Insertion Operator for Bayesian Tree Substitution Grammars

Hiroyuki Shindo, Akinori Fujino, and **Masaaki Nagata**
NTT Communication Science Laboratories, NTT Corp.
2-4 Hikaridai Seika-cho Soraku-gun Kyoto 619-0237 Japan
{shindo.hiroyuki,fujino.akinori,nagata.masaaki}@lab.ntt.co.jp

Abstract

We propose a model that incorporates an insertion operator in Bayesian tree substitution grammars (BTSG). Tree insertion is helpful for modeling syntax patterns accurately with fewer grammar rules than BTSG. The experimental parsing results show that our model outperforms a standard PCFG and BTSG for a small dataset. For a large dataset, our model obtains comparable results to BTSG, making the number of grammar rules much smaller than with BTSG.

1 Introduction

Tree substitution grammar (TSG) is a promising formalism for modeling language data. TSG generalizes context free grammars (CFG) by allowing nonterminal nodes to be replaced with subtrees of arbitrary size.

A natural extension of TSG involves adding an *insertion operator* for combining subtrees as in tree adjoining grammars (TAG) (Joshi, 1985) or tree insertion grammars (TIG) (Schabes and Waters, 1995). An insertion operator is helpful for expressing various syntax patterns with fewer grammar rules, thus we expect that adding an insertion operator will improve parsing accuracy and realize a compact grammar size.

One of the challenges of adding an insertion operator is that the computational cost of grammar induction is high since tree insertion significantly increases the number of possible subtrees. Previous work on TAG and TIG induction (Xia, 1999; Chiang, 2003; Chen et al., 2006) has addressed the problem using language-specific heuristics and a maximum likelihood estimator, which leads to overfitting the training data (Post and Gildea, 2009).

Instead, we incorporate an insertion operator in a Bayesian TSG (BTSG) model (Cohn et al., 2011) that learns grammar rules automatically without heuristics. Our model uses a restricted variant of subtrees for insertion to model the probability distribution simply and train the model efficiently. We also present an inference technique for handling a tree insertion that makes use of dynamic programming.

2 Overview of BTSG Model

We briefly review the BTSG model described in (Cohn et al., 2011). TSG uses a *substitution* operator (shown in Fig. 1a) to combine subtrees. Subtrees for substitution are referred to as *initial* trees, and leaf nonterminals in initial trees are referred to as *frontier* nodes. Their task is the unsupervised induction of TSG *derivations* from parse trees. A derivation is information about how subtrees are combined to form parse trees.

The probability distribution over initial trees is defined by using a Pitman-Yor process prior (Pitman and Yor, 1997), that is,

$$\begin{aligned} e\,|X &\sim G_X \\ G_X\,|d_X, \theta_X &\sim \mathrm{PYP}\left(d_X, \theta_X, P_0\left(\cdot\,|X\right)\right), \end{aligned}$$

where X is a nonterminal symbol, e is an initial tree rooted with X, and $P_0\left(\cdot\,|X\right)$ is a *base distribution* over the infinite space of initial trees rooted with X. d_X and θ_X are hyperparameters that are used to control the model's behavior. Integrating out all possible values of G_X, the resulting distribution is

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics:shortpapers, pages 206–211,
Portland, Oregon, June 19-24, 2011.

$$p\left(e_i \,|\mathbf{e}_{-i}, X, d_X, \theta_X\right) = \alpha_{e_i,X} + \beta_X P_0\left(e_i, |X\right), \quad (1)$$

where $\alpha_{e_i,X} = \frac{n^{-i}_{e_i,X} - d_X \cdot t_{e_i,X}}{\theta_X + n^{-i}_{\cdot,X}}$ and $\beta_X = \frac{\theta_X + d_X \cdot t_{\cdot,X}}{\theta_X + n^{-i}_{\cdot,X}}$. $\mathbf{e}_{-i} = e_1, \ldots, e_{i-1}$ are previously generated initial trees, and $n^{-i}_{e_i,X}$ is the number of times e_i has been used in $\mathbf{e}_{-i}$. $t_{e_i,X}$ is the number of *tables* labeled with e_i. $n^{-i}_{\cdot,X} = \sum_e n^{-i}_{e,X}$ and $t_{\cdot,X} = \sum_e t_{e,X}$ are the total counts of initial trees and tables, respectively. The PYP prior produces "*rich get richer*" statistics: a few initial trees are often used for derivation while many are rarely used, and this is shown empirically to be well-suited for natural language (Teh, 2006b; Johnson and Goldwater, 2009).

