Estimating Compact Yet Rich Tree Insertion Grammars

Elif Yamangil and **Stuart M. Shieber**
Harvard University
Cambridge, Massachusetts, USA
{elif, shieber}@seas.harvard.edu

Abstract

We present a Bayesian nonparametric model for estimating tree insertion grammars (TIG), building upon recent work in Bayesian inference of tree substitution grammars (TSG) via Dirichlet processes. Under our general variant of TIG, grammars are estimated via the Metropolis-Hastings algorithm that uses a context free grammar transformation as a proposal, which allows for cubic-time string parsing as well as tree-wide joint sampling of derivations in the spirit of Cohn and Blunsom (2010). We use the Penn treebank for our experiments and find that our proposal Bayesian TIG model not only has competitive parsing performance but also finds compact yet linguistically rich TIG representations of the data.

1 Introduction

There is a deep tension in statistical modeling of grammatical structure between providing good expressivity — to allow accurate modeling of the data with sparse grammars — and low complexity — making induction of the grammars and parsing of novel sentences computationally practical. Recent work that incorporated Dirichlet process (DP) nonparametric models into TSGs has provided an efficient solution to the problem of segmenting training data trees into elementary parse tree fragments to form the grammar (Cohn et al., 2009; Cohn and Blunsom, 2010; Post and Gildea, 2009). DP inference tackles this problem by exploring the space of all possible segmentations of the data, in search for fragments that are on the one hand large enough so that they incorporate the useful dependencies, and on the other small enough so that they recur and have a chance to be useful in analyzing unseen data.

The elementary trees combined in a TSG are, intuitively, primitives of the language, yet certain linguistic phenomena (notably various forms of modification) "split them up", preventing their reuse, leading to less sparse grammars than might be ideal. For instance, imagine modeling the following set of structures:

- [$_{NP}$ the [$_{NN}$ [$_{NN}$ [$_{NN}$ president] of the university] who resigned yesterday]]
- [$_{NP}$ the [$_{NN}$ former [$_{NN}$ [$_{NN}$ president] of the university]]]
- [$_{NP}$ the [$_{NN}$ [$_{NN}$ president] who resigned yesterday]]

A natural recurring structure here would be the structure "[$_{NP}$ the [$_{NN}$ president]]", yet it occurs not at all in the data.

TSGs are a special case of the more flexible grammar formalism of tree adjoining grammar (TAG) (Joshi et al., 1975). TAG augments TSG with an *adjunction operator* and a set of *auxiliary trees* in addition to the substitution operator and initial trees of TSG, allowing for "splicing in" of syntactic fragments within trees. In the example, by augmenting a TSG with an operation of adjunction, a grammar that hypothesizes auxiliary trees corresponding to adjoining "[$_{NN}$ former NN]", "[$_{NN}$ NN of the university]", and "[$_{NN}$ NN who resigned yesterday]" would be able to reuse the basic structure "[$_{NP}$ the [$_{NN}$ president]]".

Unfortunately, TAG's expressivity comes at the cost of greatly increased complexity. Parsing complexity for unconstrained TAG scales as $O(n^6)$, im-

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 110–114,
Jeju, Republic of Korea, 8-14 July 2012. ©2012 Association for Computational Linguistics

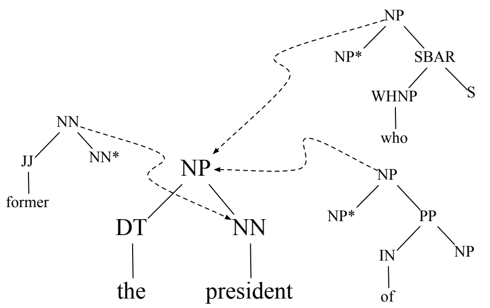

Figure 1: Example TIG derivation of an NP constituent: One left insertion (at NN) and two simultaneous right insertions (at NP).

