

Bayesian Learning of a Tree Substitution Grammar

Matt Post

Human Language Technology Center of Excellence
Johns Hopkins University
810 Wyman Park Dr
Baltimore, MD 21211

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 fragment size. The learned grammars perform significantly better than heuristically extracted ones on parsing accuracy. We also investigate an online learning scenario that more closely matches human language learning. **Keywords:** tree substitution grammars; Bayesian learning

Introduction

Tree substitution grammars (TSGs) have potential advantages over standard context-free grammars (CFGs), but there is no obvious way to learn them. 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. Since multiple 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 fragment extraction, or maximum likelihood estimators which tend to extract large fragments 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, Bouchard-Côté, & Klein, 2008; Goldwater, Griffiths, & Johnson, 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.

Tree substitution grammars

TSGs extend standard CFGs (and their probabilistic counterparts, which concern us here) by allowing nonterminals to be rewritten as fragments 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, fragments with terminal symbols can be viewed as learning dependencies among the words in the fragment,

obviating the need for the manual specification (Magerman, 1995) or automatic inference (Chiang & Bikel, 2002) of lexical dependencies.

Following standard notation for PCFGs, the probability of a parse tree t is given as:

$$\Pr(t) = \sum_{d \in \mathcal{D}(t)} \prod_{f \in d} \Pr(f)$$

where d ranges over derivations $\mathcal{D}(t)$ of t , and f over the fragments comprising each derivation. Under a standard CFG (in which all fragments have a depth of one), each parse tree uniquely identifies a derivation, i.e., $|\mathcal{D}(t)| = 1$. In contrast, multiple derivations in a TSG can produce the same parse tree, so 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 is often approximated with sampling techniques or with the Viterbi derivation. 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 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 fragments.

One approach to grammar-learning is Data-Oriented Parsing (DOP), whose strategy is to simply take *all* fragments in the training data as the grammar (Bod, 1993). The set of all fragments can be efficiently represented (Goodman, 1996) but not with arbitrary distributions over the fragments. Bod (2001) approximated all fragments by extracting from the Treebank 400K random fragments for each fragment 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% F₁ score¹ on section 23 of the WSJ.

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

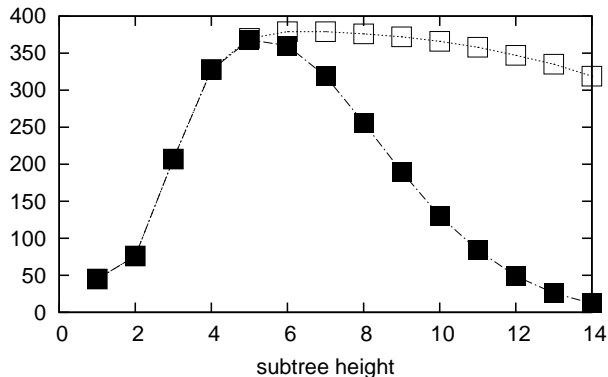


Figure 1: Fragment count (thousands) across heights for the “all fragments” grammar (□) and the superior “minimal subset” (■) from Bod (2001).

This approach is unsatisfying in some ways. Instead of heuristic extraction, we would prefer a model that explained the fragments found in the grammar. Furthermore, it seems unlikely that fragments with ten or so lexical items will be useful on average at test time — we expect there to be fewer large fragments 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 fragments should be added to our grammar, and we would like to do so in a manner that prefers smaller fragments 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.

Collapsed Gibbs sampling with a DP prior²

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:

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

$\Pr(g)$ 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 fragments in a TSG) which takes two parameters, a base measure and a concentration parameter:

$$g_X \sim DP(G_X, \alpha)$$

²Cohn, Goldwater, and Blunsom (2009) and O’Donnell, Goodman, and Tenenbaum (2009) independently proposed similar models.

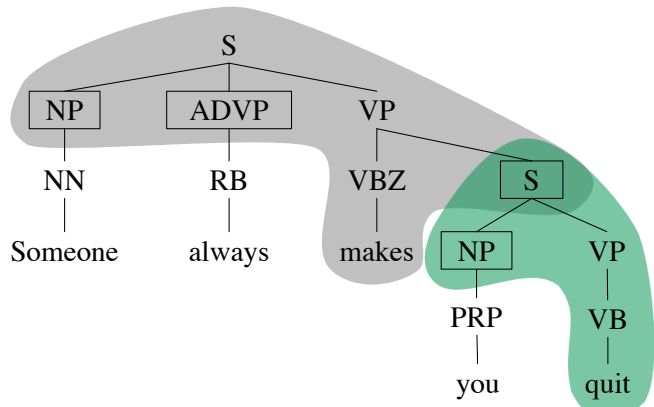


Figure 2: A small parse tree, with boxed nodes depicting fragment roots. The highlighted fragments denote $\nabla(S)$ and $\Delta(S)$ for the interior S node. These fragments also correspond with the spinal extraction heuristic.