The base probability of an initial tree, $P_0\left(e\,|X\right)$, is given as follows.

$$P_0\left(e\,|X\right) = \prod_{r \in \text{CFG}(e)} P_{\text{MLE}}\left(r\right) \times \prod_{A \in \text{LEAF}(e)} s_A \times \prod_{B \in \text{INTER}(e)} \left(1 - s_B\right), \quad (2)$$

where CFG (e) is a set of decomposed CFG productions of e, $P_{\text{MLE}}\left(r\right)$ is a maximum likelihood estimate (MLE) of r. LEAF (e) and INTER (e) are sets of leaf and internal symbols of e, respectively. s_X is a *stopping probability* defined for each X.

3 Insertion Operator for BTSG

3.1 Tree Insertion Model

We propose a model that incorporates an insertion operator in BTSG. Figure 1b shows an example of an insertion operator. To distinguish them from initial trees, subtrees for insertion are referred to as *auxiliary* trees. An auxiliary tree includes a special nonterminal leaf node labeled with the same symbol as the root node. This leaf node is referred to as a *foot* node (marked with the subscript "*"). The definitions of substitution and insertion operators are identical with those of TIG and TAG.

Since it is computationally expensive to allow any auxiliary trees, we tackle the problem by introducing *simple auxiliary trees*, i.e., auxiliary trees whose root node must generate a foot node as an immediate child. For example, "(N (JJ pretty) N*)" is a simple auxiliary tree, but "(S (NP) (VP (V think) S*))" is not. Note that we place no restriction on the initial trees.

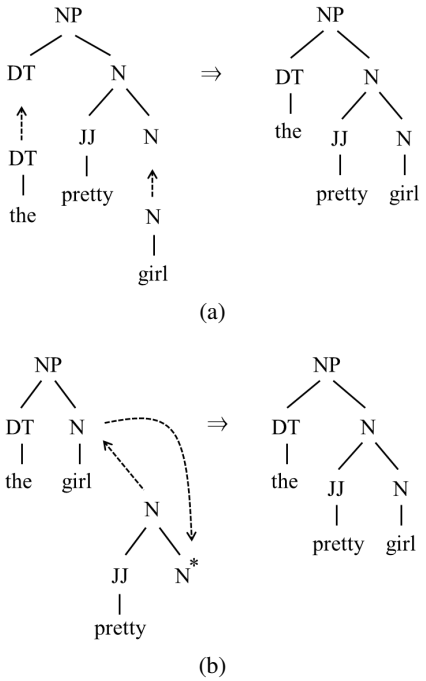

Figure 1: Example of (a) substitution and (b) insertion (dotted line).

Our restricted formalism is a strict subset of TIG. We briefly refer to some differences between TAG, TIG and our insertion model. TAG generates tree adjoining languages, a strict superset of context-free languages, and the computational complexity of parsing is $O\left(n^6\right)$. TIG is a similar formalism to TAG, but it does not allow wrapping adjunction in TAG. Therefore, TIG generates context-free languages and the parsing complexity is $O\left(n^3\right)$, which is a strict subset of TAG. On the other hand, our model prohibits neither wrapping adjunction in TAG nor simultaneous adjunction in TIG, and allows only simple auxiliary trees. The expressive power and computational complexity of our formalism is identical to TIG, however, our model allows us to define the probability distribution over auxiliary trees as having the same form as BTSG model. This ensures that we can make use of a dynamic programming technique for training our model, which we describe the detail in the next subsection.

We define a probability distribution over simple auxiliary trees as having the same form as eq. 1, that is,

$$p\left(e_i\,|\mathbf{e}_{-i}, X, d'_X, \theta'_X\right) = \alpha'_{e_i,X} + \beta'_X P'_0\left(e_i, |X\right), \quad (3)$$

where d'_X and θ'_X are hyperparameters of the insertion model, and the definition of $\left(\alpha'_{e_i,X}, \beta'_X\right)$ is the same as that of $(\alpha_{e_i,X}, \beta_X)$ in eq. 1.