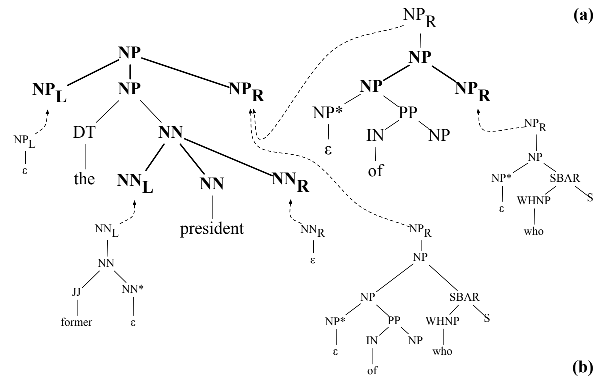

Figure 2: **TIG-to-TSG transform:** (a) and (b) illustrate transformed TSG derivations for two different TIG derivations of the same parse tree structure. The TIG nodes where we illustrate the transformation are in bold. (We suppress the rest of the transformational nodes.)

practical as compared to CFG and TSG's $O(n^3)$. In addition, the model selection problem for TAG is significantly more complicated than for TSG since one must reason about many more combinatorial options with two types of derivation operators.[1] This has led researchers to resort to heuristic grammar extraction techniques (Chiang, 2000; Carreras et al., 2008) or using a very small number of grammar categories (Hwa, 1998).

Hwa (1998) first proposed to use tree-insertion grammars (TIG), a kind of expressive compromise between TSG and TAG, as a substrate on which to build grammatical inference. TIG constrains the adjunction operation so that spliced-in material falls completely to the left or completely to the right of the splice point. By restricting the form of possible auxiliary trees to only *left* or *right* auxiliary trees in this way, TIG remains within the realm of context-free formalisms (with cubic complexity) while still modeling rich linguistic phenomena (Schabes and Waters, 1995). Figure 1 depicts some examples of TIG derivations.

Sharing the same intuitions, Shindo et al. (2011) have provided a previous attempt at combining TIG and Bayesian nonparametric principles, albeit with severe limitations. Their TIG variant (which we will refer to as TIG_0) is highly constrained in the following ways.

1. The foot node in an auxiliary tree must be the immediate child of the root node.
2. Only one adjunction can occur at a given node.
3. Even modeling multiple adjunction with root adjunction is disallowed. There is thus no recursion possibility with adjunction, no stacking of auxiliary trees.
4. As a consequence of the prior two constraints, no adjunction along the spines of auxiliary trees is allowed.
5. As a consequence of the first constraint, all nonterminals along the spine of an auxiliary tree are identical.

In this paper we explore a Bayesian nonparametric model for estimating a far more expressive version of TIG, and compare its performance against TSG and the restricted TIG_0 variant. Our more general formulation avoids these limitations by supporting the following features and thus relaxing four of the five restrictions of TIG_0.

1. Auxiliary trees may have the foot node at depth greater than one.[2]
2. Both left and right adjunctions may occur at the same node.
3. Simultanous adjunction (that is, more than one left or right adjunction per node) is allowed via root adjunction.
4. Adjunctions may occur along the spines of auxiliary trees.

The increased expressivity of our TIG variant is motivated both linguistically and practically. From a linguistic point of view: Deeper auxiliary trees can help model large patterns of insertion and potential correlations between lexical items that extend over multiple levels of tree. Combining left and right auxiliary trees can help model modifiers of the same node from left and right (combination of adjectives

[1]This can be seen by the fact that tree-path languages under TAG are context free, whereas they are regular for TSG. (Schabes and Waters, 1995)

[2]Throughout the paper, we will refer to the depth of an auxiliary tree to indicate the length of its spine.

and relative clauses for instance). Simultaneous insertion allows us to deal with multiple independent modifiers for the same constituent (for example, a series of adjectives). From a practical point of view, we show that an induced TIG provides modeling performance superior to TSG and comparable with TIG_0. However we show that the grammars we induce are *compact yet rich*, in that they succinctly represent complex linguistic structures.

