

Coping with problems in grammars automatically extracted from treebanks

Carlos A. Prolo

Computer and Information Science Department
University of Pennsylvania
Suite 400A, 3401 Walnut Street
Philadelphia, PA, USA, 19104-6228
prolo@linc.cis.upenn.edu

Abstract

We report in this paper on an experiment on automatic extraction of a Tree Adjoining Grammar from the WSJ corpus of the Penn Treebank. We use an automatic tool developed by (Xia, 2001) properly adapted to our particular need. Rather than addressing general aspects of the automatic extraction we focus on the problems we have found to extract a linguistically (and computationally) sound grammar and approaches to handle them.

1 Introduction

Much linguistic research is oriented to finding general principles for natural language, classifying linguistic phenomena, building regular models (e.g., grammars) for the well-behaved (or well-understood) part of languages and studying remaining “interesting” problems in a compartmentalized way. With the availability of large natural language corpora annotated for syntactic structure, the treebanks, e.g., (Marcus et al., 1993), automatic grammar extraction became possible (Chen and Vijay-Shanker, 2000; Xia, 1999). Suddenly, grammars started being extracted with an attempt to have “full” coverage of the constructions in a certain language (of course, to the extent that the used corpora represents the language) and that immediately poses a question: *If we do not know how to model many phenomena grammatically how can that be that we are extracting such a wide-coverage grammar?*

To answer that question we have to start a new thread at the edge of linguistics and computational linguistics. More than numbers to express coverage, we have to start analyzing the quality of automatically generated grammars, identifying extraction problems and uncovering whatever solutions are being given for them, however interesting or ugly they might be, challenging the current paradigms of linguistic research to provide answers for the problems on a “by-need” basis.

In this paper we report on a particular experience of automatic extraction of an English grammar from the WSJ corpus of the Penn Treebank (PTB) (Marcus et al., 1994)¹ using Tree Adjoining Grammar (TAGs, (Joshi and Schabes, 1997)). We use an automatic tool developed by (Xia, 2001) properly adapted to our particular needs and focus on some problems we have found to extract a linguistically (and computationally) sound grammar and the solutions we gave to them. The list of problems is a sample, far from being exhaustive² Likewise, the solutions will not always be satisfactory.

In Section 2 we introduce the method of grammar extraction employed. The problems are discussed in Section 3. We conclude in Section 4.

2 The extracted grammar

2.1 TAGs

A TAG is a set of lexicalized **elementary trees** that can be combined, through the operations of **tree adjunction** and **tree substitution**, to derive syntactic structures for sentences. We follow a common approach to grammar development for natural language using TAGs, under which, driven by locality principles, each elementary tree for a given lexical head is expected to contain its projection, and slots for its arguments (e.g., (Frank, 2002)). Figure 1 shows typical grammar template trees that can be selected by lexical items and combined to generate the structure in Figure 2. The *derivation tree*, to the right, contains the history of the tree grafting process that generated the *derived tree*, to the left.³

¹We assume some familiarity with the basic notations in the PTB as in (Marcus et al., 1994).

²(Prolo, 2002) includes a more comprehensive and detailed discussion of grammar extraction alternatives and problems.

³For a more comprehensive introduction to TAGs and Lexicalized TAGs we refer the reader to (Joshi and Schabes, 1997).

