Bayesian Learning of a Tree Substitution Grammar

Matt Post and **Daniel Gildea**
Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

Tree substitution grammars (TSGs) offer many advantages over context-free grammars (CFGs), but are hard to learn. Past approaches have resorted to heuristics. In this paper, we learn a TSG using Gibbs sampling with a nonparametric prior to control subtree size. The learned grammars perform significantly better than heuristically extracted ones on parsing accuracy.

1 Introduction

Tree substition grammars (TSGs) have potential advantages over regular context-free grammars (CFGs), but there is no obvious way to learn these grammars. In particular, learning procedures are not able to take direct advantage of manually annotated corpora like the Penn Treebank, which are not marked for derivations and thus assume a standard CFG. Since different TSG derivations can produce the same parse tree, learning procedures must guess the derivations, the number of which is exponential in the tree size. This compels heuristic methods of subtree extraction, or maximum likelihood estimators which tend to extract large subtrees that overfit the training data.

These problems are common in natural language processing tasks that search for a hidden segmentation. Recently, many groups have had success using Gibbs sampling to address the complexity issue and nonparametric priors to address the overfitting problem (DeNero et al., 2008; Goldwater et al., 2009). In this paper we apply these techniques to learn a tree substitution grammar, evaluate it on the Wall Street Journal parsing task, and compare it to previous work.

2 Model

2.1 Tree substitution grammars

TSGs extend CFGs (and their probabilistic counterparts, which concern us here) by allowing nonterminals to be rewritten as subtrees of arbitrary size. Although nonterminal rewrites are still context-free, in practice TSGs can loosen the independence assumptions of CFGs because larger rules capture more context. This is simpler than the complex independence and backoff decisions of Markovized grammars. Furthermore, subtrees with terminal symbols can be viewed as learning dependencies among the words in the subtree, obviating the need for the manual specification (Magerman, 1995) or automatic inference (Chiang and Bikel, 2002) of lexical dependencies.

Following standard notation for PCFGs, the probability of a derivation d in the grammar is given as

$$\Pr(d) = \prod_{r \in d} \Pr(r)$$

where each r is a rule used in the derivation. Under a regular CFG, each parse tree uniquely idenfifies a derivation. In contrast, multiple derivations in a TSG can produce the same parse; obtaining the parse probability requires a summation over all derivations that could have produced it. This disconnect between parses and derivations complicates both inference and learning. The inference (parsing) task for TSGs is NP-hard (Sima'an, 1996), and in practice the most probable parse is approximated (1) by sampling from the derivation forest or (2) from the top k derivations.

Grammar learning is more difficult as well. CFGs are usually trained on treebanks, especially the Wall Street Journal (WSJ) portion of the Penn Treebank. Once the model is defined, relevant

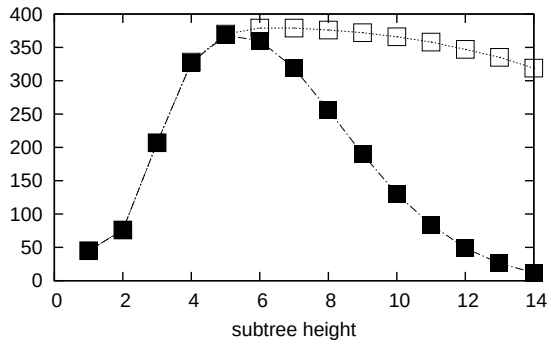

Figure 1: Subtree count (thousands) across heights for the "all subtrees" grammar (□) and the superior "minimal subset" (■) from Bod (2001).

events can simply be counted in the training data. In contrast, there are no treebanks annotated with TSG derivations, and a treebank parse tree of n nodes is ambiguous among 2^n possible derivations. One solution would be to manually annotate a treebank with TSG derivations, but in addition to being expensive, this task requires one to know what the grammar actually is. Part of the thinking motivating TSGs is to let the data determine the best set of subtrees.

One approach to grammar-learning is Data-Oriented Parsing (DOP), whose strategy is to simply take *all* subtrees in the training data as the grammar (Bod, 1993). Bod (2001) did this, approximating "all subtrees" by extracting from the Treebank 400K random subtrees for each subtree height ranging from two to fourteen, and compared the performance of that grammar to that of a heuristically pruned "minimal subset" of it. The latter's performance was quite good, achieving 90.8% F_1 score[1] on section 23 of the WSJ.

