An Empirical Evaluation of Probabilistic Lexicalized Tree Insertion Grammars *

Rebecca Hwa
Harvard University
Cambridge, MA 02138 USA
rebecca@eecs.harvard.edu

Abstract

We present an empirical study of the applicability of Probabilistic Lexicalized Tree Insertion Grammars (PLTIG), a lexicalized counterpart to Probabilistic Context-Free Grammars (PCFG), to problems in stochastic natural-language processing. Comparing the performance of PLTIGs with non-hierarchical N-gram models and PCFGs, we show that PLTIG combines the best aspects of both, with language modeling capability comparable to N-grams, and improved parsing performance over its non-lexicalized counterpart. Furthermore, training of PLTIGs displays faster convergence than PCFGs.

1 Introduction

There are many advantages to expressing a grammar in a *lexicalized form*, where an observable word of the language is encoded in each grammar rule. First, the lexical words help to clarify ambiguities that cannot be resolved by the sentence structures alone. For example, to correctly attach a prepositional phrase, it is often necessary to consider the lexical relationships between the head word of the prepositional phrase and those of the phrases it might modify. Second, lexicalizing the grammar rules increases computational efficiency because those rules that do not contain any observed words can be pruned away immediately. The Lexicalized Tree Insertion Grammar formalism (LTIG) has been proposed as a way to lexicalize context-free grammars (Schabes and Waters, 1994). We now apply a probabilistic variant of this formalism, Probabilistic Tree Insertion Grammars (PLTIGs), to natural language processing problems of stochastic parsing and language modeling. This paper presents two sets of experiments, comparing PLTIGs with non-lexicalized Probabilistic Context-Free Grammars (PCFGs) (Pereira and Schabes, 1992) and non-hierarchical N-gram models that use the right branching bracketing heuristics (period attaches high) as their parsing strategy. We show that PLTIGs can be induced from partially bracketed data, and that the resulting trained grammars can parse unseen sentences and estimate the likelihood of their occurrences in the language. The experiments are run on two corpora: the Air Travel Information System (ATIS) corpus and a subset of the Wall Street Journal TreeBank corpus. The results show that the lexicalized nature of the formalism helps our induced PLTIGs to converge faster and provide a better language model than PCFGs while maintaining comparable parsing qualities. Although N-gram models still slightly out-perform PLTIGs on language modeling, they lack high level structures needed for parsing. Therefore, PLTIGs have combined the best of two worlds: the language modeling capability of N-grams and the parse quality of context-free grammars.

The rest of the paper is organized as follows: first, we present an overview of the PLTIG formalism; then we describe the experimental setup; next, we interpret and discuss the results of the experiments; finally, we outline future directions of the research.

2 PLTIG and Related Work

The inspiration for the PLTIG formalism stems from the desire to lexicalize a context-free gram-

* This material is based upon work supported by the National Science Foundation under Grant No. IRI 9712068. We thank Yves Schabes and Stuart Shieber for their guidance; Joshua Goodman for his PCFG code; Lillian Lee and the three anonymous reviewers for their comments on the paper.

mar. There are three ways in which one might do so. First, one can modify the tree structures so that all context-free productions contain lexical items. Greibach normal form provides a well-known example of such a lexicalized context-free formalism. This method is not practical because altering the structures of the grammar damages the linguistic information stored in the original grammar (Schabes and Waters, 1994). Second, one might propagate lexical information upward through the productions. Examples of formalisms using this approach include the work of Magerman (1995), Charniak (1997), Collins (1997), and Goodman (1997). A more linguistically motivated approach is to expand the domain of productions downward to incorporate more tree structures. The Lexicalized Tree-Adjoining Grammar (LTAG) formalism (Schabes et al., 1988), (Schabes, 1990) , although not context-free, is the most well-known instance in this category. PLTIGs belong to this third category and generate only context-free languages.