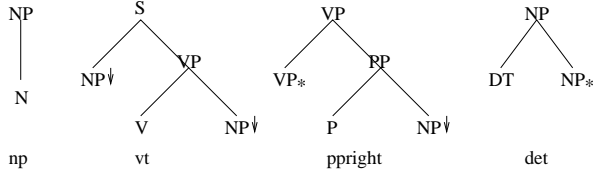


Figure 1: An example of Tree Adjoining Grammar

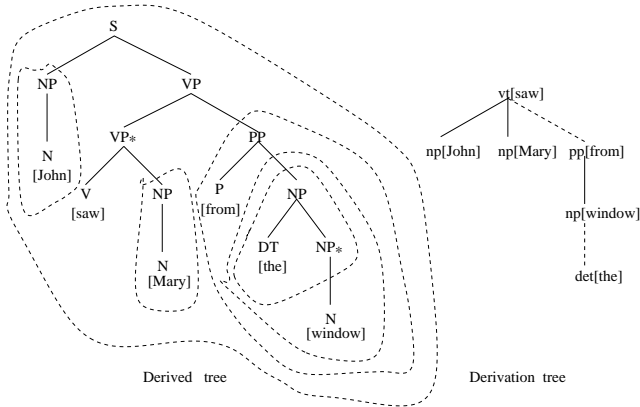


Figure 2: Derivation of *John saw Mary from the window*

2.2 LexTract

Given an annotated sentence from the PTB as input Xia’s LexTract tool (Xia, 1999; Xia, 2001) first executes a rebracketing. More precisely, additional nodes are inserted to separate arguments and modifiers and to structure the modifying process as binary branching. A typical rebracketed PTB tree is shown in Figure 3,⁴ in which we have distinguished the tree nodes inserted by LexTract.

The second stage is the extraction of the grammar trees proper shown in Figure 4. In particular, recursive modifier structures have to be detected and factored out of the derived tree to compose the auxiliary trees, the rest becoming an initial tree. The process is recursive also in the sense that factored subtree structures still undergo the spinning off process until we have all modifiers with their own trees, all the arguments of a head as substitution nodes of the tree containing their head, and the material under the argument nodes defining additional initial trees for themselves. Auxiliary trees are extracted from parent-child pairs with matching labels if the child is elected the parent’s head and the child’s sibling is marked as modifier: the parent is mapped into a root of an auxiliary tree, the head-child into its

⁴Figures 3 and 4 are thanks to Fei Xia. We are also grateful to her for allowing us to use LexTract and make changes to its source code to customize to our needs.

foot, with the sibling subtree (after being recursively processed) being carried together into the auxiliary tree. Notice that the auxiliary trees are therefore either strictly right or left branching, the foot always immediately under the root node. Other kinds of auxiliary trees are therefore not allowed.

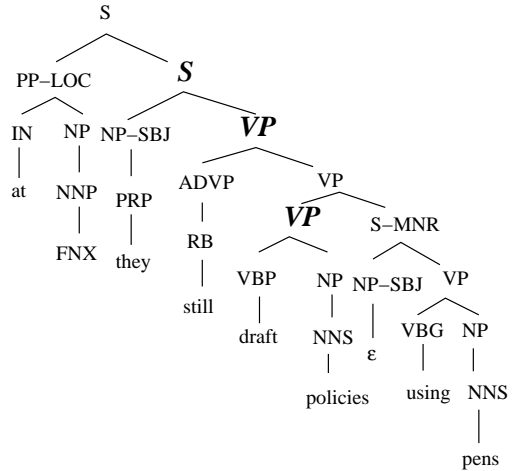


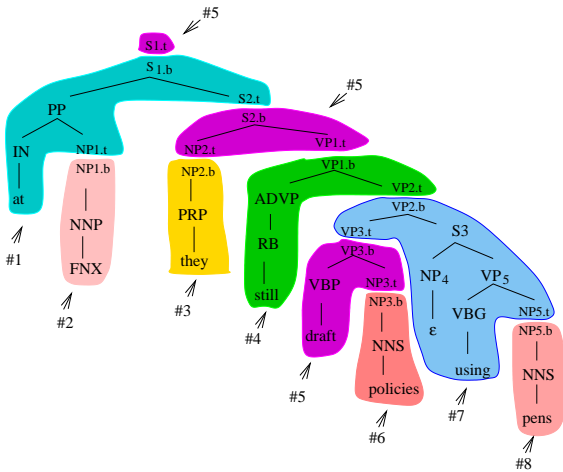
Figure 3: LexTract rebracketing stage

To extract a grammar with Xia’s tool one has to define tables for finding: the *head child* of a constituent expansion; which of the siblings of a head are acceptable arguments; and which constituent labels are plausible modifiers of another. Special provisions are made for handling coordination. For additional information see (Xia, 2001). In this paper we refer to (Xia, 1999)’s table settings and extracted grammar, which we used as our starting point, as Xia’s *sample*. We used a customized version of LexTract, plus additional pre-processing of the PTB input and post-processing of the extracted trees.

