Blocked Inference in Bayesian Tree Substitution Grammars

Trevor Cohn
Department of Computer Science
University of Sheffield
T.Cohn@dcs.shef.ac.uk

Phil Blunsom
Computing Laboratory
University of Oxford
Phil.Blunsom@comlab.ox.ac.uk

Abstract

Learning a tree substitution grammar is very challenging due to derivational ambiguity. Our recent approach used a Bayesian non-parametric model to induce good derivations from treebanked input (Cohn et al., 2009), biasing towards small grammars composed of small generalisable productions. In this paper we present a novel training method for the model using a blocked Metropolis-Hastings sampler in place of the previous method's local Gibbs sampler. The blocked sampler makes considerably larger moves than the local sampler and consequently converges in less time. A core component of the algorithm is a grammar transformation which represents an infinite tree substitution grammar in a finite context free grammar. This enables efficient blocked inference for training and also improves the parsing algorithm. Both algorithms are shown to improve parsing accuracy.

1 Introduction

Tree Substitution Grammar (TSG) is a compelling grammar formalism which allows nonterminal rewrites in the form of trees, thereby enabling the modelling of complex linguistic phenomena such as argument frames, lexical agreement and idiomatic phrases. A fundamental problem with TSGs is that they are difficult to estimate, even in the supervised scenario where treebanked data is available. This is because treebanks are typically not annotated with their TSG derivations (how to decompose a tree into elementary tree fragments); instead the derivation needs to be inferred.

In recent work we proposed a TSG model which infers an optimal decomposition under a non-parametric Bayesian prior (Cohn et al., 2009). This used a Gibbs sampler for training, which repeatedly samples for every node in every training tree a binary value indicating whether the node is or is not a substitution point in the tree's derivation. Aggregated over the whole corpus, these values and the underlying trees specify the weighted grammar. Local Gibbs samplers, although conceptually simple, suffer from slow convergence (a.k.a. poor mixing). The sampler can get easily stuck because many locally improbable decisions are required to escape from a locally optimal solution. This problem manifests itself both locally to a sentence and globally over the training sample. The net result is a sampler that is non-convergent, overly dependent on its initialisation and cannot be said to be sampling from the posterior.

In this paper we present a blocked Metropolis-Hasting sampler for learning a TSG, similar to Johnson et al. (2007). The sampler jointly updates all the substitution variables in a tree, making much larger moves than the local single-variable sampler. A critical issue when developing a Metroplis-Hastings sampler is choosing a suitable proposal distribution, which must have the same support as the true distribution. For our model the natural proposal distribution is a MAP point estimate, however this cannot be represented directly as it is infinitely large. To solve this problem we develop a grammar transformation which can succinctly represent an infinite TSG in an equivalent finite Context Free Grammar (CFG). The transformed grammar can be used as a proposal distribution, from which samples can be drawn in polynomial time. Empirically, the blocked sampler converges in fewer iterations and in less time than the local Gibbs sampler. In addition, we also show how the transformed grammar can be used for parsing, which yields theoretical and empirical improvements over our previous method which truncated the grammar.

Proceedings of the ACL 2010 Conference Short Papers, pages 225–230,
Uppsala, Sweden, 11-16 July 2010.

2 Background

A *Tree Substitution Grammar* (TSG; Bod et al. (2003)) is a 4-tuple, $G = (T, N, S, R)$, where T is a set of *terminal symbols*, N is a set of *nonterminal symbols*, $S \in N$ is the distinguished *root nonterminal* and R is a set of productions (rules). The productions take the form of tree fragments, called *elementary trees* (ETs), in which each internal node is labelled with a nonterminal and each leaf is labelled with either a terminal or a nonterminal. The *frontier nonterminal* nodes in each ET form the sites into which other ETs can be substituted. A *derivation* creates a tree by recursive substitution starting with the root symbol and finishing when there are no remaining frontier nonterminals. Figure 1 (left) shows an example derivation where the arrows denote substitution. A *Probabilistic Tree Substitution Grammar* (PTSG) assigns a probability to each rule in the grammar, where each production is assumed to be conditionally independent given its root nonterminal. A derivation's probability is the product of the probabilities of the rules therein.