LTAGs (and LTIGs) are tree-rewriting systems, consisting of a set of elementary trees combined by tree operations. We distinguish two types of trees in the set of elementary trees: the *initial trees* and the *auxiliary trees*. Unlike full parse trees but reminiscent of the productions of a context-free grammar, both types of trees may have nonterminal leaf nodes. Auxiliary trees have, in addition, a distinguished nonterminal leaf node, labeled with the same nonterminal as the root node of the tree, called the *foot* node. Two types of operations are used to construct *derived trees*, or parse trees: substitution and adjunction. An initial tree can be *substituted* into the nonterminal leaf node of another tree in a way similar to the substitution of nonterminals in the production rules of CFGs. An auxiliary tree is inserted into another tree through the adjunction operation, which splices the auxiliary tree into the target tree at a node labeled with the same nonterminal as the root and foot of the auxiliary tree. By using a tree representation, LTAGs extend the domain of locality of a grammatical primitive, so that they capture both lexical features and hierarchical structure. Moreover, the adjunction operation elegantly models intuitive linguistic concepts such as long distance dependencies between words. Unlike the N-gram model, which only offers dependencies between neighboring words, these trees can model the interaction of structurally related words that occur far apart.

Like LTAGs, LTIGs are tree-rewriting systems, but they differ from LTAGs in their generative power. LTAGs can generate some strictly context-sensitive languages. They do so by using *wrapping* auxiliary trees, which allow non-empty frontier nodes (i.e., leaf nodes whose labels are not the empty terminal symbol) on both sides of the foot node. A wrapping auxiliary tree makes the formalism context-sensitive because it coordinates the string to the left of its foot with the string to the right of its foot while allowing a third string to be inserted into the foot. Just as the ability to recursively center-embed moves the required parsing time from $O(n)$ for regular grammars to $O(n^3)$ for context-free grammars, so the ability to wrap auxiliary trees moves the required parsing time further, to $O(n^6)$ for tree-adjoining grammars [1]. This level of complexity is far too computationally expensive for current technologies. The complexity of LTAGs can be moderated by eliminating just the wrapping auxiliary trees. LTIGs prevent wrapping by restricting auxiliary tree structures to be in one of two forms: the *left auxiliary tree*, whose non-empty frontier nodes are all to the left of the foot node; or the *right auxiliary tree*, whose non-empty frontier nodes are all to the right of the foot node. Auxiliary trees of different types cannot adjoin into each other if the adjunction would result in a wrapping auxiliary tree. The resulting system is strongly equivalent to CFGs, yet is fully lexicalized and still $O(n^3)$ parsable, as shown by Schabes and Waters (1994).

Furthermore, LTIGs can be parameterized to form probabilistic models (Schabes and Waters, 1993). Informally speaking, a parameter is associated with each possible adjunction or substitution operation between a tree and a node. For instance, suppose there are V left auxiliary trees that might adjoin into node η. Then there are $V+1$ parameters associated with node η

[1] The best theoretical upper bound on time complexity for the recognition of Tree Adjoining Languages is $O(M(n^2))$, where $M(k)$ is the time needed to multiply two $k \times k$ boolean matrices.(Rajasekaran and Yooseph, 1995)

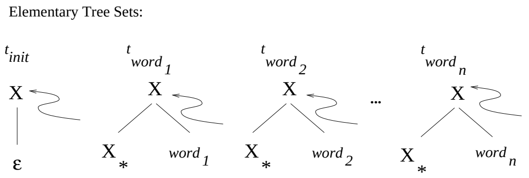

Figure 1: A set of elementary LTIG trees that represent a bigram grammar. The arrows indicate adjunction sites.

that describe the distribution of the likelihood of any left auxiliary tree adjoining into node η. (We need one extra parameter for the case of no left adjunction.) A similar set of parameters is constructed for the right adjunction and substitution distributions.

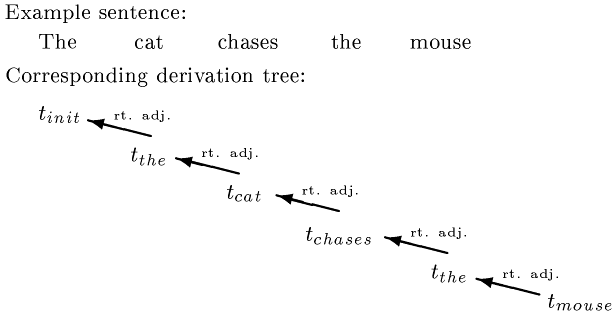

Figure 2: An example sentence. Because each tree is right adjoined to the tree anchored with the neighboring word in the sentence, the only structure is right branching.

