Some Experiments on Indicators of Parsing Complexity for Lexicalized Grammars

Anoop Sarkar, Fei Xia and Aravind Joshi
Dept. of Computer and Information Science
University of Pennsylvania
200 South 33rd Street,
Philadelphia, PA 19104-6389, USA
{anoop,fxia,joshi}@linc.cis.upenn.edu

Abstract

In this paper, we identify syntactic lexical ambiguity and sentence complexity as factors that contribute to parsing complexity in fully lexicalized grammar formalisms such as Lexicalized Tree Adjoining Grammars. We also report on experiments that explore the effects of these factors on parsing complexity. We discuss how these constraints can be exploited in improving efficiency of parsers for such grammar formalisms.

1 Introduction

The time taken by a parser to produce derivations for input sentences is typically associated with the length of those sentences. The longer the sentence, the more time the parser is expected to take. However, complex algorithms like parsers are typically affected by several factors. A common experience is that parsing algorithms differ in the number of edges inserted into the chart while parsing. In this paper, we explore some of these constraints from the perspective of lexicalized grammars and explore how these constraints might be exploited to improve parser efficiency.

We concentrate on the problem of parsing using *fully* lexicalized grammars by looking at parsers for Lexicalized Tree Adjoining Grammar (LTAG). By a fully lexicalized grammar we mean a grammar in which there are one or more syntactic structures associated with each lexical item. In the case of LTAG each structure is a tree (or, in general, a directed acyclic graph). For each structure there is an explicit structural slot for each of the arguments of the lexical item. The various advantages of defining a lexicalized grammar formalism in this way are discussed in (Joshi and Schabes, 1991).

An example LTAG is shown in Figure 1. To parse the sentence *Ms. Haag plays Elianti* the parser has to combine the trees selected by each word in the sentence by using the operations of substitution and adjunction (the two composition operations in LTAG) producing a valid derivation for the sentence.

Notice that as a consequence of this kind of lexicalized grammatical description there might be several different factors that affect parsing complexity. Each word can select many different trees; for example, the word *plays* in Figure 1 might select several trees for each syntactic context in which it can occur. The verb *plays* can be used in a relative clause, a wh-extraction clause, among others. While grammatical notions of argument structure and syntax can be processed in abstract terms just as in other kinds of formalisms, the crucial difference in LTAG is that all of this information is compiled into a finite set of trees *before* parsing. Each of these separate lexicalized trees is now considered by the parser. This compilation is repeated for other argument structures, e.g. the verb *plays* could also select trees which are intransitive thus increasing the set of lexicalized trees it can select. The set of trees selected by different lexical items is what we term in this paper as *lexical syntactic ambiguity*.

The importance of this compilation into a set of lexicalized trees is that each predicate-argument structure across each syntactic context has its own lexicalized tree. Most grammar formalisms use feature structures to capture the same grammatical and predicate-argument information. In LTAG, this larger set of lexicalized trees directly corresponds to the fact that recursive feature structures are not needed for linguistic description. Feature structures are typically atomic with a few instances of re-entrant features.

Thus, in contrast with LTAG parsing, parsing for formalisms like HPSG or LFG concentrates on efficiently managing the unification of large feature structures and also the packing of ambiguities when these feature structures subsume each other (see (Oepen and Carroll, 2000) and references cited there). We argue in this paper that the result of having compiled out abstract grammatical descriptions into a set of lexicalized trees allows us to predict the number of edges that will be proposed by the parser even before parsing begins. This allows us to explore novel methods of dealing with parsing complexity that are difficult to consider in formalisms that are not fully lexicalized.

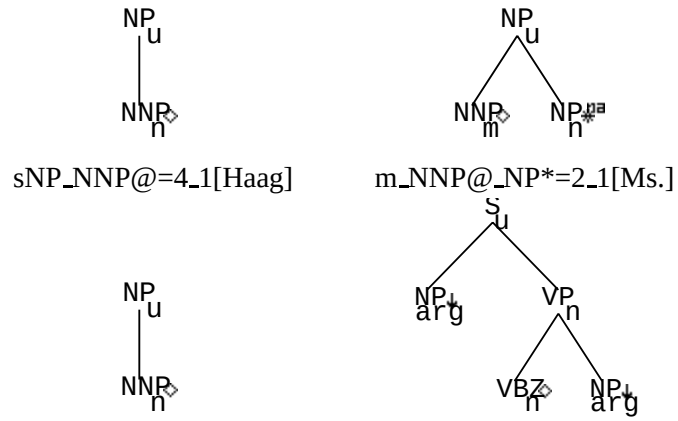

Figure 1: Example lexicalized elementary trees. They are shown in the usual notation: $\diamond$ = *anchor*, $\downarrow$= *substitution node*, $*$ = *footnode*, **na** = *null-adjunction constraint*. These trees can be combined using substitution and adjunction to parse the sentence *Ms. Haag plays Elianti*.