In this work we employ the same nonparametric TSG model as Cohn et al. (2009), which we now summarise. The inference problem within this model is to identify the posterior distribution of the elementary trees $\mathbf{e}$ given whole trees $\mathbf{t}$. The model is characterised by the use of a Dirichlet Process (DP) prior over the grammar. We define the distribution over elementary trees $\mathbf{e}$ with root nonterminal symbol c as

$$\begin{aligned} G_c|\,\alpha_c, P_0 &\sim \mathrm{DP}(\alpha_c, P_0(\cdot|\,c)) \\ e|\,c \qquad &\sim G_c \end{aligned}$$

where $P_0(\cdot|\,c)$ (the *base distribution*) is a distribution over the infinite space of trees rooted with c, and α_c (the *concentration parameter*) controls the model's tendency towards either reusing elementary trees or creating novel ones as each training instance is encountered.

Rather than representing the distribution G_c explicitly, we integrate over all possible values of G_c. The key result required for inference is that the conditional distribution of e_i, given $\mathbf{e}_{-i}, = e_1 \ldots e_n \backslash e_i$ and the root category c is:

$$p(e_i|\,\mathbf{e}_{-i}, c, \alpha_c, P_0) = \frac{n^{-i}_{e_i,c}}{n^{-i}_{\cdot,c} + \alpha_c} + \frac{\alpha_c P_0(e_i|\,c)}{n^{-i}_{\cdot,c} + \alpha_c} \quad (1)$$

where $n^{-i}_{e_i,c}$ is the number number of times e_i has been used to rewrite c in $\mathbf{e}_{-i}$, and $n^{-i}_{\cdot,c} = \sum_e n^{-i}_{e,c}$ is the total count of rewriting c. Henceforth we omit the $-i$ sub-/super-script for brevity.

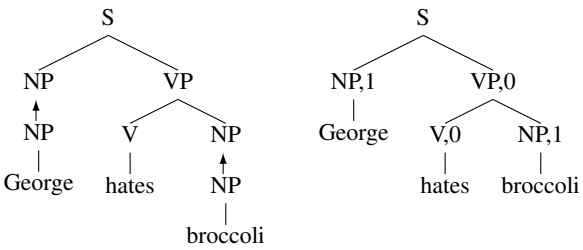

Figure 1: TSG derivation and its corresponding Gibbs state for the local sampler, where each node is marked with a binary variable denoting whether it is a substitution site.

A primary consideration is the definition of P_0. Each e_i can be generated in one of two ways: by drawing from the base distribution, where the probability of any particular tree is proportional to $\alpha_c P_0(e_i|\,c)$, or by drawing from a cache of previous expansions of c, where the probability of any particular expansion is proportional to the number of times that expansion has been used before. In Cohn et al. (2009) we presented base distributions that favour small elementary trees which we expect will generalise well to unseen data. In this work we show that if P_0 is chosen such that it decomposes with the CFG rules contained within each elementary tree,[1] then we can use a novel dynamic programming algorithm to sample derivations without ever enumerating all the elementary trees in the grammar.

The model was trained using a local Gibbs sampler (Geman and Geman, 1984), a Markov chain Monte Carlo (MCMC) method in which random variables are repeatedly sampled conditioned on the values of all other random variables in the model. To formulate the local sampler, we associate a binary variable with each non-root internal node of each tree in the training set, indicating whether that node is a substitution point or not (illustrated in Figure 1). The sampler then visits each node in a random schedule and resamples that node's substitution variable, where the probability of the two different configurations are given by (1). Parsing was performed using a Metropolis-Hastings sampler to draw derivation samples for a string, from which the best tree was recovered. However the sampler used for parsing was biased

[1] Both choices of base distribution in Cohn et al. (2009) decompose into CFG rules. In this paper we focus on the better performing one, P_0^C, which combines a PCFG applied recursively with a stopping probability, s, at each node.

because it used as its proposal distribution a truncated grammar which excluded all but a handful of the unseen elementary trees. Consequently the proposal had smaller support than the true model, voiding the MCMC convergence proofs.

