GLR* :
A ROBUST PARSER
FOR SPONTANEOUSLY SPOKEN LANGUAGE

Alon Lavie
Center for Machine Translation
Carnegie Mellon University
Pittsburgh, PA 15232
USA

Email: lavie@cs.cmu.edu

Abstract

This paper 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: a statistical disambiguation module, an integrated heuristic for evaluating and ranking the parses produced by the parser, and a parse quality heuristic, that allows the parser to self-judge the quality of the parse chosen as best.

To evaluate 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 about 30% more acceptable translations, than a corresponding version that uses the original non-robust GLR parser.

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 paper 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 GLR* 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 SLR(0) 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 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.

2 Foundation: GLR Parsing

The GLR* parsing algorithm is based on the Generalized LR (GLR) parsing algorithm that was developed by Tomita. 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 [1]. LR parsers parse the input bottom-up, scanning it from left to right, and producing a rightmost derivation. LR parsers are driven by a table of parsing actions that is pre-compiled from the grammar. For the limited class of LR grammars, these compiled parsing tables are deterministic, resulting in an extremely efficient linear-time parser. LR parsers come in several variants. The common core of all of them 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.

Tomita's Generalized LR parsing algorithm [4] 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). The GSS is a data-structure that efficiently represents the multiple parsing stacks that correspond to different sequences of parsing actions. 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. Local ambiguity packing collects multiple sub-analyses of the same category type, all of which derive the same substring, into a single structure. Further parsing actions may then refer to the single structure that represents the entire collection of sub-parses, rather than to each of the packed sub-parses separately. Shared packed forests allow common sub-parses of different analyses to be shared via pointers.

The Generalized LR Parser/Compiler [5] 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. GLR* was implemented as an extension to the unification-based Generalized LR Parser/Compiler.

3 The GLR* Parsing Algorithm

The GLR* parsing algorithm [3] 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 string that is covered by the grammar.

GLR* accommodates skipping words of the input string by allowing shift operations to be performed from *inactive* state nodes in the GSS [1]. 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.

Similar to the GLR algorithm, GLR* parses the input in a single left-to-right scan of the input, processing one word at a time. The processing of each input word is called *a stage*. Each stage consists of two main *phases*, a *reduce phase* and a *shift phase*. The reduce phase always precedes the shift phase. The outline of each stage of the algorithm is shown in Figure 1. `RACT(st)` denotes the set of reduce actions defined in the parsing table for state *st*. Similarly, `SACT(st,x)` denotes the shift actions defined for state *st* and symbol x, and `AACT(st,x)` denotes the accept action.

The last stage of the algorithm, in which the end-of-input symbol "$" is processed, is somewhat different. The `READ`, `DISTRIBUTE-REDUCE` and `REDUCE` steps are identical to those of previous stages. However, a `DISTRIBUTE-ACCEPT` step replaces the `DISTRIBUTE-SHIFT` step. In the `DISTRIBUTE-ACCEPT` step, accept actions are distributed to all state nodes *st* in the GSS (active and inactive), for which `AACT(st,$)` is true. Pointers to these state nodes are then collected into a list of final state nodes. If this list is not empty, GLR* accepts the input, otherwise, the input is rejected. Finally, the `GET-PARSES` step creates a list of all symbol nodes that are direct descendents of the final state nodes. These symbol nodes represent the set of complete parses found by the parser, and contain pointers to the roots of the parse forest.

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

[1]An *inactive* state node is one that is internal in the GSS, thus not at the top of any of the represented "stacks".

```
(1) READ:
    Read the next input token x.

(2) DISTRIBUTE-REDUCE:
    For each active state node st, get the set of reduce actions
    RACT(st) in the parsing table, and attach it to the active
    state node.

(3) REDUCE:
    Perform all reduce actions attached to active state nodes.
    Recursively perform reductions after distributing reduce
    actions to new active state nodes that result from previous
    reductions.

(4) DISTRIBUTE-SHIFT:
    For each state node st in the GSS (active and inactive),
    get SACT(st,x) from the parsing table.
    If the action is defined, attach it to the state node.

(5) SHIFT:
    Perform all shift actions attached to state nodes of the GSS.

(6) MERGE:
    Merge active state nodes of identical states into a single
    active state node.
```

Figure 1: Outline of a Stage of the Unrestricted GLR* Parsing Algorithm

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. It should be noted that due to the non-local effects of unification, such pruning carries the danger of discarding a non-maximal local ambiguity in favor of an analysis that later on fails to be incorporated into a complete parse due to unification failure. However, our experience has shown that this is an extremely rare situation that for all practical purposes can be ignored.