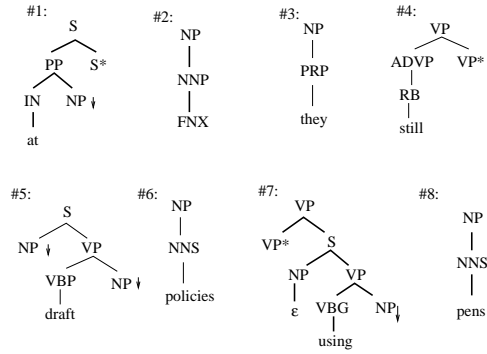
3 Extraction Problems

Extraction problems arise from several sources, including: (1) lack of proper linguistic account,⁵ (2) the (Penn Treebank) annotation style, (3) the (LexTract) extraction tool, (4) possible unsuitability of the (TAG) model, and (5) annotation errors. We refrained from making a rigid classification of the problems we present according to these sources. In particular it is often difficult to decide whether to blame sources (1), (3), or (5) for a certain problem. We will not discuss in this paper problems due to annotation errors. As for the PTB style problems we only discuss one, the first listed below.

⁵Here included the (occasional) inability on the part of grammar developers to find or make use of an existing account.



a) Input tree decomposition



b) Extracted elementary trees

Figure 4: LexTract extraction stage

```
(S-3 (NP-SBJ (PRP We))
      (VP (VBP make)
           (SBAR-NOM (WHNP-1 (WP what))
                      (S we know
                          how to make))))
```

a) As a sentential clause in the PTB

```
(S-3 (NP-SBJ (PRP We))
      (VP (VBP make)
           (NP (NP (WP what))
                (SBAR (WHNP-1 (-NONE- 0))
                      (S we know ...))))
```

b) As a Noun phrase after pre-processed

Figure 5: Free relatives in the Treebank

3.1 Free Relatives

Free relatives are annotated in the Penn Treebank as sentential complements as in Figure 5.a. The extracted tree corresponding to the occurrence of “make” would be of a verb that takes a sentential complement (SBAR). This does not seem to be correct⁶, as the proper subcategorization of the verb occurrence is transitive.

In fact, free relatives may occur wherever an NP argument may occur. So, the only reasonable extraction account consistent with maintaining them as SBARs would be one in which every NP substitution node in an extracted tree would admit the

⁶In both standard accounts for free relatives, the *Head Account* (e.g., (Bresnan and Grimshaw, 1978)) and the *Comp Account* (e.g., (Groos and von Riemsdijk, 1979)), commonly discussed in the literature, the presence of the NP (or *DP*) is clear.

existence of a counterpart tree, identical to the first, except that the NP argument label is replaced with an SBAR. Instead we opted to reflect the NP character of the free relatives by pre-processing the corpus (using the *Head-analysis*, for practical convenience). The annotated example is then automatically replaced with the one in Figure 5.b. Other cases of free-relatives (non-NP) are rare and not likely to interfere with verb subcategorization.