However, we need modify the base distribution over simple auxiliary trees, $P'_0\left(e\,|X\right)$, as follows, so that all probabilities of the simple auxiliary trees sum to one.

$$\begin{aligned} P'_0\left(e\,|X\right) &= P'_{\text{MLE}}\left(\text{TOP}\left(e\right)\right) \times \prod_{r \in \text{INTER\_CFG}(e)} P_{\text{MLE}}\left(r\right) \\ &\times \prod_{A \in \text{LEAF}(e)} s_A \times \prod_{B \in \text{INTER}(e)} \left(1 - s_B\right), \quad (4) \end{aligned}$$

where $\text{TOP}\left(e\right)$ is the CFG production that starts with the root node of e. For example, TOP (N (JJ pretty) (N*)) returns "N → JJ N*". INTER_CFG $\left(e\right)$ is a set of CFG productions of e excluding $\text{TOP}\left(e\right)$. $P'_{\text{MLE}}\left(r'\right)$ is a modified MLE for simple auxiliary trees, which is given by

$$\begin{cases} \frac{C(r')}{C(X \to X^* Y) + C(X \to Y X^*)} & \text{if } r' \text{includes a foot node} \\ 0 & \text{else} \end{cases}$$

where $C\left(r'\right)$ is the frequency of r' in parse trees. It is ensured that $P'_0\left(e\,|X\right)$ generates a foot node as an immediate child.

We define the probability distribution over both initial trees and simple auxiliary trees with a PYP prior. The base distribution over initial trees is defined as $P_0\left(e\,|X\right)$, and the base distribution over simple auxiliary trees is defined as $P'_0\left(e\,|X\right)$. An initial tree e_i replaces a frontier node with probability $p\left(e_i\,|\mathbf{e}_{-i}, X, d_X, \theta_X\right)$. On the other hand, a simple auxiliary tree e'_i inserts an internal node with probability $a_X \times p'\left(e'_i\,|\mathbf{e}'_{-i}, X, d'_X, \theta'_X\right)$, where a_X is an insertion probability defined for each X. The stopping probabilities are common to both initial and auxiliary trees.

3.2 Grammar Decomposition

We develop a *grammar decomposition* technique, which is an extension of work (Cohn and Blunsom, 2010) on BTSG model, to deal with an insertion operator. The motivation behind grammar decomposition is that it is hard to consider all possible derivations explicitly since the base distribution assigns non-zero probability to an infinite number of initial and auxiliary trees. Alternatively, we transform a derivation into CFG productions and assign the probability for each CFG production so that its assignment is consistent with the probability distributions. We can efficiently calculate an inside probability (described in the next subsection) by employing grammar decomposition.

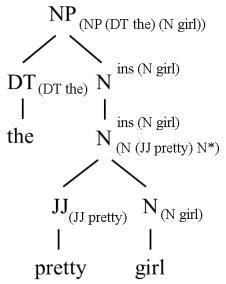

Figure 2: Derivation of Fig. 1b transformed by grammar decomposition.

CFG rule		probability
$\text{NP}_{\text{(NP (DT the) (N girl))}}$	$\to \text{DT}_{\text{(DT the)}}\text{N}^{\text{ins (N girl)}}$	$(1 - a_{\text{DT}}) \times a_{\text{N}}$
$\text{DT}_{\text{(DT the)}}$	$\to$ the	1
$\text{N}^{\text{ins (N girl)}}$	$\to \text{N}^{\text{ins (N girl)}}_{\text{(N (JJ pretty) N*)}}$	$\alpha'_{\text{(N (JJ pretty) N*),N}}$
$\text{N}^{\text{ins (N girl)}}_{\text{(N (JJ pretty) N*)}}$	$\to \text{JJ}_{\text{(JJ pretty)}}\text{N}_{\text{(N girl)}}$	$(1 - a_{\text{JJ}}) \times 1$
$\text{JJ}_{\text{(JJ pretty)}}$	$\to$ pretty	1
$\text{N}_{\text{(N girl)}}$	$\to$ girl	1

Table 1: The rules and probabilities of grammar decomposition for Fig. 2.