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. However, in practice the performance of the two parsers rapidly diverges. Whereas, on average, GLR time and space requirements increase not much more than linearly as a function of the sentence length, the performance of unrestricted GLR* appears to be closer 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.

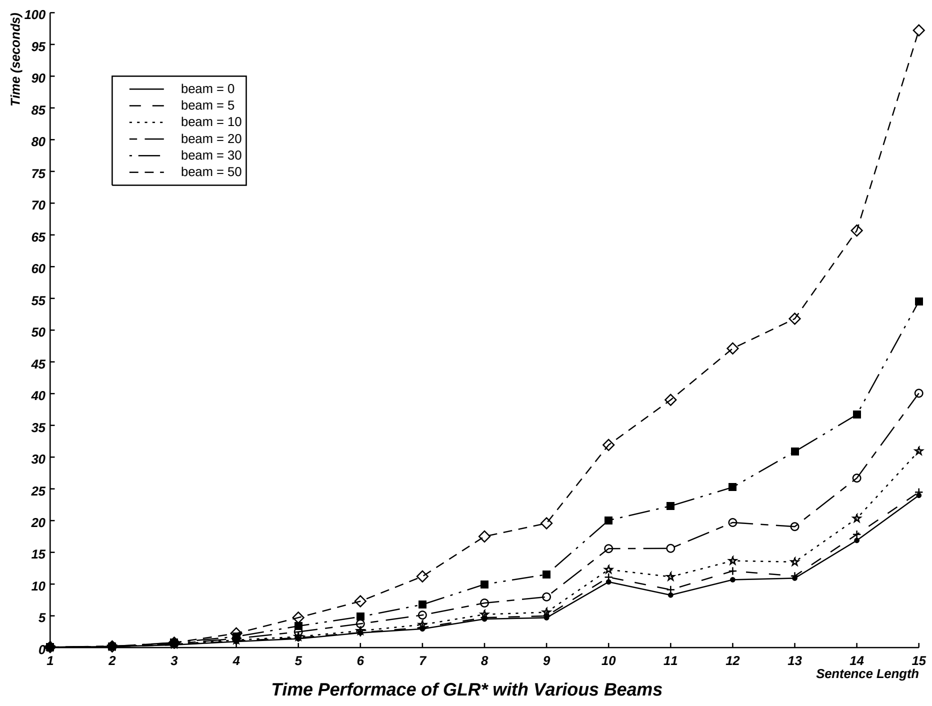

Figure 2: Time Performance of GLR* with Various Beam Widths as a Function of Sentence Length

3.1 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. For a given grammar and/or domain, 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. 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. Figure 2 shows the average parsing time as a function of input string length with various settings of the beam width parameter. The performance numbers were determined using an HP-735 Apollo workstation applied to a benchmark corpus of 552 utterances parsed with a compiled grammar for the English Scheduling domain.

4 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.

Our probabilistic model is based on a similar model developed by Carroll [2], 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. The probabilistic model assigns a probability to every possible transition from each possible state of the parser. These probabilities are not conditional on the sequence of input words, or on previous states of the parser. The relative probabilities of the input words at various states is implicitly represented via the action probabilities themselves.

4.1 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 available for training the statistical model. For other domains (such as ATIS), we developed an interactive disambiguation procedure for 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.

4.2 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 are 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 attached to 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 a root node of the forest should reflect the score of the best analysis that is packed in it. However, due to some effects of packing and unification, the score assigned to a parse node 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.

The actual task of parse disambiguation is performed at the end of the parsing process by unpacking the best scoring analysis from the parse forest. The unpacking is done "top-down", starting from the best scoring root node in the forest, and selecting one or more of the alternative packed ambiguities at each packed parse node. This results in a collection of unpacked parse trees. A unification consistency procedure verifies the validity of each of the unpacked parse trees, and the correct probability of each of the trees is computed. The best scoring tree and its corresponding analysis can then be selected.