2 Probabilistic Model

In the basic nonparametric TSG model, there is an independent DP for every grammar category (such as $c = NP$), each of which uses a base distribution P_0 that generates an initial tree by making stepwise decisions.

$$G_c^{\text{init}} \sim \text{DP}(\alpha_c^{\text{init}}, P_0^{\text{init}}(\cdot \mid c))$$

The canonical P_0 uses a probabilistic CFG $\tilde{P}$ that is fixed a priori to sample CFG rules top-down and Bernoulli variables for determining where substitutions should occur (Cohn et al., 2009; Cohn and Blunsom, 2010).

We extend this model by adding specialized DPs for left and right auxiliary trees.[3]

$$G_c^{\text{right}} \sim \text{DP}(\alpha_c^{\text{right}}, P_0^{\text{right}}(\cdot \mid c))$$

Therefore, we have an exchangeable process for generating right auxiliary trees

$$p(a_j \mid \mathbf{a}_{<j}) = \frac{n_{a_j} + \alpha_c^{\text{right}} P_0^{\text{right}}(a_j \mid c)}{j - 1 + \alpha_c^{\text{right}}} \quad (1)$$

as for initial trees in TSG.

We must define three distinct base distributions for initial trees, left auxiliary trees, and right auxiliary trees. P_0^{init} generates an initial tree with root label c by sampling CFG rules from $\tilde{P}$ and making a binary decision at every node generated whether to leave it as a frontier node or further expand (with probability β_c) (Cohn et al., 2009). Similarly, our P_0^{right} generates a right auxiliary tree with root label c by first making a binary decision whether to generate an immediate foot or not (with probability γ_c^{right}), and then sampling an appropriate CFG rule from $\tilde{P}$. For the right child, we sample an initial tree from P_0^{init}. For the left child, if decision to generate an immediate foot was made, we generate a foot node, and stop. Otherwise we recur into P_0^{right} which generates a right auxiliary tree that becomes the left child.

[3] We use right insertions for illustration; the symmetric analog applies to left insertions.

```
(VP (, ,) (VP PP (VP (, ,) VP*)))
(VP (SBAR (WHADVP (WRB (WRB When) ) ) S) (VP (, ,) VP*))
(VP (PP (IN For) (NP NN )) (VP (, ,) VP*))
(VP (CC But) (VP PP (VP (, ,) VP*)))
(VP ADVP (VP (, ,) VP*))
(IN (ADVP (RB (RB particularly) ) ) IN*)
(NP PP (NP (CC and) (NP PP NP*)))
```

Figure 3: Example left auxiliary trees that occur in the top derivations for Section 23. Simultaneous insertions occur most frequently for the labels VP (85 times), NNS (21 times), NNP (14 times).

We bring together these three sets of processes via a set of insertion parameters $\mu_c^{\text{left}}, \mu_c^{\text{right}}$. In any derivation, for every initial tree node labelled c (except for frontier nodes) we determine whether or not there are insertions at this node by sampling a Bernoulli(μ_c^{left}) distributed left insertion variable and a Bernoulli(μ_c^{right}) distributed right insertion variable. For left auxiliary trees, we treat the nodes that are *not* along the spine of the auxiliary tree the same way we treat initial tree nodes, however for nodes that are along the spine (including root nodes, excluding foot nodes) we consider only left insertions by sampling the left insertion variable (symmetrically for right insertions).

3 Inference

Given this model, our inference task is to explore optimal derivations underlying the data. Since TIG derivations are highly structured objects, a basic sampling strategy based on local node-level moves such as Gibbs sampling (Geman and Geman, 1984) would not hold much promise. Following previous work, we design a blocked Metropolis-Hastings sampler that samples derivations per entire parse trees all at once in a joint fashion (Cohn and Blunsom, 2010; Shindo et al., 2011). This is achieved by proposing derivations from an approximating distribution and stochastically correcting via accept/reject to achieve convergence into the correct posterior (Johnson et al., 2007).

