again taken from recorded dialogs in the scheduling domain. Each example is an utterance that fails to be completely parsable by a recent version of our grammar. Along with each utterance, we present the most complete subset of the utterance that is parsable by the grammar.

- Unrecognized proper noun:
  - Original input: `I'm sorry Lynn.`
  - Parsable subset : `I'm sorry.`
- Unrecognized compound noun:
  - Original input: `I'm busy during those lunch hours from one to four on the twenty seventh.`
  - Parsable subset : `I'm busy during lunch from one to four on the twenty seventh.`

- Unrecognized matrix clause:
  - Original input: `I just realized that I'm too busy on the first.`
  - Parsable subset : `I'm too busy on the first.`
- Unrecognized beginning:
  - Original input: `Listen, how about Thursday December second.`
  - Parsable subset : `How about Thursday December second.`
- Unrecognized verb form:
  - Original input: `I can't do lunch with you but we can meet later.`
  - Parsable subset : `I can't lunch with you but we can meet later.`

Another source of ungrammaticality arises due to the nature of the spoken language. People seem to adhere less to rules of the grammar when they speak (as opposed to when they write). Even so, the clauses and sentences that comprise the meaningful content of spoken utterances are for the most part structurally well formed, and can thus be considered "grammatical". The grammars that are developed for parsing spontaneous speech can describe the structure of these meaningful parts of the utterance. In cases where parts of the utterance deviate from what would be considered grammatical according to the grammar, the most we can expect of a parser is to extract and parse the parts of the utterance that are grammatical.

1.1.3. Dealing with Errors of the Speech Recognizer

Even though the quality of speech recognition systems has improved substantially over the last several years, a speech recognizer that handles spontaneous speech will frequently misrecognize words of the input utterance. Grammar based parsing is very sensitive to such misrecognitions. However, the speech recognition process produces multiple hypotheses for a given speech utterance. It is often the case that a highly accurate hypothesis exists, even though it is not the one considered best by the recognizer. Since a more accurate hypothesis is likely to be more grammatical and thus produce a better parse, we can hopefully use the parser to detect a hypothesis with fewer recognition errors. GLR* is suitable for this purpose, since it can parse hypotheses even if they are not completely grammatical. In Chapter 7 of the thesis, we investigate how this can be done efficiently using a lattice parsing version of GLR*.

2. Research Goals

This thesis was primarily motivated by the problem of parsing spontaneously spoken language. As illustrated in the examples of the previous section, spontaneous speech is often ill-formed. The spontaneous "on-line" nature of this type of input often results in disfluencies that for the most part can be regarded as noise that contaminates the meaningful segments of the utterance. Furthermore, the expression forms used by speakers are likely to exceed the language coverage of the grammar. However, we make the observation that the "core" parts of spoken utterances can be grammatically

analyzed, and that in most cases, this analysis can reflect the essential meaning being conveyed. We therefore set out to demonstrate that a parser that can focus on the maximal grammatically analyzable subset(s) of a spoken input utterance, can substantially reduce the problems of noise contamination and limited grammar coverage, and that this results in a substantial improvement in the parser's ability to correctly analyze spontaneously spoken language.

We chose to base the developed parsing architecture on a standard efficient natural language parser, so that issues of robustness will be separated as much as possible from the development of the grammars, lexicons and other linguistic tools, and so that existing tools and grammars will be usable "as is". The GLR parsing system was chosen as an appropriate parsing architecture, since it is an efficient well developed architecture that has been in extensive use for large practical machine translation applications in our local research environment at CMU.

The major research goals set for the thesis were the following:

1. **To develop an efficient robust parsing algorithm**, that can find and parse the maximal grammatical subset of a given input utterance. Furthermore, **to couple the parsing algorithm with heuristics for search control**, that will allow it to perform within feasible time and space bounds.

2. **To develop a statistical disambiguation module**, that can effectively assist in selecting the correct analysis from the set of ambiguous analyses produced by the parser for a particular parsed input. An additional sub-goal was to develop a method that will allow the statistical scores assigned by the disambiguation module to be used in the evaluation of *different* parsable subsets of the input utterance, so that the "best" parse found by the parser can be selected.

3. **To develop a general framework for parse evaluation**, that can evaluate and rank sets of competing parse analyses that correspond to different grammatical subsets of the input, and which is capable of combining a variety of knowledge sources and heuristic evaluation measures. Additionally, **to develop a set of parse evaluation measures** that are suitable for evaluating the parses produced by the GLR* parser in the domain in which we have applied it - spontaneously spoken scheduling dialogs. Furthermore, in some cases, GLR* will have to skip input segments with crucial information. Thus, parsability alone is insufficient for ensuring that the analysis is meaningful. We therefore set out **to develop a parse quality heuristic**, that will allow the parser to self-judge the quality of the "best" analysis found, using the developed parse evaluation measures.