4.3 Performance Evaluation of Statistical Disambiguation

We conducted an evaluation to assess the effectiveness of the statistical disambiguation model and compared it to the principle-based disambiguation method of Minimal Attachment. The evaluation was conducted using the December-94 version of the Spanish analysis grammar developed for the JANUS/Enthusiast project. The probabilities were trained on a corpus of "target" (correct) ILTs of 15 push-to-talk dialogs and 15 cross-talk dialogs, amounting to a total of 1687 sentences. We then evaluated the performance of the statistical disambiguation module on a unseen test set of 11 push-to-talk dialogs and 5 cross-talk dialogs, amounting to a total of 769 sentences. 524 out of the 769 sentences (68.1%) are parsable with one of the analyses returned by the parser being completely correct (thus matching the "target" ILT specified for the sentence). 213 of these 524 sentences (40.6%) are ambiguous. The statistical disambiguation module chose the correct parse in 140 out of the 213 sentences, thus resulting in a success rate of 65.7%. For comparison, we applied a Minimal Attachment disambiguation procedure to the same test set. With Minimal Attachment, the correct parse was chosen in only 79 out of the 213 sentences, resulting in a success rate of 37.1%. Thus, the statistical disambiguation method outperformed the Minimal Attachment method in this case by about 29%.

It should be noted that our training observed only a small fraction of the actual transitions allowed by the grammar. The parsing table for the Spanish grammar contained 2437 states and a total of 43023 actions. In the course of training the probabilities, only 2274 actions were observed, in 987 of the states. Thus, the training observed only 5.3% of all possible actions, and for 59% of the states, no actions were observed. With this in mind, statistical disambiguation performed remarkably well. This leads us to believe that even extremely small amounts of correct training data can establish structural preferences for resolving the most frequently occurring types of ambiguity.

5 Parse Evaluation Heuristics

To complement the GLR* parser, we developed an integrated framework 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 is designed to be general, and independent of any grammar or domain to which it is being applied. It allows different evaluation measures to be used for different tasks and domains. The evaluation framework includes tools for optimizing the performance of the integrated evaluation function on a given training corpus.

5.1 The Parse Evaluation Score Functions

We developed a set of four evaluation measures for the JANUS scheduling domain and for the ATIS domain: 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. In order to deal with false starts and repeated words, a slightly higher penalty is assigned to skipped words that appear later in the input. We further developed this penalty scheme 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 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. Empirically determined common misrecognitions are listed in the table. When working with a confusion table, whenever the GLR* parser encounters a word 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 wish to choose an analysis that contains a substitution only in cases where this results in a better parse. We currently use a simple penalty scheme, where each substitution is given a penalty of 1.0.

In order to cope with the looser notions of grammaticality that are typical of spontaneous speech, our grammars allow parses to consist of fragments and incomplete sentences. This significantly increases 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 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 model assigns a probability to every possible parser transition sequence. Thus, transition sequences that correspond to parses of different inputs are directly comparable. This enables us to use parse probabilities as an evaluation measure. The parse probability is converted into a penalty score, where statistically more likely parse trees receive lower penalties.

5.2 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 threshold is a function of the length of the input utterance, since longer input utterances can be expected to accommodate more word skipping, while still producing a good analysis. The threshold parameters were determined empirically.

6 Parsing Spontaneous Speech using GLR*

The GLR* parser has been integrated into JANUS [6], a practical speech-to-speech translation system, designed to facilitate communication between a pair of different language speaking speakers, attempting to schedule a meeting.

6.1 Parsing Full Utterances with GLR*

Speech translation in the JANUS system is guided by the general principle that spoken utterances can be analyzed and translated as a sequential collection of semantic dialogue units (SDUs), each of which roughly corresponds to a speech-act. SDUs are semantically coherent pieces of information. The interlingua representation in our system was designed to capture meaning at the level of such SDUs. Each semantic dialogue unit is analyzed into an interlingua representation. The output of the speech recognizer is a textual hypothesis of the complete utterance, and contains no explicit representation of the boundaries between its contained SDUs. Segmentation into SDUs is achieved in a two-stage process, partly prior to and partly during parsing.