Furthermore, as the sentence length increases, the number of lexicalized trees increase proportionally increasing the attachment ambiguity. Each sentence is composed of several clauses. In a lexicalized grammar, each clause can be seen as headed by a single predicate tree with its arguments and associated adjuncts. We shall see that empirically the number of clauses grow with increasing sentence length only up to a certain point. For sentences greater than a certain length the number of clauses do not keep increasing.

Based on these intuitions we identify the following factors that affect parsing complexity for lexicalized grammars:

Syntactic Lexical Ambiguity The number of trees selected by the words in the sentence being parsed. We show that this is a better indicator of parsing time than sentence length. This is also a predictor of the number of edges that will be proposed by a parser, allowing us to better handle difficult cases *before* parsing.

Sentence Complexity The clausal complexity in the sentences to be parsed. We observe that the number of clauses in a sentence stops growing in proportion to the sentence length after a point. We show that before this point parsing complexity is related to attachment of adjuncts rather than attachment of arguments.

2 LTAG **Treebank Grammar**

The grammar we used for our experiments was a LTAG Treebank Grammar which was automatically extracted from Sections 02–21 of the Wall Street Journal Penn Treebank II corpus (Marcus et al., 1993). The extraction tool (Xia, 1999) converted the *derived* trees of the Treebank into *derivation* trees in LTAG which represent the attachments of lexicalized elementary trees. There are 6789 tree templates in the grammar with $47,752$ tree nodes. Each word in the corpus selects some set of tree templates. The total number of lexicalized trees is $123,039$. The total number of word types in the lexicon is $44,215$. The average number of trees per word type is 2.78. However, this average is misleading since it does not consider the frequency with which words that select a large number of trees occur in the corpus. In Figure 2 we see that many frequently seen words can select a large number of trees.

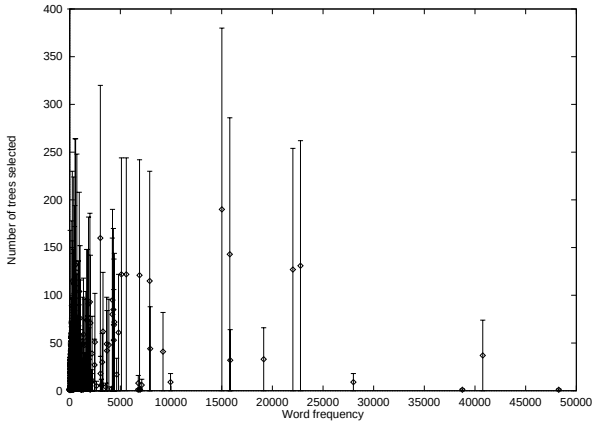

Figure 2: Number of trees selected plotted against words with a particular frequency. (x-axis: words of frequency x; y-axis: number of trees selected, error bars indicate least and most ambiguous word of a particular frequency x)

Another objection that can be raised against a

Treebank grammar which has been automatically extracted is that any parsing results using such a grammar might not be indicative of parsing using a hand-crafted linguistically sophisticated grammar. To address this point (Xia and Palmer, 2000), compares this Treebank grammar with the XTAG grammar (XTAG-Group, 1998), a large-scale hand-crafted LTAG grammar for English. The experiment shows that 82.1% of template tokens in the Treebank grammar matches with a corresponding template in the XTAG grammar; 14.0% are covered by the XTAG grammar but the templates in two grammars look different because the Treebank and the XTAG grammar have adopted different analyses for the corresponding constructions; 1.1% of template tokens in the Treebank grammar are not linguistically sound due to annotation errors in the original Treebank; and the remaining 2.8% are not currently covered by the XTAG grammar. Thus, a total of 96.1% of the structures in the Treebank grammar match up with structures in the XTAG grammar.