4. **To investigate the effectiveness and practical feasibility of the developed GLR* parser** in parsing spontaneous speech input, by integrating it into JANUS, a large speech-to-speech-translation project, and by conducting extensive evaluations of the end-to-end performance of the system using GLR*.

5. **To construct a version of the robust GLR* parser that is capable of parsing lattices produced by the speech recognizer**, to integrate it into a lattice processing version of the JANUS system, and to investigate the ability of this architecture to overcome speech recognition errors and to improve the end-to-end performance of the system.

3. Thesis Summary

3.1. Foundation: GLR Parsing

The GLR* parsing algorithm developed in this thesis is based on The Generalized LR (GLR) parsing algorithm that was developed by Tomita. The GLR parsing algorithm and the foundations of the LR parsing method are described in Chapter 2 of the thesis.

3.1.1. Principles of LR Parsing

The GLR parsing algorithm evolved out of the LR parsing techniques that were originally developed for parsing programming languages in the late 1960s and early 1970s. LR parsers parse the input bottom-up, scanning it from left to right, and producing a rightmost derivation. They are deterministic and extremely efficient, being driven by a table of parsing actions that is pre-compiled from the grammar.

The common core of all of the LR parser variants is a basic Shift-Reduce parser, which is fundamentally no more than a finite-state pushdown automaton (PDA). The parser scans a given input left to right, word by word. At each step, the LR parser may either *shift* the next input word onto the stack, *reduce* the current stack according to a grammar rule, *accept* the input, or *reject* it. An action table that is pre-compiled from the grammar guides the LR parser when parsing an input. The action table specifies the next action that the parser must take, as a function of its current state and the next word of the input.

3.1.2. The GLR Parsing Algorithm

Tomita's Generalized LR parsing algorithm extended the original LR parsing algorithm to the case of non-LR languages, where the parsing tables contain entries with multiple parsing actions. The algorithm deterministically simulates the non-determinism introduced by the conflicting actions in the parsing table by efficiently pursuing in a pseudo-parallel fashion all possible actions. The primary tool for performing this simulation efficiently is the *Graph Structured Stack* (GSS). Two additional techniques, *local ambiguity packing* and *shared parse forests*, are used in order to efficiently represent the various parse trees of ambiguous sentences when they are parsed.

3.1.3. The Unification-based GLR Parsing System

The Generalized LR Parser/Compiler is a unification based practical natural language system that was designed around the GLR parsing algorithm at the Center for Machine Translation at Carnegie Mellon University. The system supports grammatical specification in an LFG framework, that consists of context-free grammar rules augmented with feature bundles that are associated with the non-terminals of the rules. Feature structure computation is, for the most part, specified and implemented via unification operations. This allows the grammar to constrain the applicability of context-free rules. A reduction by a context-free rule succeeds only if the associated feature structure unification is successful as well. The Generalized LR Parser/Compiler is implemented in Common Lisp, and has been used as the analysis component of several different projects at the Center for Machine Translation at CMU in the course of the last several years.

3.2. The GLR* Parsing Algorithm

Chapter 3 of the thesis describes the GLR* parsing algorithm, analyzes its computational complexity and evaluates its practical performance. It also describes the search control heuristics that were developed to ensure that the algorithm performs within feasible time and space bounds, and evaluates the effectiveness of these heuristics.

The GLR* parsing algorithm was designed to be robust to two particular types of extra-grammaticality: noise in the input, and limited grammar coverage. It attempts to overcome these forms of extra-grammaticality by ignoring the unparsable words and fragments and conducting a search for the maximal subset of the original input that is covered by the grammar.

3.2.1. The Unrestricted GLR* Parsing Algorithm

GLR* accommodates skipping words of the input string by allowing shift operations to be performed from inactive state nodes in the GSS. Shifting an input symbol from an inactive state is equivalent to skipping the words of the input that were encountered after the parser reached the inactive state and prior to the current word that is being shifted. Since the parser is LR(0), previous reduce operations remain valid even when words further along in the input are skipped, since the reductions do not depend on any lookahead.