3.2 Wh percolation up

In the Penn Treebank the same constituent is annotated with different syntactic categories depending on whether it possesses or not the *wh* feature. For instance, a regular noun phrase has the syntactic category NP, whereas when the constituent is *wh*-marked, and is in the landing site of *wh*-movement, it carries the label WHNP.⁷ While that might look appealing since the two constituents seem to have distinct distributional properties, it poses a design problem. While regular constituents inherit their syntactic categorial feature (i.e. their label) from their heads, *wh* projections are often formed by inheritance from their modifiers. For instance: “the father” is an NP, but modified by a *wh* expression (“the father of whom”, “whose father”, “which father”), it becomes a WHNP. The only solution we see is to allow for nouns and NPs to freely project up to WHNPs during extraction.⁸ On the other hand, in

⁷When the constituent is not *wh*-moved, it is correctly preserved as an NP, as “what” in “Who ate what?”.

⁸Of course another simple solution would be merging the *wh* constituents with their non-*wh* counterparts.

```

(NP (UCP (NN construction)
         (CC and)
         (JJ commercial))
     (NNS loans))

a) NP modifiers

(VP (VB be)
     (UCP-PRD (NP (CD 35))
                (CC or)
                (ADJP (JJR older))))

b) non-verbal Predicates

(VP (VB take)
     (NP (NN effect))
     (UCP-TMP (ADVP 96 days later)
                (, ,)
                (CC or)
                (PP in early February)))

c) adverbial modifiers

```

Figure 6: “Unlike Coordinated Phrases”

cases when the wh constituent is in a non-wh position, we need the opposite effect: a WHNP (or wh-noun POS tag) is allowed to project up to an NP.

3.3 Unlike Coordinated Phrases (UCP)

This is the expression used in the PTB to denote coordinated phrases in which the coordinated constituents are not of the same syntactic category. The rationale for the existence of such constructions is that the coordinated constituents are alternative realizations of the same grammatical function with respect to a lexical head. In Figure 6.a, both a noun and an adjective are allowed to modify another noun, and therefore they can be conjoined while realizing that function. Two other common cases are: coordination of predicates in copular constructions (Figure 6.b) and adverbial modification (Figure 6.c).

We deal with the problem as follows. First, we allow for a UCP to be extracted as an argument when the head is a verb and the UCP is marked predicative (PRD function tag) in the training example; or whenever the head is seen to have an obligatory argument requirement (e.g., prepositions: “They come from ((NP the house) and (PP behind the tree))”). Second, a UCP is allowed to modify (adjoin to) most of the nodes, according to evidence in the corpus and common sense (in the first and third examples above we had NP and VP modification). With respect to the host tree, when attached as an argument they are treated like any other non-terminal: a substitution node. The left tree in Figure 7 shows

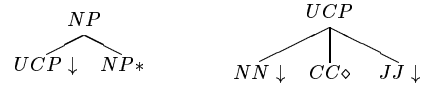


Figure 7: Extracted trees for UCP

the case where the UCP is treated as a modifier. In fact the trees are both for the example in Figure 6.a. Notice that the tree is non-lexicalized to avoid effects of sparseness. The UCP is then expanded as in the right tree in Figure 7: an initial tree anchored by the conjunction (the tree attaches either to a tree like the one in the left or as a true argument – the latter would be the case for the example in Figure 6.b).

Now, the caveats. First, we are giving the UCP the status of an independent non-terminal, as if it had some intrinsic categorial significance (as a syntactic projection). The assumption of independence of expansion, that for context-free grammars is inherent to each non-terminal, in TAGs is further restricted to the substitution nodes. For example, when an NP appears as substitution node, in a subject or object position, or as an argument of a preposition or a genitive marker, we are stating that any possible expansion for the NP is licensed there. The same happens for other labels in argument positions as well. While that is an overgenerating assumption (e.g. the expletive “there” cannot be the realization of an NP in object position), it is generally true. For the UCP, however, we know that its expansion is in fact strongly dependent on where the substitution node is, as we have argued before. In fact it is lexically dependent (cf. “I know ((the problem) and (that there is no solution to it))”, where the conjuncts are licensed by the subcategorizations of the verb “know”). On the other hand, it does not seem reasonable to expand the UCP node at the hosting tree – a cross product explosion. A possible way of alleviating this effect could be to expand only the auxiliary trees (a UCP modifying a VP is distinct from a UCP modifying an NP, and moreover they are independent of lexical items). But for true argument positions there seems to be no clear solution.