3 Experiments

In the following experiments we show that PLTIGs of varying sizes and configurations can be induced by processing a large training corpus, and that the trained PLTIGs can provide parses on unseen test data of comparable quality to the parses produced by PCFGs. Moreover, we show that PLTIGs have significantly lower entropy values than PCFGs, suggesting that they make better language models. We describe the induction process of the PLTIGs in Section 3.1. Two corpora of very different nature are used for training and testing. The first set of experiments uses the Air Travel Information System (ATIS) corpus. Section 3.2 presents the complete results of this set of experiments. To determine if PLTIGs can scale up well, we have also begun another study that uses a larger and more complex corpus, the Wall Street Journal TreeBank corpus. The initial results are discussed in Section 3.3. To reduce the effect of the data sparsity problem, we back off from lexical words to using the part of speech tags as the anchoring lexical items in all the experiments. Moreover, we use the deleted-interpolation smoothing technique for the N-gram models and PLTIGs. PCFGs do not require smoothing in these experiments.

3.1 Grammar Induction

The technique used to induce a grammar is a subtractive process. Starting from a universal grammar (i.e., one that can generate any string made up of the alphabet set), the parameters are iteratively refined until the grammar generates, hopefully, all and only the sentences in the target language, for which the training data provides an adequate sampling. In the case of a PCFG, the initial grammar production rule set contains all possible rules in Chomsky Normal Form constructed by the nonterminal and terminal symbols. The initial parameters associated with each rule are randomly generated subject to an admissibility constraint. As long as all the rules have a non-zero probability, any string has a non-zero chance of being generated. To train the grammar, we follow the Inside-Outside re-estimation algorithm described by Lari and Young (1990). The Inside-Outside re-estimation algorithm can also be extended to train PLTIGs. The equations calculating the inside and outside probabilities for PLTIGs can be found in Hwa (1998).

As with PCFGs, the initial grammar must be able to generate any string. A simple PLTIG that fits the requirement is one that simulates a bigram model. It is represented by a tree set that contains a right auxiliary tree for each lexical item as depicted in Figure 1. Each tree has one adjunction site into which other right auxiliary trees can adjoin. The tree set has only one initial tree, which is anchored by an empty lexical item. The initial tree represents the start of the sentence. Any string can be constructed by right adjoining the words together in order. Training the parameters of this grammar yields the same result as a bigram model: the parameters reflect close correlations between words

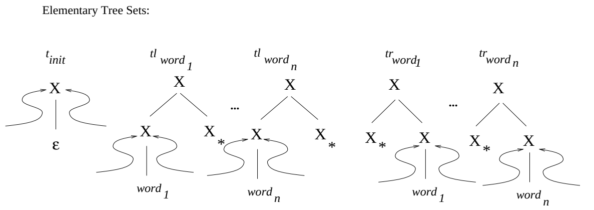

Figure 3: An LTIG elementary tree set that allow both left and right adjunctions.

that are frequently seen together, but the model cannot provide any high-level linguistic structure. (See example in Figure 2.)

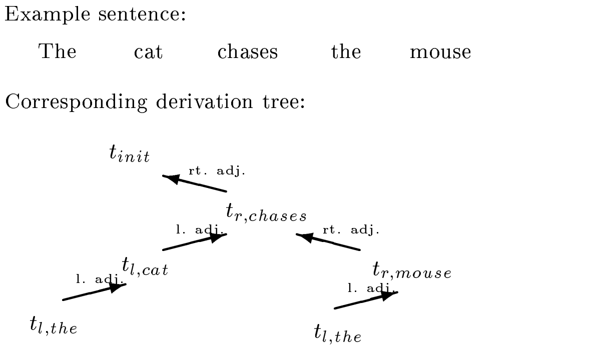

Figure 4: With both left and right adjunctions possible, the sentences can be parsed in a more linguistically plausible way

To generate non-linear structures, we need to allow adjunction in both left and right directions. The expanded LTIG tree set includes a left auxiliary tree representation as well as right for each lexical item. Moreover, we must modify the topology of the auxiliary trees so that adjunction in both directions can occur. We insert an intermediary node between the root and the lexical word. At this internal node, at most one adjunction of each direction may take place. The introduction of this node is necessary because the definition of the formalism disallows right adjunction into the root node of a left auxiliary tree and vice versa. For the sake of uniformity, we shall disallow adjunction into the root nodes of the auxiliary trees from now on. Figure 3 shows an LTIG that allows at most one left and one right adjunction for each elementary tree. This enhanced LTIG can produce hierarchical structures that the bigram model could not (See Figure 4.)