3 Syntactic Lexical Ambiguity

In a fully lexicalized grammar such as LTAG the combinations of trees (by substitution and adjunction) can be thought of as *attachments*. It is this perspective that allows us to define the parsing problem in two steps (Joshi and Schabes, 1991):

1. Assigning a set of lexicalized structures to each word in the input sentence.
2. Finding the correct attachments between these structures to get all parses for the sentence.

In this section we will try to find which of these factors determines parsing complexity when finding all parses in an LTAG parser.

To test the performance of LTAG parsing on a realistic corpus using a large grammar (described above) we parsed 2250 sentences from the Wall Street Journal using the lexicalized grammar described in Section 2.[1] All of these sentences were of length 21 words or less. These sentences were taken from the same sections (02-21) of the Treebank from which the original grammar was extracted. This was done to avoid the complication of using default rules for unknown words.

In all of the experiments reported here, the parser produces all parses for each sentence. It produces a shared derivation forest for each sentence which stores, in compact form, all derivations for each sentence.

We found that the observed complexity of parsing for LTAG is dominated by factors other than sentence length.[2] Figure 3 shows the time taken in seconds by the parser plotted against sentence length. We see a great deal of variation in timing for the same sentence length, especially for longer sentences.

We wanted to find the relevant variable other than sentence length which would be the right predictor of parsing time complexity. There can be a large variation in syntactic lexical ambiguity which might be a relevant factor in parsing time complexity. To draw this out, in Figure 4 we plotted the number of trees selected by a sentence against the time taken to parse that sentence. By examining this graph we can visually infer that the number of trees selected is a better predictor of increase in parsing complexity than sentence length. We can also compare numerically the two hypotheses by computing the coefficient of determination (R^2) for the two graphs. We get a R^2 value of 0.65 for Figure 3 and a value of 0.82 for Figure 4. Thus, we infer that it is the syntactic lexical ambiguity of the words in the sentence which is the major contributor to parsing time complexity.

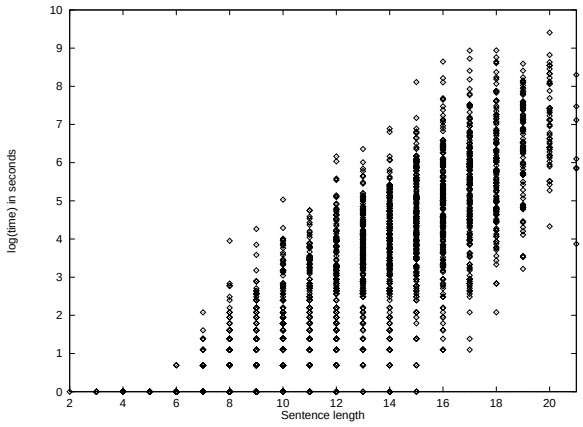

Figure 3: Parse times plotted against sentence length. Coefficient of determination: $R^2 = 0.65$. (x-axis: Sentence length; y-axis: log(time in seconds))

Since we can easily determine the number of trees selected by a sentence before we start parsing, we can use this number to predict the number of edges that will be proposed by a parser when parsing this sentence, allowing us to better handle difficult cases *before* parsing.

[1] Some of these results appear in (Sarkar, 2000). In this section we present some additional data on the previous results and also the results of some new experiments that do not appear in the earlier work.

[2] Note that the precise number of edges proposed by the parser and other common indicators of complexity can be obtained only while or after parsing. We are interested in *predicting* parsing complexity.

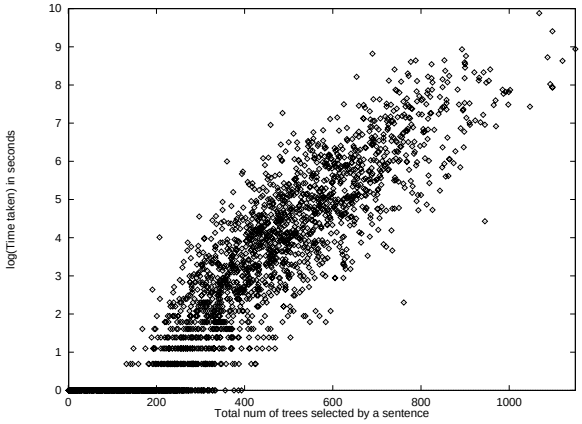

Figure 4: The impact of syntactic lexical ambiguity on parsing times. Log of the time taken to parse a sentence plotted against the total number of trees selected by the sentence. Coefficient of determination: $R^2 = 0.82$. (x-axis: Total number of trees selected by a sentence; y-axis: log(time) in seconds).