Here we provide an example of the derivation shown in Fig. 1b. First, we can transform the derivation in Fig. 1b to another form as shown in Fig. 2. In Fig. 2, all the derivation information is embedded in each symbol. That is, $\text{NP}_{\text{(NP (DT the) (N girl))}}$ is a root symbol of the initial tree "(NP (DT the) (N girl))", which generates two child nodes: $\text{DT}_{\text{(DT the)}}$ and $\text{N}_{\text{(N girl)}}$. $\text{DT}_{\text{(DT the)}}$ generates the terminal node "the". On the other hand, $\text{N}^{\text{ins (N girl)}}$ denotes that $\text{N}_{\text{(N girl)}}$ is inserted by some auxiliary tree, and $\text{N}^{\text{ins (N girl)}}_{\text{(N (JJ pretty) N*)}}$ denotes that the inserted simple auxiliary tree is "(N (JJ pretty) (N*))". The inserted auxiliary tree, "(N (JJ pretty) (N*))", must generate a foot node: "(N girl)" as an immediate child.

Second, we decompose the transformed tree into CFG productions and then assign the probability for each CFG production as shown in Table 1, where a_{DT}, a_{N} and a_{JJ} are insertion probabilities for non-terminal DT, N and JJ, respectively. Note that the probability of a derivation according to Table 1 is the same as the probability of a derivation obtained from the distribution over the initial and auxiliary trees (i.e. eq. 1 and eq. 3).

In Table 1, we assume that the auxiliary tree "(N (JJ pretty) (N*))" is sampled from the first term of eq. 3. When it is sampled from the second term, we alternatively assign the probability $\beta'_{\text{(N (JJ pretty) N*), N}}$.

3.3 Training

We use a blocked Metropolis-Hastings (MH) algorithm (Cohn and Blunsom, 2010) to train our model. The MH algorithm learns BTSG model parameters efficiently, and it can be applied to our insertion model. The MH algorithm consists of the following three steps. For each sentence,

1. Calculate the inside probability (Lari and Young, 1991) in a bottom-up manner using the grammar decomposition.
2. Sample a derivation tree in a top-down manner.
3. Accept or reject the derivation sample by using the MH test.

The MH algorithm is described in detail in (Cohn and Blunsom, 2010). The hyperparameters of our model are updated with the auxiliary variable technique (Teh, 2006a).

4 Experiments

We ran experiments on the British National Corpus (BNC) Treebank [3] and the WSJ English Penn Treebank. We did not use a development set since our model automatically updates the hyperparameters for every iteration. The treebank data was binarized using the *CENTER-HEAD* method (Matsuzaki et al., 2005). We replaced lexical words with counts ≤ 1 in the training set with one of three unknown words using lexical features. We trained our model using a training set, and then sampled 10k derivations for each sentence in a test set. Parsing results were obtained with the MER algorithm (Cohn et al., 2011) using the 10k derivation samples. We show the bracketing F1 score of predicted parse trees evaluated by EVALB[4], averaged over three independent runs.

corpus	method	F1
BNC	CFG	54.08
	BTSG	67.73
	BTSG + insertion	**69.06**
WSJ	CFG	64.99
	BTSG	77.19
	BTSG + insertion	**78.54**
	(Petrov et al., 2006)	77.93[1]
	(Cohn and Blunsom, 2010)	78.40

Table 2: Small dataset experiments

	# rules (# aux. trees)	F1
CFG	35374 (-)	71.0
BTSG	80026 (0)	85.0
BTSG + insertion	65099 (25)	**85.3**
(Post and Gildea, 2009)	-	82.6[2]
(Cohn and Blunsom, 2010)	-	85.3

Table 3: Full Penn Treebank dataset experiments

In small dataset experiments, we used BNC (1k sentences, 90% for training and 10% for testing) and WSJ (section 2 for training and section 22 for testing). This was a small-scale experiment, but large enough to be relevant for low-resource languages. We trained the model with an MH sampler for 1k iterations. Table 2 shows the parsing results for the test set. We compared our model with standard PCFG and BTSG models implemented by us.