Second, the oddity of the UCP as a label becomes apparent once again when there are multiple conjuncts, as in Figure 8: it is enough for one of them to be distinct to turn the entire constituent into a UCP. Recursive decomposition in the grammar in these situations clearly leads to some non-standard trees.

Finally, and more crucially, we have omitted one case in our discussion: the case in which the UCP

```
(NP (UCP (JJ electronic)
      ( , , )
      (NN computer)
      (CC and)
      (NN building))
  (NNS products))
```

Figure 8: UCP with multiple conjuncts

```
(S (NP-SBJ-1 The Series 1989 B bonds)
  (VP (VBP are)
      (VP (VBN rated)
            (S *-1 double-A))))
(S (NP-SBJ-1 The Series 1989 B bonds)
  (VP (VBP are)
      (UCP-PRD (ADJP-PRD (JJ uninsured))
                (CC and)
                (VP (VBN rated)
                      (S *-1 double-A))))))
```

Figure 9: UCP involving VP argument of the copula

is the natural head-child of some node. Under some accounts of grammar development this never happens: we have observed that UCP does not appear as head child in the account where the head is the *syntactic head* of a node. We have not always followed this rule. With respect to the VP head, so far we have followed one major tendency in the computational implementation of lexicalized grammars, according to which *lexical verbs* are preferred to *auxiliary verbs* to head the VP. Now, consider the pair of sentences in Figure 9.

Under the *lexical verb* paradigm, in the first sentence the derivation would start with an initial tree anchored by the past participle verb (“rated”). But then we have an interesting problem in the second sentence, for which we do not currently have a neat solution. Following Xia’s *sample* settings of LexTract parameters, in these cases the extraction is rescued by switching to the other paradigm: the initial tree is extracted anchored by the auxiliary verb with a UCP argument, and the VP is accepted as a possible conjunct. A systematic move to the *syntactic head* paradigm, which we may indeed try, would have important consequences in the locality assumptions for the grammar development.

3.4 VP topicalization

Another problem with the lexical verb paradigm (see also discussion under UCP above) is the VP topicalization as in the sentence in Figure 10. The solution currently adopted (again, inherited from

```
(SINV (ADVP (RB Also))
      (VP-TPC-2 (VBN excluded)
                 (NP (-NONE- *-1)))
      (VP (MD will)
          (VP (VB be)
              (VP (-NONE- *T*-2))))
      (NP-SBJ-1 investments in ...))
```

Figure 10: VP topicalization

```
(S (NP-SBJ (NNP Congress))
  (VP (MD could)
      (VP (VB pass)
            (ADVP-MNR (RB quickly))
            (NP (NP (DT a)
                    (' ' ' '))
                  (JJ clean)
                  (' ' ' '))
                  (NN bill))
            (VP (VBG containing)
                (ADVP (JJ only))
                (NP ... ))))))
```

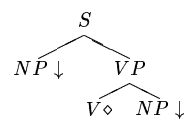


Figure 11: The extraposition problem

Xia’s sample settings) is as above: the paradigm is switched and the auxiliary verb (“be”) is chosen as the anchor of the initial tree.

3.5 Extraposition and Verb Subcategorization

One of the key design principles that have been guiding grammar development with TAGs is to keep verb arguments as substitution slots local to the tree anchored by the verb. It is widely known that the Penn Treebank does not distinguish verb objects from adjuncts. So some sorts of heuristics are needed to decide, among the candidates, which are to be taken as arguments (Kinyon and Prolo, 2002); the rest is extracted as separate VP modifier trees. However, this step is not enough for the trees to correctly reflect verb subcategorizations. The occurrence of discontinuous arguments, frequently explained as *argument extraposition* (the argument is raised past the adjunct) creates a problem. In the sentence in Figure 11 the verb “pass” should anchor a tree with one NP object.