This approach is unsatisfying in some ways, however. Instead of heuristic extraction we would prefer a model that explained the subtrees found in the grammar. Furthermore, it seems unlikely that subtrees with ten or so lexical items will be useful on average at test time (Bod did not report how often larger trees are used, but did report that including subtrees with up to twelve lexical items improved parser performance). We expect there to be fewer large subtrees than small ones. Repeating Bod's grammar extraction experiment, this is indeed what we find when comparing these two grammars (Figure 1).

In summary, we would like a principled (model-based) means of determining from the data which set of subtrees should be added to our grammar, and we would like to do so in a manner that prefers smaller subtrees but permits larger ones if the data warrants it. This type of requirement is common in NLP tasks that require searching for a hidden segmentation, and in the following sections we apply it to learning a TSG from the Penn Treebank.

2.2 Collapsed Gibbs sampling with a DP prior[2]

For an excellent introduction to collapsed Gibbs sampling with a DP prior, we refer the reader to Appendix A of Goldwater et al. (2009), which we follow closely here. Our training data is a set of parse trees $\mathcal{T}$ that we assume was produced by an unknown TSG g with probability $\Pr(\mathcal{T}|g)$. Using Bayes' rule, we can compute the probability of a particular hypothesized grammar as

$$\Pr(g \mid \mathcal{T}) = \frac{\Pr(\mathcal{T} \mid g)\Pr(g)}{\Pr(\mathcal{T})}$$

$\Pr(g)$ is a distribution over grammars that expresses our *a priori* preference for g. We use a set of Dirichlet Process (DP) priors (Ferguson, 1973), one for each nonterminal $X \in N$, the set of nonterminals in the grammar. A sample from a DP is a distribution over events in an infinite sample space (in our case, potential subtrees in a TSG) which takes two parameters, a base measure and a concentration parameter:

$$\begin{aligned} g_X &\sim DP(G_X, \alpha) \\ G_X(t) &= \Pr_\$(|t|; p_\$) \prod_{r \in t} \Pr_{\text{MLE}}(r) \end{aligned}$$

The base measure G_X defines the probability of a subtree t as the product of the PCFG rules $r \in t$ that constitute it and a geometric distribution $\Pr_\$$ over the number of those rules, thus encoding a preference for smaller subtrees.[3] The parameter α contributes to the probability that previously unseen subtrees will be sampled. All DPs share parameters $p_\$$ and α. An entire grammar is then given as $g = \{g_X : X \in N\}$. We emphasize that no head information is used by the sampler.

Rather than explicitly consider each segmentation of the parse trees (which would define a TSG and its associated parameters), we use a collapsed Gibbs sampler to integrate over all possi-

[1] The harmonic mean of precision and recall: $F_1 = \frac{2PR}{P+R}$.

[2] Cohn et al. (2009) and O'Donnell et al. (2009) independently developed similar models.

[3] $G_X(t) = 0$ unless root$(t) = X$.

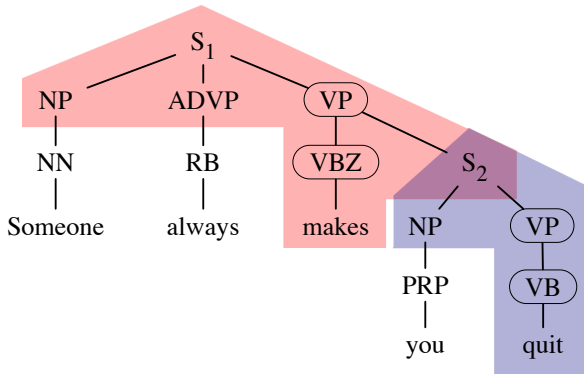

Figure 2: Depiction of $\overline{\text{sub}}(S_2)$ and $\underline{\text{sub}}(S_2)$. Highlighted subtrees correspond with our spinal extraction heuristic (§3). Circles denote nodes whose flag=1.