3 Grammar Transformation

We now present a blocked sampler using the Metropolis-Hastings (MH) algorithm to perform sentence-level inference, based on the work of Johnson et al. (2007) who presented a MH sampler for a Bayesian PCFG. This approach repeats the following steps for each sentence in the training set: 1) run the inside algorithm (Lari and Young, 1990) to calculate marginal expansion probabilities under a MAP approximation, 2) sample an analysis top-down and 3) accept or reject using a Metropolis-Hastings (MH) test to correct for differences between the MAP proposal and the true model. Though our model is similar to Johnson et al. (2007)'s, we have an added complication: the MAP grammar cannot be estimated directly. This is a consequence of the base distribution having infinite support (assigning non-zero probability to infinitely many unseen tree fragments), which means the MAP has an infinite rule set. For example, if our base distribution licences the CFG production NP → NP PP then our TSG grammar will contain the infinite set of elementary trees NP → NP PP, NP → (NP NP PP) PP, NP → (NP (NP NP PP) PP) PP, ... with decreasing but non-zero probability.

However, we can represent the infinite MAP using a grammar transformation inspired by Goodman (2003), which represents the MAP TSG in an equivalent finite PCFG.[2] Under the transformed PCFG inference is efficient, allowing its use as the proposal distribution in a blocked MH sampler. We represent the MAP using the grammar transformation in Table 1 which separates the $n_{e,c}$ and P_0 terms in (1) into two separate CFGs, A and B. Grammar A has productions for every ET with $n_{e,c} \geq 1$ which are assigned unsmoothed probabilities: omitting the P_0 term from (1).[3] Grammar B has productions for every CFG production licensed under P_0; its productions are denoted using primed (') nonterminals. The rule $c \rightarrow c'$ bridges from A to B, weighted by the smoothing term excluding P_0, which is computed recursively via child productions. The remaining rules in grammar B correspond to every CFG production in the underlying PCFG base distribution, coupled with the binary decision whether or not nonterminal children should be substitution sites (frontier nonterminals). This choice affects the rule probability by including a s or $1-s$ factor, and child substitution sites also function as a bridge back from grammar B to A. In this way there are often two equivalent paths to reach the same chart cell using the same elementary tree – via grammar A using observed TSG productions and via grammar B using P_0 backoff; summing these yields the desired net probability.

For every ET, e, rewriting c with non-zero count:	
$c \rightarrow \text{sign}(e)$	$\frac{n^-_{e,c}}{n^-_{\cdot,c}+\alpha_c}$
For every internal node e_i in e with children $e_{i,1}, \ldots, e_{i,n}$	
$\text{sign}(e_i) \rightarrow \text{sign}(e_{i,1}) \ldots \text{sign}(e_{i,n})$	1
For every nonterminal, c:	
$c \rightarrow c'$	$\frac{\alpha_c}{n^-_{\cdot,c}+\alpha_c}$
For every pre-terminal CFG production, $c \rightarrow t$:	
$c' \rightarrow t$	$P_{CFG}(c \rightarrow t)$
For every unary CFG production, $c \rightarrow a$:	
$c \rightarrow a$	$P_{CFG}(c \rightarrow a)s_a$
$c' \rightarrow a'$	$P_{CFG}(c \rightarrow a)(1-s_a)$
For every binary CFG production, $c \rightarrow ab$:	
$c' \rightarrow ab$	$P_{CFG}(c \rightarrow ab)s_a s_b$
$c' \rightarrow ab'$	$P_{CFG}(c \rightarrow ab)s_a(1-s_b)$
$c' \rightarrow a'b$	$P_{CFG}(c \rightarrow ab)(1-s_a)s_b$
$c' \rightarrow a'b'$	$P_{CFG}(c \rightarrow ab)(1-s_a)(1-s_b)$

Table 1: Grammar transformation rules to map a MAP TSG into a CFG. Production probabilities are shown to the right of each rule. The sign(e) function creates a unique string signature for an ET e (where the signature of a frontier node is itself) and s_c is the Bernoulli probability of c being a substitution variable (and stopping the P_0 recursion).