Segmentation decisions in our system can be made much more reliably during parsing, at which point multiple sources of knowledge can be applied. Pre-parsing segmentation can, however, significantly reduce the amount of parse-time segmentation ambiguity, resulting in significantly faster parser performance. Thus, the goal of pre-parsing segmentation is to detect confident SDU boundaries in the utterance, and then pre-break the utterance into sub-utterances that may still contain multiple SDUs. Each of these sub-utterances is then parsed separately. Pre-parsing segmentation at SDU boundaries is determined using acoustic, statistical and lexical information. The remaining segmentation is performed during parsing. The analysis grammars contain rules that allow the input utterance to be analyzed as a concatenation of several SDUs. To further cut down the amount of segmentation ambiguity, we developed a powerful set of heuristics, including a statistical clause boundary predictor, that constrain the segmentation possibilities that are considered by the parser.

6.2 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. 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 average utterance word length is about 33. The transcribed text was pre-broken into SDUs according to explicit markings in the transcriptions. This produced a set of 360 SDUs (with an average length of about 9 words per SDU), each of which was parsed separately. The top-best speech recognized hypotheses were segmented using our pre-parsing segmentation procedure. This produced a set of 192 sub-utterances (some of which contain multiple SDUs).

On transcribed input, GLR succeeds to parse 55.8% of the SDUs, while GLR* succeeds in parsing 95.0% of the SDUs. This amounts to a gain of about 39% in the number of parsable SDUs. 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 SDUs, 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 SDUs 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 parses 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.

7 Conclusions

Our performance evaluations of the GLR* parser within the JANUS system have indicated that the parser is highly suitable for analyzing spontaneously spoken language. The version of the JANUS system that uses GLR* produced about 30% more acceptable translations, than a corresponding version that uses the original non-robust GLR parser. The parser appears to be equally well adjusted to handling both the disfluencies typical of spoken language and limitations of grammar coverage. Manual inspection of the transcribed test input utterances shows that about one-third of the SDUs contain some disfluencies, while the other two-thirds are grammatical. Due to grammar coverage limitations, the non-robust GLR parser fails to parse about 30% of the "grammatical" SDUs. By skipping over portions of the input, GLR* parses 97% of the SDUs. GLR* produced acceptable translations for 90.0% of the SDUs, compared with 67.5% for GLR. Thus, by overcoming limitations in the grammar coverage, GLR* produces over 22% more acceptable translations. This appears to indicate that using GLR* in conjunction with a grammar of reasonably adequate coverage can result in very high levels of performance, where minor unimportant portions of the input that are not covered by the grammar are detected by the parser and ignored. Thus, GLR* should prove to be very useful in overcoming grammar coverage problems in other tasks, where input is for the most part grammatical, but complete parsability is not crucial.

References

[1] A. V. Aho and S.C. Johnson. LR Parsing. *Computing Surveys*, 6(2):99–124, 1974.

[2] J. A. Carroll. *Practical Unification-Based Parsing of Natural Language*. PhD thesis, University of Cambridge, Cambridge, UK, October 1993. Computer Laboratory Technical Report 314.

[3] A. Lavie. *GLR*: A Robust Grammar-Focused parser for Spontaneously Spoken Language*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, May 1996. Technical Report CMU-CS-96-126.

[4] M. Tomita. An Efficient Augmented Context-free Parsing Algorithm. *Computational Linguistics*, 13(1-2):31–46, 1987.

[5] M. Tomita. The Generalized LR Parser/Compiler - Version 8.4. In *Proceedings of International Conference on Computational Linguistics (COLING'90)*, pages 59–63, Helsinki, Finland, 1990.

[6] M. Woszczyna, N. Aoki-Waibel, F. D. Buo, N. Coccaro, T. Horiguchi, K., T. Kemp, A. Lavie, A. McNair, T. Polzin, I. Rogina, C. P. Rosé, T. Schultz, B. Suhm, M. Tomita, and A. Waibel. JANUS-93: Towards Spontaneous Speech Translation. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'94)*, 1994.