Due to the word skipping behavior of the GLR* parser, local ambiguities occur on a much more frequent basis than before. In many cases, a portion of the input sentence may be reduced to a non-terminal symbol in many different ways, when considering different subsets of the input that may be skipped. Since the ultimate goal of the GLR* parser is to produce maximal (or close to maximal) parses, the process of local ambiguity packing can be used to discard partial parses that are not likely to lead to the desired maximal parse. Local ambiguities that differ in their coverage of an input segment can be compared, and when the word coverage of one strictly subsumes that of another, the subsumed ambiguity can be discarded.

3.2.2. Complexity Analysis and Performance Evaluation

The complexity analysis of the unrestricted GLR* algorithm proves that the asymptotic time complexity of the algorithm is $O(n^{p+1})$, where n is the length of the input, and p is the length of the longest grammar rule. This complexity bound is similar to that of the original GLR parsing algorithm. This result may seem surprising, but we provide several explanations for it. With respect to the grammar size, GLR has a non-polynomially-bound runtime, and once again, GLR* has a similar bound. These complexity results thus do not provide much insight into the actual differences in runtime behavior between GLR and GLR* on common practical grammars. We therefore conducted experiments to evaluate the practical performance of the unrestricted GLR* algorithm.

Time and space performance of the unrestricted GLR* parser was evaluated on a benchmark setting from the JANUS speech-to-speech translation project. We used a version of the English analysis grammar for the scheduling domain. The benchmark consists of 552 English sentences, with sentence length ranging from 2 to 15 words, and an average length of about 6 words per sentence. The evaluation results show that the time and space behavior of the unrestricted GLR* parser diverges very rapidly from that of GLR, and becomes infeasible for even sentences of medium length. Whereas GLR time and space requirements increase almost linearly as a function

of the sentence length, the performance of unrestricted GLR* appears to be similar to that predicted by the worst case complexity analysis. This suggests that effective search heuristics are required to ensure that the algorithm performs within feasible time and space bounds.

3.2.3. The Search Control Heuristics

Since the purpose of GLR* is to find only maximal (or close to maximal) parsable subsets of the input, we can drastically reduce the amount of search by limiting the amount of word skipping that is considered by the parser. We developed and experimented with two heuristic search techniques. With *the k-word Skip Limit Heuristic*, the parser is restricted to skip no more than k consecutive input words at any point in the parsing process. With the *Beam Search Heuristic*, the parser is restricted to pursue a "beam" of a fixed size k of parsing actions, which are selected according to a criterion that locally minimizes the number of words skipped. There exists a direct tradeoff between the amount of search (and word skipping) that the parser is allowed to pursue and the time and space performance of the parser itself. The goal is to determine the smallest possible setting of the control parameters that allows the parser to find the desired parses in an overwhelming majority of cases.

We conducted empirical evaluations of parser behavior with different settings of the search control parameters, using the same benchmark setting on which the unrestricted GLR* parser was evaluated. The results show that time and space requirements of GLR* increase as the control parameters are set to higher values. This increase however is gradual, and in the case of the beam parameter, even with considerably high settings, parser performance remains in the feasible range, and does not approach the time and space requirements experienced when running the unrestricted version of GLR*. A comparison of the performance figures of the two heuristic control mechanisms revealed a clear advantage of using the beam search, versus the simpler skip word limit heuristic.

3.3. Statistical Disambiguation

The abundance of parse ambiguities for a given input is greatly exacerbated in the case of the GLR* parser, due to the fact that the parser produces analyses that correspond to many different maximal (or close to maximal) parsable subsets of the original input. It is therefore imperative that the the parser be augmented with a statistical disambiguation module, which can assist the parser in selecting among the ambiguities of a particular parsable input subset. Chapter 4 describes a newly developed statistical disambiguation module, designed to augment the unification-based versions of both the GLR and GLR* parsing systems.

3.3.1. The Statistical Framework

Our probabilistic model is based on a similar model developed by Carroll, where probabilities are associated *directly* with the actions of the parser, as they are defined in the pre-compiled parsing table. Because the state of the LR parser partially reflects the left and right context of the parse being constructed, modeling the probabilities at this level has the potential of capturing contextual preferences that cannot be captured by probabilistic context-free grammars. Because we use a finite-state probabilistic model, the action probabilities do not depend on the given input sentence. The relative probabilities of the input words at various states is implicitly represented by the action probabilities.

3.3.2. Training the Probabilities

The correctness of a parse result in our parsing system is determined according to the feature structure associated with the parse and not the parse tree itself. We are therefore interested in parse disambiguation at the feature structure level. To allow adequate training of the statistical model, we collect a corpus of sentences and their correct feature structures. Within the JANUS project, a large collection of sentences and their correct feature structures had been already constructed for development and evaluation purposes, and was thus directly available for training the probabilities of the statistical model. For other domains (including ATIS), we developed an interactive disambiguation procedure for efficiently constructing a corpus of disambiguated feature structures.