It is, however, still too limiting to allow only one adjunction from each direction. Many words often require more than one modifier. For example, a transitive verb such as "give" takes at least two adjunctions: a direct object noun phrase, an indirect object noun phrase, and possibly other adverbial modifiers. To create more adjunction sites for each word, we introduce yet more intermediary nodes between the root and the lexical word. Our empirical studies show that each lexicalized auxiliary tree requires at least 3 adjunction sites to parse all the sentences in the corpora. Figure 5(a) and (b) show two examples of auxiliary trees with 3 adjunction sites. The number of parameters in a PLTIG is dependent on the number of adjunction sites just as the size of a PCFG is dependent on the number of nonterminals. For a language with V vocabulary items, the number of parameters for the type of PLTIGs used in this paper is $2(V+1)+2V(K)(V+1)$, where K is the number of adjunction sites per tree. The first term of the equation is the number of parameters contributed by the initial tree, which always has two adjunction sites in our experiments. The second term is the contribution from the auxiliary trees. There are $2V$ auxiliary trees, each tree has K adjunction sites; and $V+1$ parameters describe the distribution of adjunction at each site. The number of parameters of a PCFG with M nonterminals is $M^3 + MV$. For the experiments, we try to choose values of K and M for the PLTIGs and PCFGs such that

$$2(V+1) + 2V(K)(V+1) \approx M^3 + MV$$

3.2 ATIS

To reproduce the results of PCFGs reported by Pereira and Schabes, we use the ATIS corpus for our first experiment. This corpus contains 577 sentences with 32 part-of-speech tags. To ensure statistical significance, we generate ten random train-test splits on the corpus. Each set randomly partitions the corpus into three sections according to the following distribution: 80% training, 10% held-out, and 10% testing. This gives us, on average, 406 training sentences, 83 testing sentences, and 88 sentences for held-out testing. The results reported here are the averages of ten runs.

We have trained three types of PLTIGs, varying the number of left and right adjunction sites. The L2R1 version has two left adjunction sites and one right adjunction site; L1R2 has one

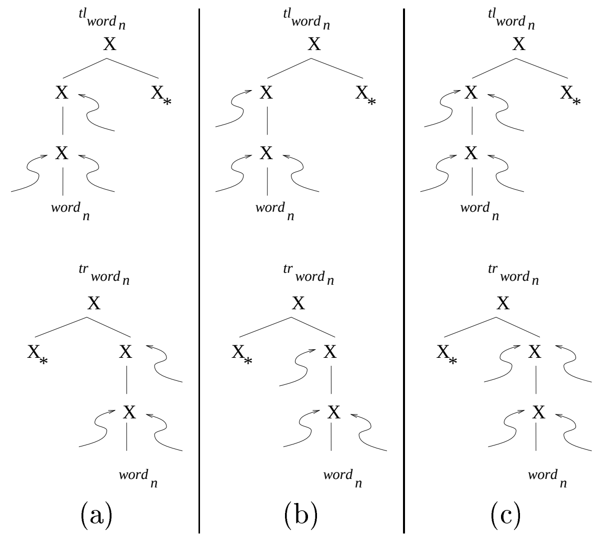

Figure 5: Prototypical auxiliary trees for three PLTIGs: (a) L1R2, (b) L2R1, and (c) L2R2.

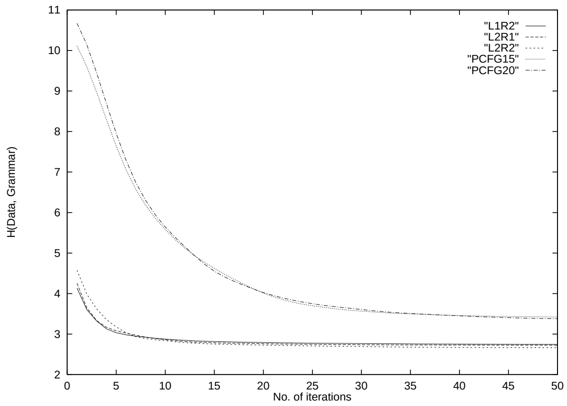

Figure 6: Average convergence rates of the training process for 3 PLTIGs and 2 PCFGs.