Our insertion model successfully outperformed CFG and BTSG. This suggests that adding an insertion operator is helpful for modeling syntax trees accurately. The BTSG model described in (Cohn and Blunsom, 2010) is similar to ours. They reported an F1 score of 78.40 (the score of our BTSG model was 77.19). We speculate that the performance gap is due to data preprocessing such as the treatment of rare words.

[1] Results from (Cohn and Blunsom, 2010).

[2] Results for length ≤ 40.

[3] http://nclt.computing.dcu.ie/~jfoster/resources/

[4] http://nlp.cs.nyu.edu/evalb/

(N̄P (N̄P) (: –))
(N̄P (N̄P) (ADVP (RB respectively)))
(P̄P (P̄P) (, ,))
(V̄P (V̄P) (RB then))
(Q̄P (Q̄P) (IN of))
(SB̄AR (SB̄AR) (RB not))
(S̄ (S̄) (: ;))

Table 4: Examples of lexicalized auxiliary trees obtained from our model in the full treebank dataset. Nonterminal symbols created by binarization are shown with an over-bar.

We also applied our model to the full WSJ Penn Treebank setting (section 2-21 for training and section 23 for testing). The parsing results are shown in Table 3. We trained the model with an MH sampler for 3.5k iterations.

For the full treebank dataset, our model obtained nearly identical results to those obtained with BTSG model, making the grammar size approximately 19% smaller than that of BTSG. We can see that only a small number of auxiliary trees have a great impact on reducing the grammar size. Surprisingly, there are many fewer auxiliary trees than initial trees. We believe this to be due to the tree binarization and our restricted assumption of simple auxiliary trees.

Table 4 shows examples of lexicalized auxiliary trees obtained with our model for the full treebank data. We can see that punctuation ("–", ",", and ";") and adverb (RB) tend to be inserted in other trees. Punctuation and adverb appear in various positions in English sentences. Our results suggest that rather than treat those words as substitutions, it is more reasonable to consider them to be "insertions", which is intuitively understandable.

5 Summary

We proposed a model that incorporates an insertion operator in BTSG and developed an efficient inference technique. Since it is computationally expensive to allow any auxiliary trees, we tackled the problem by introducing a restricted variant of auxiliary trees. Our model outperformed the BTSG model for a small dataset, and achieved comparable parsing results for a large dataset, making the number of grammars much smaller than the BTSG model. We will extend our model to original TAG and evaluate its impact on statistical parsing performance.

References

J. Chen, S. Bangalore, and K. Vijay-Shanker. 2006. Automated extraction of Tree-Adjoining Grammars from treebanks. *Natural Language Engineering*, 12(03):251–299.

D. Chiang, 2003. *Statistical Parsing with an Automatically Extracted Tree Adjoining Grammar*, chapter 16, pages 299–316. CSLI Publications.

T. Cohn and P. Blunsom. 2010. Blocked inference in Bayesian tree substitution grammars. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 225–230, Uppsala, Sweden, July. Association for Computational Linguistics.

T. Cohn, P. Blunsom, and S. Goldwater. 2011. Inducing tree-substitution grammars. *Journal of Machine Learning Research*. To Appear.

M. Johnson and S. Goldwater. 2009. Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 317–325, Boulder, Colorado, June. Association for Computational Linguistics.

A.K. Joshi. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, pages 206–250.

K. Lari and S.J. Young. 1991. Applications of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech & Language*, 5(3):237–257.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL)*, pages 75–82. Association for Computational Linguistics.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (ICCL-ACL)*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

J. Pitman and M. Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, 25(2):855–900.

M. Post and D. 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.

Y. Schabes and R.C. Waters. 1995. Tree insertion grammar: a cubic-time, parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Fuzzy Sets and Systems*, 76(3):309–317.

Y. W. Teh. 2006a. A Bayesian interpretation of interpolated Kneser-Ney. Technical Report TRA2/06, School of Computing, National University of Singapore.

Y. W. Teh. 2006b. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (ICCL-ACL)*, pages 985–992.

F. Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS)*, pages 398–403.