Once we have a corpus of input sentences and their disambiguated feature structures, we can train the finite-state probabilistic model that corresponds to the LR parsing table. In order for this to be done, a method was developed by which a correct feature structure of a parsed sentence can be converted back into the sequence of transitions the parser took in the process of creating the correct parse. Using this method, we process the training corpus and accumulate counters that keep track of how many times each particular transition of the LR finite-state machine was taken in the course of correctly parsing all the sentences in the training corpus. After processing the entire training corpus and obtaining the counter totals, the counter values are treated as action frequencies and converted into probabilities by a process of normalization. A smoothing technique is applied to compensate for transitions that did not occur in the training data.

3.3.3. Runtime Parse Disambiguation

Statistical disambiguation for the unification-based GLR/GLR* parsing system is performed as a post-process, after the entire parse forest has been constructed by the parser. At runtime, the statistical model is used to score the alternative parse analyses that were produced by the parser. Once computed, these scores can be used for disambiguation, by unpacking the analysis that received the best probabilistic score from the parse forest.

Due to ambiguity packing, the set of alternative analyses and their corresponding parse trees are packed in the parse forest. For packed nodes, a statistical score is computed for each of the alternative sub-analyses that are packed into the node. The packed node itself is then assigned the "best" of these scores, by taking a maximum. This score will then be used to compute the scores of any parent nodes in the forest. According to this scheme, the score that is attached to the root node of the forest should reflect the score of the best analysis that is packed in it. Because the score computation of an internal node in the parse forest requires the prior computation of the scores of the node's children, parse node scoring is performed in a "bottom-up" fashion. Because of some effects of packing and unification, the score assigned to a parse node it is not guaranteed to be the actual true probability of the best sub-analysis rooted at the node. However, the score is always a close over-estimate of this probability. It may thus be viewed as a heuristic for determining the actual best scoring ambiguity packed in the forest.

Once the preliminary task of scoring all the nodes in the parse forest is complete, the actual parse disambiguation can be performed by a process of unpacking the best scoring analysis from the forest. While the node scoring was done in a "bottom-up" fashion, unpacking is done "top-down", starting from the node at the root of the forest. Unpacking is done by going down the parse tree, and selecting one of the alternative packed ambiguities at each packed parse node.

3.3.4. Evaluation

We evaluated the statistical disambiguation module for two large practical grammars. The first is a syntactic grammar developed for the ATIS domain. The second grammar is the Spanish analysis grammar developed for the JANUS/Enthusiast project. It is a predominantly semantic grammar (with some syntactic structural rules) for the appointment scheduling domain.

To train the probabilities for the ATIS grammar, we used the interactive training procedure, where for each sentence, the user is required to select the correct analysis (feature structure) from the set of analyses produced by the parser. Because this task is time consuming, training was conducted on a set of only 500 sentences, from which a smoothed probability table was constructed. We then tested the disambiguation module on an unseen set of 119 sentences. 67 out of the 119 test sentences (56.3%) are ambiguous. In 52 out of these 67 ambiguous sentences (77.6%), one of the analyses produced by the parser is the correct analysis. The disambiguation module selected the correct analysis in 39 out of the 52 sentences, resulting in a success rate of 75.0%. Without statistical disambiguation, the parser is capable of selecting the correct parse in 30 of the 52 sentences (57.7%), by simply choosing the first analysis out of the set of analyses returned by the parse. This is a good result, when we consider the fact that the size of the training set was inadequately small (the training observed a mere 4.4% of the possible actions, and no actions at all were observed for 60% of the states).

For the Spanish analysis grammar, the probabilities for statistical disambiguation were trained on a corpus of "target" (correct) ILTs of 1687 sentences. We then evaluated the performance of the statistical disambiguation on the same set of sentences. Out of the 1687 sentences in the corpus, 596 (35.3%) are ambiguous. For 516 of these sentences (86.6%), one of the analyses returned by the parser is completely correct. The statistical disambiguation module successfully chose the correct parse in 375 out of the 516 sentences, thus resulting in a success rate of 72.3%. Without statistical disambiguation, the parser manages to select the correct parse for 50% of the 516 sentences, by simply choosing the first analysis out of the set returned by the parser. Although the size of the corpus that was available for training for the Spanish grammar was about three times larger than that used for the ATIS grammar, we still observed and modeled only a small fraction of the actual transitions allowed by the grammar. The training observed only 5.3% of all possible actions, and for 59% of the states, no actions were observed.