left adjunction site and two right adjunction sites; L2R2 has two of each. The prototypical auxiliary trees for these three grammars are shown in Figure 5. At the end of every training iteration, the updated grammars are used to parse sentences in the held-out test sets D, and the new language modeling scores (by measuring the cross-entropy estimates $\hat{H}(D, L2R1)$, $\hat{H}(D, L1R2)$, and $\hat{H}(D, L2R2)$) are calculated. The rate of improvement of the language modeling scores determines convergence. The PLTIGs are compared with two PCFGs: one with 15-nonterminals, as Pereira and Schabes have done, and one with 20-nonterminals, which has comparable number of parameters to L2R2, the larger PLTIG.

In Figure 6 we plot the average iterative improvements of the training process for each grammar. All training processes of the PLTIGs converge much faster (both in numbers of iterations and in real time) than those of the PCFGs, even when the PCFG has fewer parameters to estimate, as shown in Table 1. From Figure 6, we see that both PCFGs take many more iterations to converge and that the cross-entropy value they converge on is much higher than the PLTIGs.

During the testing phase, the trained grammars are used to produce bracketed constituents on unmarked sentences from the testing sets T. We use the crossing bracket metric to evaluate the parsing quality of each grammar. We also measure the cross-entropy estimates $\hat{H}(T, L2R1)$, $\hat{H}(T, L1R2)$,$\hat{H}(T, L2R2)$, $\hat{H}(T, PCFG_{15})$, and $\hat{H}(T, PCFG_{20})$ to determine the quality of the language model. For a baseline comparison, we consider bigram and trigram models with simple right branching bracketing heuristics. Our findings are summarized in Table 1.

The three types of PLTIGs generate roughly the same number of bracketed constituent errors as that of the trained PCFGs, but they achieve a much lower entropy score. While the average entropy value of the trigram model is the lowest, there is no statistical significance between it and any of the three PLTIGs. The relative statistical significance between the various types of models is presented in Table 2. In any case, the slight language modeling advantage of the trigram model is offset by its inability to handle parsing.

Our ATIS results agree with the findings of Pereira and Schabes that concluded that the performances of the PCFGs do not seem to depend heavily on the number of parameters once a certain threshold is crossed. Even though $PCFG_{20}$ has about as many number of parameters as the larger PLTIG (L2R2), its language modeling score is still significantly worse than that of any of the PLTIGs.

	Bigram/Trigram	PCFG 15	PCFG 20	L1R2	L2R1	L2R2
Number of parameters	1088 / 34880	3855	8640	6402	6402	8514
Iterations to convergence	–	45	45	19	17	24
Real-time convergence (min)	–	62	142	8	7	14
$\hat{H}(T, Grammar)$	2.88 / 2.71	3.81	3.42	2.87	2.85	2.78
Crossing bracket (on T)	66.78	93.46	93.41	93.07	93.28	94.51

Table 1: Summary results for ATIS. The machine used to measure real-time is an HP 9000/859.

	Bigram/Trigram	PCFG 15	PCFG 20	PCFG 23	L1R2	L2R1	L2R2
Number of parameters	2400 / 115296	4095	8960	13271	14210	14210	18914
Iterations to convergence	–	80	60	70	28	30	28
Real-time convergence (hr)	–	143	252	511	38	41	60
$\hat{H}(T, Grammar)$	3.39/3.20	4.31	4.27	4.13	3.58	3.56	3.59
Crossing bracket (T)	49.44	56.41	78.82	79.30	80.08	82.43	80.832

Table 3: Summary results for WSJ

PLTIGs	better		
bigram	better	–	
trigram	better	–	better
	PCFGs	PLTIGs	bigram

Table 2: Summary of pair-wise t-test for all grammars. If "better" appears at cell (i,j), then the model in row i has an entropy value lower than that of the model in column j in a statistically significant way. The symbol "–" denotes that the difference of scores between the models bears no statistical significance.