$$G_X(f) = \Pr_S(|f|; p_S) \prod_{r \in f} \Pr_{MLE}(r)$$

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

Rather than explicitly consider each derivation of the parse trees (which would define a TSG and its associated parameters), we use a collapsed Gibbs sampler to integrate over all possible grammars and sample directly from the posterior. 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 fragment rooted at that node (a height one CFG rule, at minimum) is part of the fragment 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 fragments 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 $\nabla(n)$ denote the fragment above and including node n and $\Delta(n)$ the fragment rooted at n ; \circ is a binary operator that forms a single fragment from two adjacent ones. The outcome probabilities are:

$$\begin{aligned} \phi(h_1) &= \theta(f) \\ \phi(h_2) &= \theta(\nabla(c)) \cdot \theta(\Delta(c)) \end{aligned}$$

where $f = \nabla(c) \circ \Delta(c)$. Under the CRP, the fragment probability $\theta(f)$ is a function of the current state of the rest of the

³ $G_X(f) = 0$ unless $\text{root}(f) = X$.

training corpus, the appropriate base measure $G_{\text{root}(f)}$, and the concentration parameter α :

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

where z_f is the multiset of fragments in the frozen portion of the training corpus sharing the same root as f , and $\text{count}_{z_f}(f)$ is the count of fragment f among them.

Online Learning

The Bayesian approach that we adopt is attractive as a model for human language learning in that it allows for learning grammars based on general principles for optimal learning from data, without appealing to language-specific apparatus. However, the training procedure that we have described is unrealistic in that it requires the learner to store all the sentences it has seen and iteratively re-analyze them. We have also conducted experiments in a more realistic online setting. Here, the learner is presented with each sentence once, analyzes it into tree fragments, increments the counts for the relevant trees in the grammar, and then moves on to the next sentence. While a human learner would not have access to Penn Treebank parse trees, we make the assumption that the trees are a reasonable proxy for the ability of the learner to interpret the sentence in the real-world context of its use.

In the online training regimen, although early sentences cannot be re-analyzed in light of later data, we would still expect eventual convergence to a similar grammar as more data is presented. An important question for experimentation is the degree to which online learning is competitive with traditional Gibbs sampling for realistic data sizes.

Experiments

We trained our grammars on sections 2 to 21 of the WSJ portion of the Penn Treebank, and report results on sentences with no more than forty words from section 23.

We compare with three other grammars: (1) a standard Treebank PCFG; (2) a ‘‘spinal’’ TSG, produced by extracting n lexicalized fragments from each length n sentence in the training data. Each fragment 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 smooth with counts from the Treebank CFG rules; and (3) an in-house version of the heuristic DOP ‘‘minimal subset’’ grammar of Bod (2001).⁴

The Gibbs samplers were initialized with the spinal grammar derivations. We construct sampled grammars in two ways: (1) by summing fragment counts from the derivation states of the first i sampling iterations (denoted $(\alpha, p_s, \leq i)$), and (2) by taking the counts only from iteration i (denoted (α, p_s, i)). Depth-one CFG counts were added for smoothing, and fragment probabilities were set using relative frequency.

⁴All rules of height one, plus 400K fragments sampled at each height $h, 2 \leq h \leq 14$, minus unlexicalized fragments of $h > 6$ and lexicalized fragments 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 |
| (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,≤100) | 798K | 82.38 | 82.27 | 82.32 |
| (100,0.8,≤100) | 506K | 82.27 | 81.95 | 82.10 |
| (100,0.9,≤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,≤500) | 2.05M | 82.81 | 82.01 | 82.40 |
| (100,0.8,≤500) | 1.13M | 83.06 | 82.10 | 82.57 |
| (100,0.9,≤500) | 528K | 83.17 | 81.91 | 82.53 |

Table 1: Labeled precision, recall, and F₁ on WSJ§23.

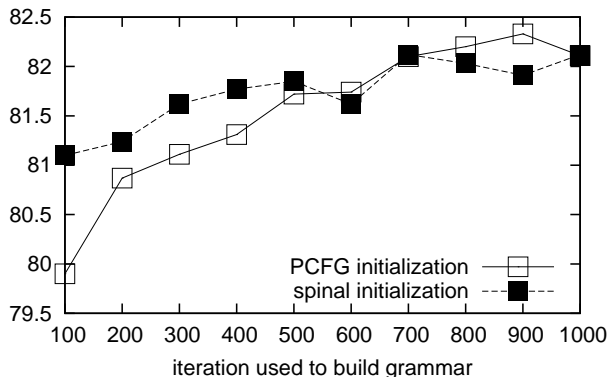


Figure 3: Accuracy for the $(100, 0.7, \cdot)$ grammar family from different initialization states.

Results

Table 1 contains parser scores. The spinal TSG outperforms a standard unlexicalized PCFG and 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 displays parsing accuracy as a function of sampling iteration, from two initialization points. The grammars for this plot were taken from a single iteration. Accuracy seems to converge around the 500th iteration, although there is some evidence that these samplers are still mixing too slowly (Cohn & Blunsom, 2010).