ble grammars and sample directly from the posterior. This is based on the Chinese Restaurant Process (CRP) representation of the DP. The Gibbs sampler is an iterative procedure. At initialization, each parse tree in the corpus is annotated with a specific derivation by marking each node in the tree with a binary flag. This flag indicates whether the subtree rooted at that node (a height one CFG rule, at minimum) is part of the subtree containing its parent. The Gibbs sampler considers every non-terminal, non-root node c of each parse tree in turn, freezing the rest of the training data and randomly choosing whether to join the subtrees above c and rooted at c (outcome h_1) or to split them (outcome h_2) according to the probability ratio $\phi(h_1)/(\phi(h_1)+\phi(h_2))$, where ϕ assigns a probability to each of the outcomes (Figure 2).

Let $\overline{\text{sub}}(n)$ denote the subtree above and including node n and $\underline{\text{sub}}(n)$ the subtree rooted at n; $\circ$ is a binary operator that forms a single subtree from two adjacent ones. The outcome probabilities are:

$$\begin{aligned}\phi(h_1) &= \theta(t)\\ \phi(h_2) &= \theta(\overline{\text{sub}}(c)) \cdot \theta(\underline{\text{sub}}(c))\end{aligned}$$

where $t = \overline{\text{sub}}(c) \circ \underline{\text{sub}}(c)$. Under the CRP, the subtree probability $\theta(t)$ is a function of the current state of the rest of the training corpus, the appropriate base measure $G_{\text{root}(t)}$, and the concentration parameter α:

$$\theta(t) = \frac{\text{count}_{z_t}(t) + \alpha G_{\text{root}(t)}(t)}{|z_t| + \alpha}$$

where z_t is the multiset of subtrees in the frozen portion of the training corpus sharing the same root as t, and $\text{count}_{z_t}(t)$ is the count of subtree t among them.

3 Experiments

3.1 Setup

We used the standard split for the Wall Street Journal portion of the Treebank, training on sections 2 to 21, and reporting results on sentences with no more than forty words from section 23.

We compare with three other grammars.

- A standard Treebank PCFG.
- A "spinal" TSG, produced by extracting n lexicalized subtrees from each length n sentence in the training data. Each subtree is defined as the sequence of CFG rules from leaf upward all sharing a head, according to the Magerman head-selection rules. We detach the top-level unary rule, and add in counts from the Treebank CFG rules.
- An in-house version of the heuristic "minimal subset" grammar of Bod (2001).[4]

We note two differences in our work that explain the large difference in scores for the minimal grammar from those reported by Bod: (1) we did not implement the smoothed "mismatch parsing", which permits lexical leaves of subtrees to act as wildcards, and (2) we approximate the most probable parse with the top single derivation instead of the top 1,000.

Rule probabilities for all grammars were set with relative frequency. The Gibbs sampler was initialized with the spinal grammar derivations. We construct sampled grammars in two ways: by summing all subtree counts from the derivation states of the first i sampling iterations together with counts from the Treebank CFG rules (denoted $(\alpha, p_\$, \leq i)$), and by taking the counts only from iteration i (denoted $(\alpha, p_\$, i)$).

Our standard CKY parser and Gibbs sampler were both written in Perl. TSG subtrees were flattened to CFG rules and reconstructed afterward, with identical mappings favoring the most probable rule. For pruning, we binned nonterminals according to input span and degree of binarization, keeping the ten highest scoring items in each bin.

3.2 Results

Table 1 contains parser scores. The spinal TSG outperforms a standard unlexicalized PCFG and

[4] All rules of height one, plus 400K subtrees sampled at each height $h, 2 \leq h \leq 14$, minus unlexicalized subtrees of $h > 6$ and lexicalized subtrees with more than twelve words.