The statistical disambiguation results for both grammars demonstrate that even extremely small amounts of correct training data can establish structural preferences for resolving the most frequently occurring types of ambiguity. Thus, significant benefits can be gained from using the statistical disambiguation module even when trained on relatively small amounts of data.

3.4. Parse Evaluation Heuristics

Chapter 5 describes an integrated framework that we have developed for the GLR* parser, that allows different heuristic parse evaluation measures to be combined into a single evaluation function, by which sets of parse results can be scored, compared and ranked. The framework itself is designed to be general, and independent of any particular grammar or domain to which it is being applied. The framework allows different evaluation measures to be used for different tasks and domains. Additional evaluation measures can be developed and added to the system. The

evaluation framework includes tools for optimizing the performance of the integrated evaluation function on a given training corpus.

The utility of a parser such as GLR* obviously depends on the semantic coherency of the parse results that it returns. Since the parser is designed to succeed in parsing almost any input, parsing success by itself can no longer provide a likely guarantee of such coherency. We thus developed a classification heuristic by which the parser tries to self-identify situations in which even the "best" parse result is inadequate.

3.4.1. The Integrated Framework for Parse Evaluation

The integrated parse evaluation framework is a method for combining different evaluation measures into a single evaluation function. Each of the available evaluation measures is called a score function. A score function can reflect any property of either a parse candidate or the skipped (or substituted) portions of the input that correspond to a parse candidate. We use linear combination as the scheme for combining the individual score functions. With linear combination, each of the individual scores are first scaled by a weight factor, and the results are then summed.

Selecting the weight assigned to each of the score functions allows us to intentionally enforce the dominance of certain evaluation measures over others. To fine tune these weights, we developed a semi-automatic training procedure, which attempts to find the optimal combination weights for a given training corpus. The goal of the optimization procedure is to adjust the weights of the score functions in a way that maximizes the number of sentences in the training corpus for which the correct parse is assigned the best score. This is done by analyzing the set of sentences for which the correct parse was not assigned the best score. These sentences are classified into error vectors, which reflect which weights should be increased and which should be decreased. The optimization procedure works iteratively. It is only semi-automatic, due to the fact that it consults with the user before actually modifying the weights of the score functions, and proceeding to the next iteration.

3.4.2. The Parse Evaluation Score Functions

For both the JANUS scheduling domain and the ATIS grammar, we developed a set of four evaluation measures: a penalty function for skipped words, a penalty function for substituted words, a penalty function for the fragmentation of the parse analysis, and a penalty function based on the statistical score of the parse.

In its basic form, the penalty for skipping words assigns a penalty in the range of $[0.95, 1.05]$ for each skipped word. This scheme assigns a slightly higher penalty to skipped words that appear later in the input. This was designed to help us to deal with false starts and repeated words. This simple scheme was further developed into one in which the penalty for skipping a word is a function of the saliency of the word in the domain. To determine the saliency of our vocabulary words, we compared their frequency of occurrence in transcribed *domain* text with the corresponding frequency in "general" text.

For the ATIS domain, we attempted to deal with substitution errors of the speech recognizer using a confusion table. Words that are common misrecognitions of other words are listed in the table along with one or more words that are likely to be their corresponding correct word. When working with a confusion table, whenever the GLR* parser encounters a possible substitution that appears in the table, it attempts to continue with both the original input word and the possible

"correct" word(s). A word substitution should incur some penalty, since we would like to choose an analysis that contains a substitution only in cases where the substitution resulted in a better parse. We currently use a simple penalty scheme, where each substitution is given a penalty of 1.0.

In terms of their grammatical restrictness, our grammars for parsing spontaneous speech are rather "loose". The major negative consequence of this looseness is a significant increase in the degree of ambiguity of the grammar. In particular, utterances that can be analyzed as a single grammatical sentence, can often also be analyzed in various ways as collections of clauses and fragments. Our experiments have indicated that, in most such cases, a less fragmented analysis is more desirable. We thus developed a method for associating a numerical fragmentation value with each parse. This fragmentation value is then used as a penalty score.

Our statistical disambiguation uses a finite-state model that assigns an actual probability to every possible transition sequence. Thus, transition sequences that correspond to parses of different inputs are directly comparable. This enables us to use the statistical score as a parse evaluation measure. To enable parse probabilities to be used as an evaluation measure, we convert the probability into a penalty score, such that more "common" parse trees receive a lower penalty.