Figure 4 shows a histogram of fragment size taken from the counts of the fragments (by token) actually used in parsing WSJ§23. Parsing with the ‘‘minimal subset’’ grammar uses

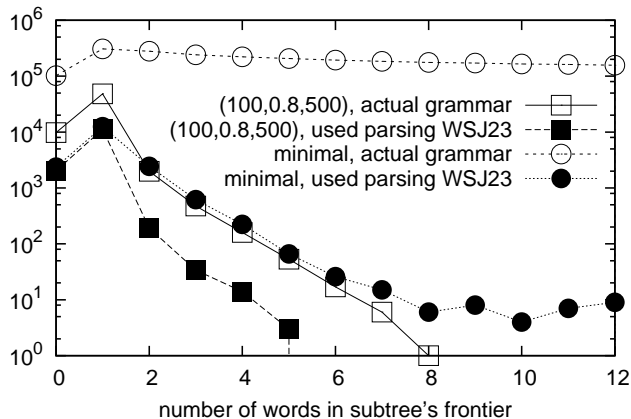


Figure 4: Histogram of fragments sizes used in parsing WSJ23 (filled points), as well as from the grammars themselves (outlined points).

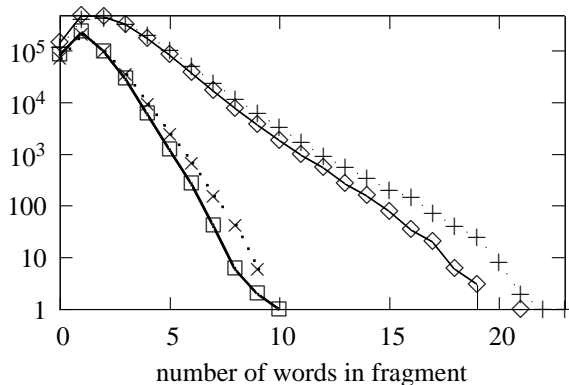


Figure 5: Counts of fragments containing varying numbers of words. The grammars are + (10, 0.7, 500), \diamond (100, 0.7, 500), \times (10, 0.9, 500), and \square (100, 0.9, 500).

highly lexicalized fragments that do not improve accuracy. Its larger fragments do correlate with sentence-level accuracy, however, they are too specific to be of general use. In contrast, the histogram for 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.

As expected, the number of fragment types and their growth rate is directly affected by α , p_S , and the number of iterations the sampler is run. Figure 5 presents lexicalized statistics for four grammars; we plot the number of fragments in that grammar according to their degree of lexicalization, as measured by the number of words among a fragment's leaves. From this, we can see that the stop probability has a more pronounced effect on lexicalization than α .

The DP is not perfect, and still allows some very large fragments to enter and remain in the grammar. As examples, the most lexicalized fragment in the (10, 0.9, 100) grammar had thirty-two terminals (a flat NP with many commas), and the tallest rule had a height of fourteen. These sorts of rules are

unlikely to be useful for inference, and could be discouraged in the model with a more informative base measure.

In our online setting, we find that by sweeping through the sentences of the Treebank only once, and resampling each node in the tree 100 times, we achieve a parser with 76.14 precision and 77.36 recall, when we initialize the sampling of node with all nodes being segmentation points (a PCFG baseline). While this is not competitive with our best results, it shows that significant improvement over the PCFG baseline of 71.72 precision and 72.81 recall is possible with just one pass through the data.

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 over fragment size. They outperform heuristically extracted grammars, and can do so with many fewer fragments.

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

References

- Bod, R. (1993). Using an annotated corpus as a stochastic grammar. In *Proc. ACL*.
- Bod, R. (2001). What is the minimal set of fragments that achieves maximal parse accuracy? In *Proc. ACL*.
- Chiang, D., & Bikel, D. M. (2002). Recovering latent information in treebanks. In *Proc. COLING*.
- Cohn, T., & Blunsom, P. (2010, July). Blocked inference in Bayesian tree substitution grammars. In *Proc. ACL*. Uppsala, Sweden.
- Cohn, T., Goldwater, S., & Blunsom, P. (2009). Inducing compact but accurate tree-substitution grammars. In *Proc. NAACL*.
- DeNero, J., Bouchard-Côté, A., & Klein, D. (2008). Sampling alignment structure under a Bayesian translation model. In *Proc. EMNLP*.
- Ferguson, T. S. (1973). A Bayesian analysis of some non-parametric problems. *Annals of Mathematical Statistics*, 1(2), 209–230.
- Goldwater, S., Griffiths, T. L., & Johnson, M. (2009). A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1), 21–54.
- Goodman, J. (1996, May). Efficient algorithms for parsing the DOP model. In *Proc. EMNLP*. Philadelphia, Pennsylvania, USA.
- Magerman, D. M. (1995). Statistical decision-tree models for parsing. In *Proc. ACL*.
- O'Donnell, T. J., Goodman, N. D., & Tenenbaum, J. B. (2009, March). *Fragment grammars: Exploring computation and reuse in language* (Tech. Rep.). MIT.
- Sima'an, K. (1996). Computational complexity of probabilistic disambiguation by means of tree grammars. In *Proc. COLING*.