Since our base distributions factorize over levels of tree, CFG is the most convenient choice for a

CFG rule	CFG probability
	Base distribution: P_0^{init}
$\text{NP} \rightarrow \text{NP}^{\text{init}}$	$\alpha_c^{\text{init}} / (n_{\text{NP}}^{\text{init}} + \alpha_c^{\text{init}})$
$\text{NP}^{\text{init}} \rightarrow \text{NP}_{\text{L}}$ _NP^{init} NP_{R}	1.0
_$\text{NP}^{\text{init}} \rightarrow$ DT NN	$\tilde{P}(\text{NP} \rightarrow \text{DT NN}) \times (1 - \beta_{\text{DT}}) \times (1 - \beta_{\text{NN}})$
_$\text{NP}^{\text{init}} \rightarrow$ DT NN^{init}	$\tilde{P}(\text{NP} \rightarrow \text{DT NN}) \times (1 - \beta_{\text{DT}}) \times \beta_{\text{NN}}$
_$\text{NP}^{\text{init}} \rightarrow \text{DT}^{\text{init}}$ NN	$\tilde{P}(\text{NP} \rightarrow \text{DT NN}) \times \beta_{\text{DT}} \times (1 - \beta_{\text{NN}})$
_$\text{NP}^{\text{init}} \rightarrow \text{DT}^{\text{init}}$ NN^{init}	$\tilde{P}(\text{NP} \rightarrow \text{DT NN}) \times \beta_{\text{DT}} \times \beta_{\text{NN}}$
	Base distribution: P_0^{right}
$\text{NP}_{\text{R}} \rightarrow \text{NP}^{\text{right}}$	$\mu_{\text{NP}}^{\text{right}} \times \left(\alpha_c^{\text{right}} / (n_{\text{NP}}^{\text{right}} + \alpha_c^{\text{right}})\right)$
$\text{NP}_{\text{R}} \rightarrow \epsilon$	$1 - \mu_{\text{NP}}^{\text{right}}$
$\text{NP}^{\text{right}} \rightarrow$ _NP^{right} NP_{R}	1.0
_$\text{NP}^{\text{right}} \rightarrow \text{NP}*$ $\text{SBAR}^{\text{init}}$	$\tilde{P}(\text{NP} \rightarrow \text{NP SBAR} \mid \text{NP} \rightarrow \text{NP }\_) \times (1 - \gamma_{\text{NP}}^{\text{right}}) \times (1 - \beta_{\text{SBAR}})$
_$\text{NP}^{\text{right}} \rightarrow \text{NP}*$ SBAR	$\tilde{P}(\text{NP} \rightarrow \text{NP SBAR} \mid \text{NP} \rightarrow \text{NP }\_) \times (1 - \gamma_{\text{NP}}^{\text{right}}) \times \beta_{\text{SBAR}}$
_$\text{NP}^{\text{right}} \rightarrow \text{NP}^{\text{right}}$ $\text{SBAR}^{\text{init}}$	$\tilde{P}(\text{NP} \rightarrow \text{NP SBAR} \mid \text{NP} \rightarrow \text{NP }\_) \times \gamma_{\text{NP}}^{\text{right}} \times (1 - \beta_{\text{SBAR}})$
_$\text{NP}^{\text{right}} \rightarrow \text{NP}^{\text{right}}$ SBAR	$\tilde{P}(\text{NP} \rightarrow \text{NP SBAR} \mid \text{NP} \rightarrow \text{NP }\_) \times \gamma_{\text{NP}}^{\text{right}} \times \beta_{\text{SBAR}}$

Figure 4: Transformation CFG rules that represent infinite base distributions. P_0^{init} is taken from Cohn and Blunsom (2010). Underscored labels (such as _NP^{right} as opposed to NP^{right}) are used to differentiate the pre-insertion nodes in Figure 2 from the post-insertion ones. P_0^{left} rules are omitted for brevity and mirror the P_0^{right} rules above.