3.4.3. The Parse Quality Heuristic

We developed a classification heuristic that is designed to try and identify situations in which even the "best" parse result is inadequate. Our parse quality heuristic is designed to classify the parse chosen as best by the parser into one of two categories: "Good" or "Bad". The classification is done by a judgement on the combined penalty score of the parse that was chosen as best. The parse is classified as "Good" if the value of penalty score does not accede a threshold value. The value of the threshold is relative to the length of the input utterance, since longer input utterances can be expected to accommodate more word skipping, while still producing a good analysis.

3.4.4. Performance Evaluation

We evaluated the performance of the parse selection heuristics and the parse quality heuristic in two different settings. The first setting was an unseen test set of 513 transcribed Spanish dialogs in the scheduling domain, using the Spanish analysis grammar developed for the JANUS/Enthusiast project. The second setting is a unseen test set of 120 sentences in the ATIS domain, using the ATIS syntactic grammar.

In the Spanish grammar evaluation, 500 out of the 513 sentences are parsable, and for 300 out of the 500 parsable sentences, one of the parses produced by the parser completely matches its target ILT. In 33 cases, the correct parse requires some word skipping. With the full set of parse evaluation heuristics, 26 of the 33 cases (78.8%) were correct. A simple evaluation heuristic that selects the maximal parsable subset (breaking ties randomly) finds the correct parse in 24 of the 33 cases (72.7%). 200 out of the 500 parsable test sentences do not produce a completely correct ILT. 148 out of the 200 are sentences that require some amount of skipping in order to be parsed. With the simple parse selection heuristic, the parser selects a parse that results in an acceptable translation in 72 out of the 148 sentences. With the full set of heuristics, an acceptable translation is produced in 78 out of the 148 cases. Additionally, we found that in only 6 additional cases there exists an analysis that would produce an acceptable translation that was not selected by either of the methods.

We also evaluated the parse quality heuristic for the Spanish grammar. 57 out of the 500 parsable sentences are marked as "Bad" by the heuristic. None of the 57 sentences marked "Bad" have a completely correct parse that matches the corresponding target ILT. For 47 out of the 57 sentences marked "Bad" (82.4%), the selected parse produces a bad translation. Of the 443 sentences marked "Good" by the parse quality heuristic, the selected parse for 274 of the sentences (61.9%) matches its target ILT, and for 115 of the sentences (26.0%), the selected parse is not a complete match, but produces an acceptable English translation. Thus, the translation accuracy success rate for parses marked as "Good" by the parse quality heuristic is 87.9%.

The results for the ATIS evaluation were rather similar. 119 of the 120 test sentences were parsable. The simple parse selection heuristic selects an acceptable parse for 69 out of the 119 parsable sentences (58.0%). The full parse selection heuristics select an acceptable parse in 4 additional cases, for a total of 73 out of 119 (61.3%). 23 out of the 119 parsed sentences are marked as "Bad". For 21 of these 23 sentences (91.3%), the selected parse is indeed bad. Also, in only 14 cases (14.6%) did the parse quality heuristic fail to mark a truely bad parse as "Bad".

3.5. Parsing Spontaneous Speech using GLR*

Chapter 6 describes how the GLR* parser has been integrated into JANUS, a practical speech-to-speech translation system, designed to facilitate communication between a pair of different language speaking speakers, attempting to schedule a meeting. We also evaluate the performance of GLR* on speech recognized input in the ATIS domain.

3.5.1. Parsing Full Utterances with GLR*

In JANUS, the output of the speech recognizer is a textual hypothesis of the complete utterance, and contains no explicit representation of the boundaries between the clauses and sentences of which the utterance is composed. The analysis grammars developed for the scheduling domain were constructed to primarily analyze clauses. Multi-clause utterances must therefore be broken into clauses, in order to be properly parsed. We handle multi-clause utterances using a combination of methods. First, the analysis grammars were extended with rules that allows the input utterance to be analyzed as a concatenation of several clauses. However, for typical multi-clause utterances, there are many different possible ways of breaking the utterance into clauses that are analyzable by the grammar. For long multi-clause utterances, the computational complexity of pursuing all such possibilities results in infeasible parse times and memory requirements. We significantly reduce the severity of this problem by using a pre-processor to break the utterance into smaller sub-utterances, that may still contain a number of clauses. In addition, we developed a powerful set of heuristics, including a statistical clause boundary predictor, for further constraining the breaking possibilities that are considered by the parser.

3.5.2. JANUS Performance Evaluation

Evaluations were conducted on both the English and Spanish analysis grammars, and processing both transcribed and speech recognized input. We compared the performance of the system incorporating GLR* with a version that uses the original GLR parser, and with a version that uses the Phoenix parser. The results for both the Spanish and English are rather similar, and we thus summarize here only the English results.