grammar	size	LP	LR	F_1
PCFG	46K	75.37	70.05	72.61
spinal	190K	80.30	78.10	**79.18**
minimal subset	2.56M	76.40	78.29	77.33
$(10, 0.7, 100)$	62K	81.48	81.03	81.25
$(10, 0.8, 100)$	61K	81.23	80.79	81.00
$(10, 0.9, 100)$	61K	82.07	81.17	**81.61**
$(100, 0.7, 100)$	64K	81.23	80.98	81.10
$(100, 0.8, 100)$	63K	82.13	81.36	**81.74**
$(100, 0.9, 100)$	62K	82.11	81.20	81.65
$(100, 0.7, \leq 100)$	798K	82.38	82.27	82.32
$(100, 0.8, \leq 100)$	506K	82.27	81.95	82.10
$(100, 0.9, \leq 100)$	290K	82.64	82.09	**82.36**
$(100, 0.7, 500)$	61K	81.95	81.76	81.85
$(100, 0.8, 500)$	60K	82.73	82.21	**82.46**
$(100, 0.9, 500)$	59K	82.57	81.53	82.04
$(100, 0.7, \leq 500)$	2.05M	82.81	82.01	82.40
$(100, 0.8, \leq 500)$	1.13M	83.06	82.10	**82.57**
$(100, 0.9, \leq 500)$	528K	83.17	81.91	82.53

Table 1: Labeled precision, recall, and F_1 on WSJ§23.

the significantly larger "minimal subset" grammar. The sampled grammars outperform all of them. Nearly all of the rules of the best single iteration sampled grammar $(100, 0.8, 500)$ are lexicalized (50,820 of 60,633), and almost half of them have a height greater than one (27,328). Constructing sampled grammars by summing across iterations improved over this in all cases, but at the expense of a much larger grammar.

Figure 3 shows a histogram of subtree size taken from the counts of the subtrees (by token, not type) actually used in parsing WSJ§23. Parsing with the "minimal subset" grammar uses highly lexicalized subtrees, but they do not improve accuracy. We examined sentence-level F_1 scores and found that the use of larger subtrees did correlate with accuracy; however, the low overall accuracy (and the fact that there are so many of these large subtrees available in the grammar) suggests that such rules are overfit. In contrast, the histogram of subtree sizes used in parsing with the sampled grammar matches the shape of the histogram from the grammar itself. Gibbs sampling with a DP prior chooses smaller but more general rules.

4 Summary

Collapsed Gibbs sampling with a DP prior fits nicely with the task of learning a TSG. The sampled grammars are model-based, are simple to specify and extract, and take the expected shape

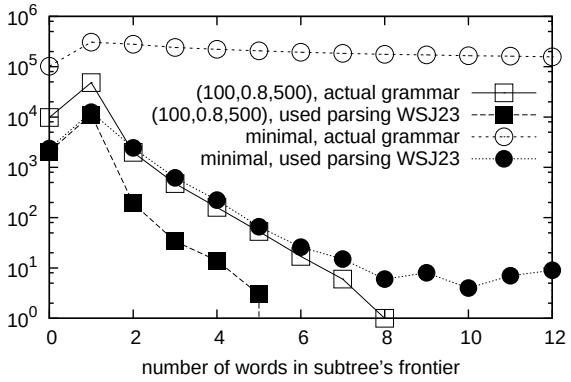

Figure 3: Histogram of subtrees sizes used in parsing WSJ§23 (filled points), as well as from the grammars themselves (outlined points).

over subtree size. They substantially outperform heuristically extracted grammars from previous work as well as our novel spinal grammar, and can do so with many fewer rules.

Acknowledgments This work was supported by NSF grants IIS-0546554 and ITR-0428020.

References

Rens Bod. 1993. Using an annotated corpus as a stochastic grammar. In *Proc. ACL*.

Rens Bod. 2001. What is the minimal set of fragments that achieves maximal parse accuracy. In *Proc. ACL*.

David Chiang and Daniel M. Bikel. 2002. Recovering latent information in treebanks. In *COLING*.

Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *Proc. NAACL*.

John DeNero, Alexandre Bouchard-Côté, and Dan Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *EMNLP*.

Thomas S. Ferguson. 1973. A Bayesian analysis of some nonparametric problems. *Annals of Mathematical Statistics*, 1(2):209–230.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proc. ACL*.

T.J. O'Donnell, N.D. Goodman, J. Snedeker, and J.B. Tenenbaum. 2009. Computation and reuse in language. In *Proc. Cognitive Science Society*.

Khalil Sima'an. 1996. Computational complexity of probabilistic disambiguation by means of tree grammars. In *COLING*.