We test the above hypothesis further by parsing the same set of sentences as above but this time using an oracle which tells us the correct elementary lexicalized structure for each word in the sentence. This eliminates lexical syntactic ambiguity but does not eliminate attachment ambiguity for the parser. The graph comparing the parsing times is shown in Figure 5. As the comparison shows, the elimination of lexical ambiguity leads to a drastic increase in parsing efficiency. The total time taken to parse all 2250 sentences went from 548K seconds to 31.2 seconds.

Figure 5 shows us that a model which disambiguates syntactic lexical ambiguity can potentially be extremely useful in terms of parsing efficiency. Thus disambiguation of tree assignment or SuperTagging (Srinivas, 1997) of a sentence before parsing it might be a way of improving parsing efficiency. This gives us a way to reduce the parsing complexity for precisely the sentences which were problematic: the ones which selected too many trees. To test whether parsing times are reduced after SuperTagging we conducted an experiment in which the output of an n-best SuperTagger was taken as input to the parser. In our experiment we set n to be 60.[3] The time taken to parse the same set of sentences was again dramatically reduced (the total time taken was 21K seconds). However, the disadvantage of this method was that the coverage of the parser was reduced: 926 sentences (out of the 2250) did not get any parse. This was because some crucial tree was missing in the n-best output. The results are graphed in Figure 6. The total number of derivations for all sentences went down to 1.01e+10 (the original total number was 1.4e+18) indicating (not surprisingly) that some attachment ambiguities persist although the number of trees are reduced. We are experimenting with techniques where the output of the n-best SuperTagger is combined with other pieces of evidence to improve the coverage of the parser while retaining the speedup.

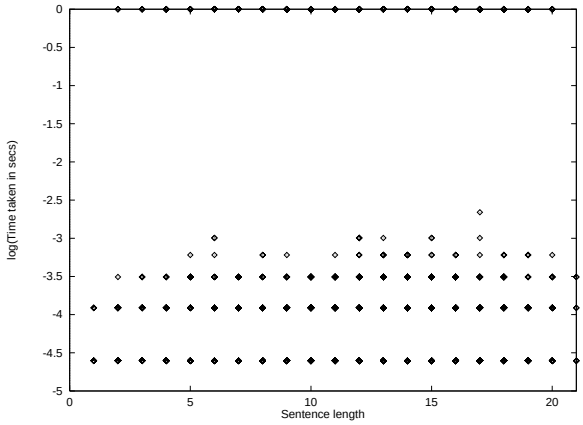

Figure 5: Parse times when the parser gets the correct tree for each word in the sentence (eliminating any syntactic lexical ambiguity). The parsing times for all the 2250 sentences for all lengths never goes above 1 second. (x-axis: Sentence length; y-axis: log(time) in seconds)

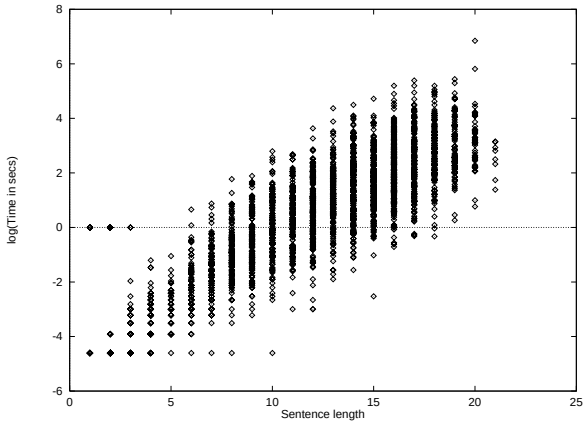

Figure 6: Time taken by the parser after n-best SuperTagging ($n = 60$). (x-axis: Sentence length; y-axis: log(time) in seconds)

4 Sentence Complexity

There are many ways of describing sentence complexity, which are not necessarily independent of

[3](Chen et al., 1999) shows that to get greater than 97% accuracy using SuperTagging the value of n must be quite high ($n > 40$). They use a different set of SuperTags and so we used their result simply to get an approximate estimate of the value of n.

each other. In the context of lexicalized tree-adjoining grammar (and in other lexical frameworks, perhaps with some modifications) the complexity of syntactic and semantic processing is related to the number of predicate-argument structures being computed for a given sentence.

In this section, we explore the possibility of characterizing sentence complexity in terms of the number of clauses which is used as an approximation to the number of predicate-argument structures to be found in a sentence.