The test set consisted of 99 push-to-talk unseen utterances. The transcribed text was pre-broken into clauses according to transcribed markings. This produced a set of 360 sub-utterances or clauses, each of which was parsed separately. The top-best speech recognized hypotheses were pre-broken using the two consecutive noise words criterion. This produced a set of 192 sub-utterances.

On transcribed input, GLR succeeds to parse 55.8% of the clauses, while GLR* succeeds in parsing 95.0% of the clauses. This amounts to a gain of about 39% in the number of parsable clauses. On the speech recognized input, GLR parses only 21.4% of the input sub-utterances, while GLR* succeeds to parse 93.2%. In this case, the increase in parsable sub-utterances amounts to almost 72%. On transcribed input, GLR* produced acceptable translations for 85.6% of the clauses, compared with 54.2% for GLR. Thus, GLR* produces over 30% more acceptable translations. This is also the case when only the parses marked "Good" by the parse quality heuristic of GLR* are considered. The results on speech recognized input are rather similar. While GLR produced an acceptable translation only 17.2% of the time, GLR* did so in 47.9% of the time. Once again, this amounted to an increase of about 30.0% in the number of acceptable translations. Similar results were achieved when only the parses marked "Good" by GLR* are considered.

We also evaluated the effectiveness of the parse quality heuristic. On transcribed data, 92.7% of the parses marked "Good" by the parse quality heuristic produce acceptable translations. 10 out of 12 clauses marked by GLR* as "bad" (83.3%) result in a bad translation. On speech data, the effectiveness of the parse quality heuristic is hindered by errors of the speech recognizer. Consequently, parses marked as "Good" by the parse quality heuristic, produced acceptable translations for only 66.9% of the sub-utterances. However, when errors that are completely due to poor recognition are considered acceptable, 96.2% of the clauses marked "Good" by the parse quality heuristic are in fact acceptable. Parses marked "Bad" by the parse quality heuristic produced a bad translation in 93.5% of the time.

The performance of GLR* and Phoenix on this test set was quite similar. On both transcribed and speech input, the percentage of utterances with acceptable translations using Phoenix was slightly better than the corresponding figure when using GLR*. On transcribed input, GLR* produces acceptable translations for 83.9% of the utterances, versus 85.5% for Phoenix. On speech input, GLR* has acceptable translations for 44.5% of the utterances, while Phoenix has acceptable translations for 45.8%.

3.5.3. GLR* Performance on ATIS Domain

For the ATIS evaluation, we used a set of 200 unseen speech recognized sentences. The portion of parsable sentences increased from 51.7% for GLR to 99.2% for GLR*. However, the portion of parsable sentences with an acceptable parse increased only by 10.8% (from 50.0% for GLR, to 60.8% for GLR*). The evaluation of the parse quality heuristic shows that while the portion of bad parses out of the parses marked "Bad" by the parser is still very high (91.3%), the parse quality heuristic is somewhat less effective on parses marked "Good". Only 74% of the parses marked "Good" were in fact acceptable. Further inspection, however, indicated that 11 marked "Good" sentences had bad parses due to incorrect ambiguity resolution. If the correct ambiguity were chosen for these parses, the portion of good parses out of parses marked "Good" would reach 85%, similar to the comparable figures found in the JANUS evaluations.

3.6. Parsing Speech Lattices using GLR*

In practice, the top-best hypotheses produced by our speech recognition system are far from perfect. Speech recognition errors hinder on the ability of the parser to find a correct analysis for the utterance, and the effects of this toll are reflected in the disparity between our performance results on transcribed and speech recognized input. Processing multiple speech hypotheses instead of a single hypothesis thus has the potential of detecting a hypothesis with fewer recognition errors, which should lead to an improvement in the overall translation performance. In Chapter 7, we investigate how GLR* can be adapted into a parser capable of directly parsing speech produced word lattices.

3.6.1. The Advantages of Parsing Speech Lattices

Parsing the speech lattice directly attempts to efficiently accomplish the same results of parsing an n-best list. Instead of first extracting the list of hypotheses from the speech lattice, and then parsing each of them separately, we use a version of the parser that can parse the lattice directly. Each word in the lattice is parsed only once, although it may contribute to many different hypotheses. The lattice parser produces a large set of possible parses, corresponding to the parses of various complete word paths through the lattice. This set of parses can be scored and ranked according to the parse score, or according to some optimized combination of the parse score and recognizer score.