Model	FMeasure	# Initial Trees	# Auxiliary Trees (# Left)
TSG	77.51	6.2K	-
TIG_0	78.46	6.0K	251 (137)
TIG	78.62	5.6K	604 (334)

Figure 5: EVALB results after training on Section 2 and testing on Section 23. Note that TIG finds a compact yet rich representation. Elementary tree counts are based on ones with count > 1.

proposal distribution. Fortunately, Schabes and Waters (1995) provide an (exact) transformation from a fully general TIG into a TSG that generates the same string languages. It is then straightforward to represent this TSG as a CFG using the Goodman transform (Goodman, 2002; Cohn and Blunsom, 2010). Figure 4 lists the additional CFG productions we have designed, as well as the rules used that trigger them.

4 Evaluation Results

We use the standard Penn treebank methodology of training on sections 2–21 and testing on section 23. All our data is head-binarized and words occurring only once are mapped into unknown categories of the Berkeley parser. As has become standard, we carried out a small treebank experiment where we train on Section 2, and a large one where we train on the full training set. All hyperparameters are resampled under appropriate vague gamma and beta priors. All reported numbers are averages over three runs. Parsing results are based on the maximum probability parse which was obtained by sampling derivations under the transform CFG.

We compare our system (referred to as TIG) to our implementation of the TSG system of (Cohn and Blunsom, 2010) (referred to as TSG) and the constrained TIG variant of (Shindo et al., 2011) (referred to as TIG_0). The upshot of our experiments is that, while on the large training set all models have similar performance ($85.6, 85.3, 85.4$ for TSG, TIG_0 and TIG respectively), on the small dataset insertion helps nonparametric model to find more compact and generalizable representations for the data, which affects parsing performance (Figure 4). Although TIG_0 has performance close to TIG, note that TIG achieves this performance using a more succinct representation and extracting a rich set of auxiliary trees. As a result, TIG finds many chances to apply insertions to test sentences, whereas TIG_0 depends mostly on TSG rules. If we look at the most likely derivations for the test data, TIG_0 assigns 663 insertions (351 left insertions) in the parsing of entire Section 23, meanwhile TIG assigns 3924 (2100 left insertions). Some of these linguistically sophisticated auxiliary trees that apply to test data are listed in Figure 3.

5 Conclusion

We described a nonparametric Bayesian inference scheme for estimating TIG grammars and showed the power of TIG formalism over TSG for returning rich, generalizable, yet compact representations of data. The nonparametric inference scheme presents a principled way of addressing the difficult model selection problem with TIG which has been prohibitive in this area of research. TIG still remains within context free and both our sampling and parsing techniques are highly scalable.

Acknowledgements

The first author was supported in part by a Google PhD Fellowship in Natural Language Processing.

References

Xavier Carreras, Michael Collins, and Terry Koo. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, CoNLL '08, pages 9–16, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 456–463, Morristown, NJ, USA. Association for Computational Linguistics.

Trevor Cohn and Phil Blunsom. 2010. Blocked inference in Bayesian tree substitution grammars. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 225–230, Stroudsburg, PA, USA. Association for Computational Linguistics.

Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 548–556, Morristown, NJ, USA. Association for Computational Linguistics.

S. Geman and D. Geman. 1984. Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images. pages 6:721–741.

J. Goodman. 2002. Efficient parsing of DOP with PCFG-reductions. *Bod et al. 2003*.

Rebecca Hwa. 1998. An empirical evaluation of probabilistic lexicalized tree insertion grammars. In *Proceedings of the 17th international conference on Computational linguistics - Volume 1*, pages 557–563, Morristown, NJ, USA. Association for Computational Linguistics.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York, April. Association for Computational Linguistics.

Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1):136–163.

Matt Post and Daniel Gildea. 2009. Bayesian learning of a tree substitution grammar. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 45–48, Suntec, Singapore, August. Association for Computational Linguistics.

Remko Scha and Rens Bod. 2003. Efficient parsing of DOP with PCFG-reductions, October.

Yves Schabes and Richard C. Waters. 1995. Tree insertion grammar: a cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Comput. Linguist.*, 21:479–513, December.

Hiroyuki Shindo, Akinori Fujino, and Masaaki Nagata. 2011. Insertion operator for Bayesian tree substitution grammars. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 206–211, Stroudsburg, PA, USA. Association for Computational Linguistics.