3.3 WSJ

Because the sentences in ATIS are short with simple and similar structures, the difference in performance between the formalisms may not be as apparent. For the second experiment, we use the Wall Street Journal (WSJ) corpus, whose sentences are longer and have more varied and complex structures. We use sections 02 to 09 of the WSJ corpus for training, section 00 for held-out data D, and section 23 for test T. We consider sentences of length 40 or less. There are 13242 training sentences, 1780 sentences for the held-out data, and 2245 sentences in the test. The vocabulary set consists of the 48 part-of-speech tags. We compare three variants of PCFGs (15 nonterminals, 20 nonterminals, and 23 nonterminals) with three variants of PLTIGs (L1R2, L2R1, L2R2). A PCFG with 23 nonterminals is included because its size approximates that of the two smaller PLTIGs. We did not generate random train-test splits for the WSJ corpus because it is large enough to provide adequate sampling. Table 3 presents our findings. From Table 3, we see several similarities to the results from the ATIS corpus. All three variants of the PLTIG formalism have converged at a faster rate and have far better language modeling scores than any of the PCFGs. Differing from the previous experiment, the PLTIGs produce slightly better crossing bracket rates than the PCFGs on the more complex WSJ corpus. At least 20 nonterminals are needed for a PCFG to perform in league with the PLTIGs. Although the PCFGs have fewer parameters, the rate seems to be indifferent to the size of the grammars after a threshold has been reached. While upping the number of nonterminal symbols from 15 to 20 led to a 22.4% gain, the improvement from $PCFG_{20}$ to $PCFG_{23}$ is only 0.5%. Similarly for PLTIGs, L2R2 performs worse than L2R1 even though it has more parameters. The baseline comparison for this experiment results in more extreme outcomes. The right branching heuristic receives a crossing bracket rate of 49.44%, worse than even that of $PCFG_{15}$. However, the N-gram models have better cross-entropy measurements than PCFGs and PLTIGs; bigram has a score of 3.39 bits per word, and trigram has a score of 3.20 bits per word. Because the lexical relationship modeled by the PLTIGs presented in this paper is limited to those between two words, their scores are close to that of the bigram model.

4 Conclusion and Future Work

In this paper, we have presented the results of two empirical experiments using Probabilistic Lexicalized Tree Insertion Grammars. Comparing PLTIGs with PCFGs and N-grams, our studies show that a lexicalized tree representation drastically improves the quality of language modeling of a context-free grammar to the level of N-grams without degrading the parsing accuracy. In the future, we hope to continue to improve on the quality of parsing and language modeling by making more use of the lexical information. For example, currently, the initial untrained PLTIGs consist of elementary trees that have uniform configurations (i.e., every auxiliary tree has the same number of adjunction sites) to mirror the CNF representation of PCFGs. We hypothesize that a grammar consisting of a set of elementary trees whose number of adjunction sites depend on their lexical anchors would make a closer approximation to the "true" grammar. We also hope to apply PLTIGs to natural language tasks that may benefit from a good language model, such as speech recognition, machine translation, message understanding, and keyword and topic spotting.

References

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the AAAI*, pages 598–603, Providence, RI. AAAI Press/MIT Press.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the ACL*, pages 16–23, Madrid, Spain.

Joshua Goodman. 1997. Probabilistic feature grammars. In *Proceedings of the International Workshop on Parsing Technologies 1997*.

Rebecca Hwa. 1998. An empirical evaluation of probabilistic lexicalized tree insertion grammars. Technical Report 06-98, Harvard University. Full Version.

K. Lari and S.J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56.

David Magerman. 1995. Statistical decision–models for parsing. In *Proceedings of the 33rd Annual Meeting of the ACL*, pages 276–283, Cambridge, MA.

Fernando Pereira and Yves Schabes. 1992. Inside-Outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the ACL*, pages 128–135, Newark, Delaware.

S. Rajasekaran and S. Yooseph. 1995. Tal recognition in $O(M(n^2))$ time. In *Proceedings of the 33rd Annual Meeting of the ACL*, pages 166–173, Cambridge, MA.

Y. Schabes and R. Waters. 1993. Stochastic lexicalized context-free grammar. In *Proceedings of the Third International Workshop on Parsing Technologies*, pages 257–266.

Y. Schabes and R. Waters. 1994. Tree insertion grammar: A cubic-time parsable formalism that lexicalizes context-free grammar without changing the tree produced. Technical Report TR-94-13, Mitsubishi Electric Research Laboratories.

Y. Schabes, A. Abeille, and A. K. Joshi. 1988. Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING '88)*, August.

Yves Schabes. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars.* Ph.D. thesis, University of Pennsylvania, August.