The number of clauses of a given sentence in the Penn Treebank is counted using the bracketing tags. The count is computed to be the number of S/SINV/SQ/RRC nodes which have a VP child or a child with -PRD function tag. In principle number of clauses can grow continuously as the sentence length increases. However it is interesting to note that 99.1% of sentences in the Penn Treebank contain 6 or fewer clauses.

Figure 7 shows the average number of clauses plotted against sentence length. For sentences with no more than 50 words, which accounts for 98.2% of the corpus, we see a linear increase in the average number of clauses with respect to sentence length. But from that point on, increasing the sentence length does not lead to a proportional increase in the number of clauses. Thus, empirically, the number of clauses is bounded by a constant. For some very long sentences, the number of clauses actually decreases because these sentences include long but flat coordinated phrases.

Figure 8 shows the standard deviation of the clause number plotted against sentence length. There is an increase in deviation for sentences longer than 50 words. This is due to two reasons: first, quite often, long sentences either have many embedded clauses or are flat with long coordinated phrases; second, the data become sparse as the sentence length grows, resulting in high deviation.[4]

In Figure 9 and Figure 10 we show how parsing time varies as a function of the number of clauses present in the sentence being parsed. The figures are analogous to the earlier graphs relating parsing time with other factors (see Figure 3 and Figure 4). Surprisingly, in both graphs we see that when the number of clauses is small (in this case less than 5), an increase in the number of clauses has no effect on the parsing complexity. Even when the number of clauses is 1 we find the same pattern of time complexity that we have seen in the earlier graphs when we ignored clause complexity. Thus, when the number of clauses is small parsing complexity

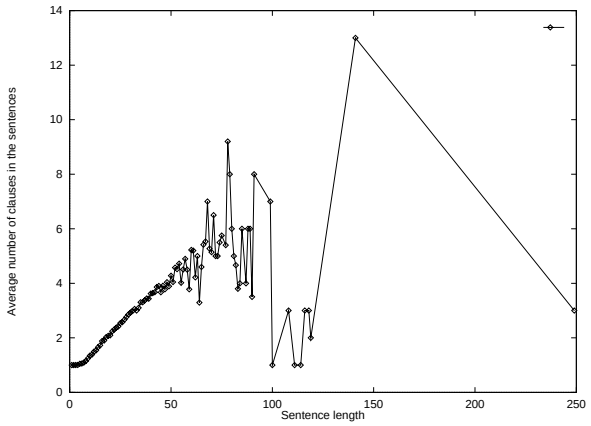

Figure 7: Average number of clause plotted against sentence length

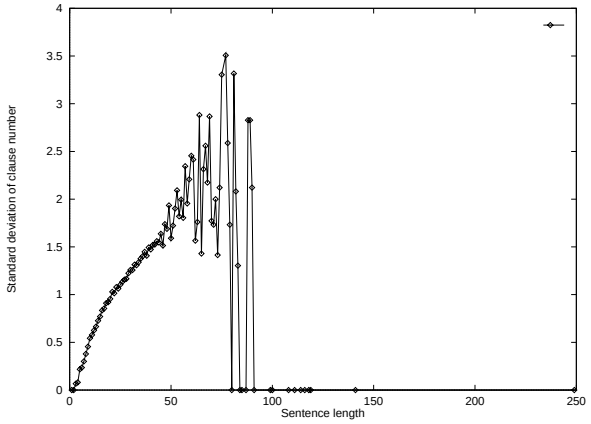

Figure 8: Standard deviation of clause number plotted against sentence length

is related to attachment of adjuncts rather than arguments. It would be interesting to continue increasing the number of clauses and the sentence length and then compare the differences in parsing times.[5]

We have seen that beyond a certain sentence length, the number of clauses do not increase proportionally. We conjecture that a parser can exploit this observed constraint on clause complexity in sentences to improve its efficiency. In a way similar to methods that account for low attachment of adjuncts while parsing, we can introduce constraints on how many clauses a particular node can dominate in a parse. By making the parser sensitive to this measure, we can prune out unlikely derivations previously considered to be plausible by the parser. There is also an independent reason for pursuing this measure of clausal complexity. It can be extended to a notion of syntactic and semantic complexity as they relate to both the representational

[4]For some sentence lengths (e.g., length $= 250$), there is only one sentence with that length in the whole corpus, resulting in zero deviation.

[5]We plan to conduct this experiment and present the results during the workshop.