However in such a tree it would be impossible to adjoin the tree for the intervening ADVP “quickly” as a VP modifier and still have it between the verb and the NP.⁹ LexTract then would instead extract an

⁹A striking use of *sister adjunction* in (Chiang, 2000) is exactly the elegant way it solves this problem: the non-argument tree can be adjoined onto a node (say, VP), positioning itself in between the VP’s children, which is not possible with TAGs.

```
(NP (NP the 3 billion New Zealand dollars)
  (PRN (-LRB- -LRB-)
    (NP US$ 1.76 billion *U*)
    (-RRB- -RRB-)))
```

a) A parenthetical NP attached to another NP

```
(S (NP-SBJ The total relationship)
  (PRN (, ,)
    (SBAR-ADV as Mr. Lee sees it)
    (, ,))
  (VP (VBZ is) ...))
```

b) A parenthetical S between subject and verb

Figure 12: Parentheticals

intransitive tree for the VB “pass”, onto which the ADVP modifier tree would adjoin. The second oddity is that the NP object would also be extracted as a VP modifier tree. In a nutshell, objects in extracted trees are restricted to those which are not extraposed and hence the trees may not truly reflect the proper domain of locality. One view is that the set of trees for a certain subcategorization frame would include these degenerate cases. LexTract has an option to allow limited discontinuity, i.e., a non-argument sequence between the verb and the first object (but not between two objects). The non-arguments would then be adjoined to the V node.¹⁰ So far we have used only the latter alternative.

It is worth mentioning two other cases of extraposition. Subject extraposition is handled by having the extraposed subject, usually a sentential form, adjoin at the VP of which it is the logical subject (the original position is still occupied by an NP with the expletive pronoun “it”). Relative clause extraposition is modeled by a relative clause tree, only it adjoins at a VP, instead of at an NP as is usual.

3.6 Parentheticals

Parenthetical expressions are ubiquitous in language: they may appear almost everywhere in a sentence and can be of almost any category (Fig. 12).

We model them as adjoining, either to the left or right of the constituent they are dominated by, depending on whether they are to the left or right of the head child of the parent’s node. Occasionally such trees can also be initial. The respective trees for the examples of Figure 12 are drawn in Figure 13. It

¹⁰Of course, although the solution covers most of the occurrences, and apart of any linguistic concern, there are still uncovered cases, e.g., when a parenthetical expression intervenes between the first and the second argument.

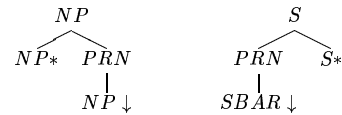


Figure 13: Extracted trees for parentheticals

is always the case that the label PRN dominates a single substitution node. Whenever this was not the case in the training corpus, heuristics based on observation were used to enforce that, by inserting an appropriate missing node.

3.7 Projection labels

LexTract extracts trees with no concern for the appropriate projective structure of constituents when not explicitly marked in the PTB. Figure 14 shows two examples of NP modification where the modifiers are single lexical items. The extracted modifier trees, shown on the right, do not have the projection for the modifiers JJR “stronger” and the NNP “October” (which should be, respectively, an ADJP and an NP). That is so, because those nodes are not found in the annotation.

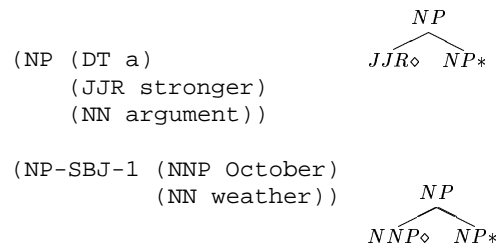


Figure 14: Simple modification annotation and extracted trees

However, if the modifiers are complex, that is, if the modifiers are themselves modified, the PTB inserts their respective projections, and therefore they appear in the extracted trees, as shown in Figure 15.