3.6.2. The GLR* Lattice Parsing Algorithm

For the lattice parser to function correctly, it is essential that the words of the lattice be parsed in a proper order. If a lattice word $[w_1(i_1, j_1)]$ appears prior to $[w_2(i_2, j_2)]$ in some path through the lattice, then they must be parsed in that order. A simple procedure can verify that the lattice words are properly ordered prior to parsing.

In the lattice parsing version of GLR*, the shift distribution step was modified to handle lattice input words. Similar to the string parsing version of GLR*, shift actions must first be distributed to state nodes that do not result in any new skipped words. Subsequently, if the number of shift actions distributed has not already acceded the beam-limit, other inactive state nodes are considered as well. State nodes that result in fewer skipped words are considered first. This is done by first detecting all levels that correspond to words of distance one from the current word. Shift actions are then distributed to state nodes of these levels. If the total number of shift actions distributed is still below the beam-limit, we proceed to find the levels that correspond to words of distance two, and distribute shift actions to the state nodes of these levels as well. This process continues until the total number of distributed shift actions reaches the beam-limit.

When dealing with a general word lattice, determining word connectivity and lattice paths is more complicated. For two arbitrary words $[w_1\ (i_1, j_1)]$ and $[w_2\ (i_2, j_2)]$ in the lattice, there could be zero, one, or multiple paths of lattice words that connect them. We thus developed a procedure for determining the connectivity of two points in the lattice.

An additional field was added to the symbol node structure, in order to keep track of the set of lattice words that are "covered" by the symbol. This set contains all the words that were actually parsed in the sub-parse tree rooted by the symbol. The ambiguity packing procedure of the parser

was also slightly modified. Two symbol nodes of the same grammar category can be packed by the lattice parser only if they both correspond to a parse of the exact same sub-path in the lattice.

3.6.3. The Lattice Processing Version of JANUS

The lattice parsing version of GLR* has been incorporated into a lattice processing version of the JANUS system. This required some changes in the configuration of the system. For each input utterance, instead of producing a top-best hypothesis or an n-best list, the speech recognition module outputs a complete scored word lattice. The lattices produced by the speech recognizer are usually far too large and redundant to be parsed in feasible time and space, and require significant pruning in order to be parsed. We use a lattice pre-processor for this task. The pre-processor performs four major functions. First, it collapses noise-words, and removes redundant paths that differ only in noise-words. Next, it breaks the lattice into sub-lattices according to pauses in the speech signal, which are highly correlated with clause boundaries. Third, the transformed sub-lattices are rescored with a language model. Finally, each sub-lattice is pruned to a parsable size using a method that ensures that the best scoring hypotheses are not pruned out. The resulting sub-lattices are then separately passed on to the lattice parser for parsing. A post-parsing procedure combines the results of parsing the sub-lattices into a ranked list of output results which is passed on to the discourse processor and to the generation module.

3.6.4. Performance Evaluation

We have recently conducted an evaluation to measure the performance of the lattice parsing approach. On a common test set of 14 unseen English dialogs (99 utterances), we compared the results of using the lattice processing version of the JANUS system with the results of processing the top-best speech hypothesis. The preliminary results show only a slight improvement in the total percentage of acceptable translations, which increased from 41.4% for parsing the top-best speech hypothesis, to 44.5% when parsing lattices. This compares with a performance of 83.9% acceptable translations on the transcribed input of the same data. The lattice parser succeeded to produce an acceptable translation for 15 utterances that were badly translated due to poor recognition when the top-best speech hypothesis was used. However, 12 utterances that were acceptably translated by using the top-best speech hypothesis, were translated poorly when using the lattice parser. In most of these cases, this was due to the fact that the lattice parser prefered a hypothesis in the lattice that produced a better parse score than the top-best speech hypothesis, even though it contained more speech recognition errors. Such problems could possibly be resolved by a better combination scheme between the acoustic score and the parser score, and by improvements in the parse score heuristics, and require further investigation.

4. Thesis Contributions

The major contributions of this thesis are the following:

1. The development of an efficient robust general parsing architecture, capable of significantly reducing the problems of noise contaminated input and limited grammar coverage.

2. The development of a general framework for evaluating and ranking sets of competing parse analyses, that is capable of combining a variety of knowledge sources and heuristic evaluation measures. Additionally, the development of a parse quality heuristic, that allows the parser to self-judge the quality of its output results.

3. The demonstration of the effectiveness and practical feasibility of the methods employed by GLR* in parsing spontaneous speech input, by substantially enhancing the end-to-end performance of the JANUS speech-to-speech translation system.