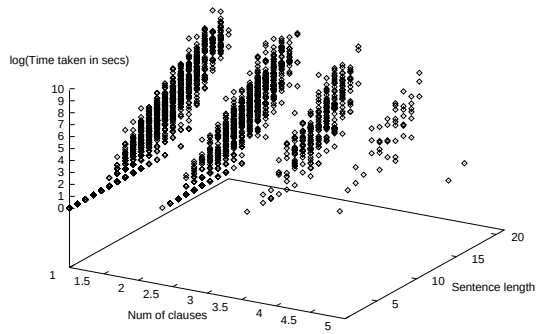

Figure 9: Variation in times for parsing plotted against length of each sentence while identifying the number of clauses.

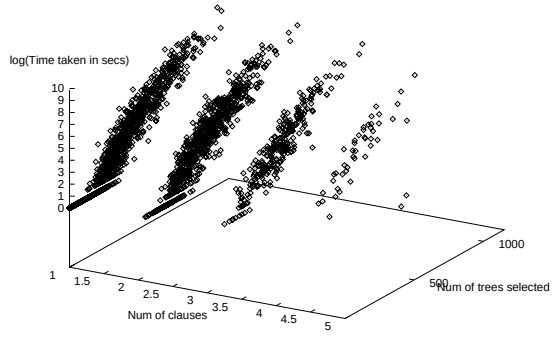

Figure 10: Variation in times for parsing plotted against the number of trees selected by each sentence while identifying the number of clauses.

and processing aspects (Joshi, 2000). The empirical study of clausal complexity described in this section might shed some light on the general issue of syntactic and semantic complexity.

5 Conclusion

In this paper, we identified syntactic lexical ambiguity and sentence complexity as factors that contribute to parsing complexity in fully lexicalized grammars.

We showed that lexical syntactic ambiguity has a strong effect on parsing time and that a model which disambiguates syntactic lexical ambiguity can potentially be extremely useful in terms of parsing efficiency. By assigning each word in the sentence with the correct elementary tree showed that parsing times were reduced by several orders of magnitude (the total time taken to parse 2250 sentences went from 548K seconds to 31.2 seconds).

We conducted an experiment in which the output of an n-best SuperTagger was taken as input to the parser. The time taken to parse the same set of sentences was again dramatically reduced (the total time taken was 21K seconds). The disadvantage of this approach was that 926 out of the original 2250 sentences did not get any parse.

We showed that even as sentence length increases the number of clauses is empirically bounded by a constant. The number of clauses in 99.1% of sentences in the Penn Treebank was bounded by 6. We discussed how this finding affects parsing efficiency and showed that for when the number of clauses is smaller than 4, parsing efficiency is dominated by adjunct attachments rather than argument attachments.

References

John Chen, Srinivas Bangalore, and K. Vijay-Shanker. 1999. New models for improving supertag disambiguation. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics*, Bergen, Norway.

A. Joshi and Y. Schabes. 1991. Tree adjoining grammars and lexicalized grammars. In M. Nivat and A. Podelski, editors, *Tree automata and languages*. North-Holland.

Aravind K. Joshi. 2000. Some aspects of syntactic and semantic complexity and underspecification. Talk given at *Syntactic and Semantic Complexity in Natural Language Processing Systems, Workshop at ANLP-NAACL 2000, Seattle*, May.

M. Marcus, B. Santorini, and M. Marcinkiewiecz. 1993. Building a large annotated corpus of english. *Computational Linguistics*, 19(2):313–330.

Stephan Oepen and John Carroll. 2000. Ambiguity packing in constraint-based parsing – practical results. In *Proceedings of the 1st Meeting of the North American ACL, NAACL-2000*, Seattle, Washington, Apr 29 – May 4.

Anoop Sarkar. 2000. Practical experiments in parsing using tree adjoining grammars. In *Proceedings of the Fifth Workshop on Tree Adjoining Grammars*, Paris, France, May 25–27.

The XTAG-Group. 1998. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS 98-18, University of Pennsylvania.

B. Srinivas. 1997. Performance Evaluation of Supertagging for Partial Parsing. In *Proceedings of Fifth International Workshop on Parsing Technology*, Boston, USA, September.

Fei Xia and Martha Palmer. 2000. Evaluating the Coverage of LTAGs on Annotated Corpora. In *Proceedings of LREC satellite workshop Using Evaluation within HLT Programs: Results and Trends*.

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proc. of NLPRS-99*, Beijing, China.