There seems to be no reason for the two pairs of extracted trees to be different. Much of this is caused by the acknowledged flatness in the Penn Treebank annotation. That said, the trees like those in the second pair should be preferred. The projection node (ADJP or NP) is understood to be dominating its head even when there is no further modification, and it should be a concern of a good extraction process to insert the missing node into the grammar. Since LexTract do not allow us to spec-

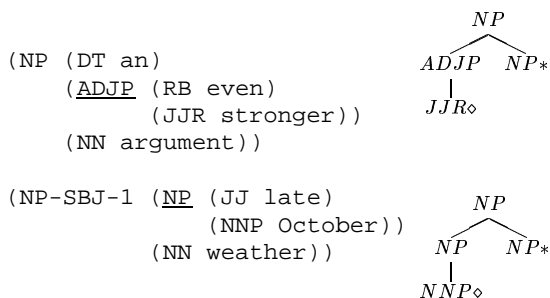


Figure 15: Complex modification annotation and extracted trees

ify for the insertion of “obligatory” projections we had to accomplish this through a somewhat complicated post-processing step using a projection table. Some of our current projections are: nouns, personal pronouns and the existential expletive to NP; adjectives to ADJP; adverbs to ADVP; sentences either to SBAR (S, SINV) or to SBARQ (SQ); Cardinals (CD) to Quantifier Phrases (QP) which themselves project to NP. Notice that not all categories are forcefully projected. For instance, verbs are not, allowing for simple auxiliary extraction. IN is also not projected due to its double role as PP head (true preposition) and subordinate conjunction, which should project onto SBARs.

4 Conclusion

We discussed an experiment in grammar extraction from corpora with focus on problems arising while trying to give an adequate account for naturally occurring phenomena. Without being exhaustive in our list, we expect to have brought some attention to the need to discuss solutions for them which are as reasonable as possible given the current state-of-the-art of the linguistic research, computational grammar development and automatic extraction, and given the current corpus resources at our disposition.

ACKNOWLEDGEMENTS: Thanks to Tonia Bleam, Erwin Chan, Alexandra Kinyon, Rashmi Prasad, Beatrice Santorini, Fei Xia and the XTAG Group for valuable discussions along the realization of this work and/or comments on this paper or related material.

References

Joan Bresnan and Jane Grimshaw. 1978. The syntax of free relatives in english. *Linguistic Inquiry*, 9(3):331–391.

John Chen and K. Vijay-Shanker. 2000. Automated extraction of TAGs from the Penn Treebank. In *Proceedings of the 6th International Workshop on Parsing Technologies*, Trento, Italy.

David Chiang. 2000. Statistical parsing with an automatically-extracted Tree Adjoining Grammar. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong, China.

Robert Frank. 2002. *Phrase Structure Composition and Syntactic Dependencies*. MIT Press, Cambridge, MA, USA.

Anneke Groos and Henk von Riemsdijk. 1979. The matching effects in free relatives: a parameter of core grammar. In *Theory of Markedness in Generative Grammar*. Scuola Normale Superiore di Pisa, Italy.

Aravind K. Joshi and Yves Schabes. 1997. Tree-Adjoining Grammars. In *Handbook of Formal Languages*, volume 3, pages 69–123. Springer-Verlag, Berlin.

Alexandra Kinyon and Carlos A. Prolo. 2002. Identifying verb arguments and their syntactic function in the Penn Treebank. In *Proc. of the Third LREC*, pages 1982–87, Las Palmas, Spain.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the 1994 Human Language Technology Workshop*.

Carlos A. Prolo. 2002. LR parsing for Tree Adjoining Grammars and its application to corpus-based natural language parsing. Ph.D. Dissertation Proposal, University of Pennsylvania.

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, Beijing, China.

Fei Xia. 2001. *Investigating the Relationship between Grammars and Treebanks for Natural Languages*. Ph.D. thesis, Department of Computer and Information Science, Un. of Pennsylvania.