Figure 2 shows an example of the transformation of an elementary tree with non-zero count, $n_{e,c} \geq 1$, into the two types of CFG rules. Both parts are capable of parsing the string NP, saw, NP into a S, as illustrated in Figure 3; summing the probability of both analyses gives the model probability from (1). Note that although the probabilities exactly match the true model for a single elementary tree, the probability of derivations composed of many elementary trees may not match because the model's caching behaviour has been suppressed, i.e., the counts, n, are not incremented during the course of a derivation.

For training we define the MH sampler as follows. First we estimate the MAP grammar over

[2] Backoff DOP uses a similar packed representation to encode the set of smaller subtrees for a given elementary tree (Sima'an and Buratto, 2003), which are used to smooth its probability estimate.

[3] The transform assumes inside inference. For Viterbi replace the probability for $c \rightarrow \text{sign}(e)$ with $\frac{n^-_{e,c}+\alpha_c P_0(e|\,c)}{n^-_{\cdot,c}+\alpha_c}$.

S → NP VP{V{saw},NP}	$\frac{n^-_{e,S}}{n^-_{\cdot,S}+\alpha_S}$
VP{V{saw},NP} → V{saw} NP	1
V{saw} → saw	1
S → S'	$\frac{\alpha_S}{n^-_{\cdot,S}+\alpha_S}$
S' → NP VP'	$P_{CFG}(\text{S} \rightarrow \text{NP VP}) s_{NP}(1-s_{VP})$
VP' → V' NP	$P_{CFG}(\text{VP} \rightarrow \text{V NP})(1-s_V) s_{NP}$
V' → saw	$P_{CFG}(\text{V} \rightarrow \text{saw})$

Figure 2: Example of the transformed grammar for the ET (S NP (VP (V saw) NP)). Taking the product of the rule scores above the line yields the left term in (1), and the product of the scores below the line yields the right term.

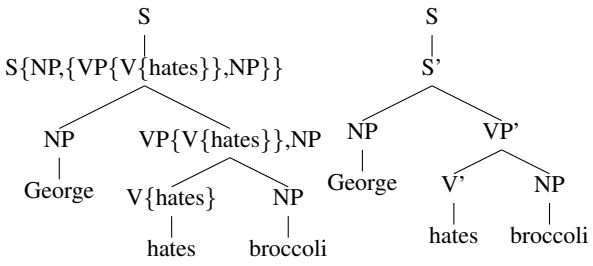

Figure 3: Example trees under the grammar transform, which both encode the same TSG derivation from Figure 1. The left tree encodes that the S → NP (VP (V hates) NP elementary tree was drawn from the cache, while for the right tree this same elementary tree was drawn from the base distribution (the left and right terms in (1), respectively).

the derivations of training corpus excluding the current tree, which we represent using the PCFG transformation. The next step is to sample derivations for a given tree, for which we use a constrained variant of the inside algorithm (Lari and Young, 1990). We must ensure that the TSG derivation produces the given tree, and therefore during inside inference we only consider spans that are constituents in the tree and are labelled with the correct nonterminal. Nonterminals are said to match their primed and signed counterparts, e.g., NP′ and NP{DT,NN{car}} both match NP. Under the tree constraints the time complexity of inside inference is linear in the length of the sentence. A derivation is then sampled from the inside chart using a top-down traversal (Johnson et al., 2007), and converted back into its equivalent TSG derivation. The derivation is scored with the true model and accepted or rejected using the MH test; accepted samples then replace the current derivation for the tree, and rejected samples leave the previous derivation unchanged. These steps are then repeated for another tree in the training set, and the process is then repeated over the full training set many times.

Parsing The grammar transform is not only useful for training, but also for parsing. To parse a sentence we sample a number of TSG derivations from the MAP which are then accepted or rejected into the full model using a MH step. The samples are obtained from the same transformed grammar but adapting the algorithm for an unsupervised setting where parse trees are not available. For this we use the standard inside algorithm applied to the sentence, omitting the tree constraints, which has time complexity cubic in the length of the sentence. We then sample a derivation from the inside chart and perform the MH acceptance test. This setup is theoretically more appealing than our previous approach in which we truncated the approximation grammar to exclude most of the zero count rules (Cohn et al., 2009). We found that both the maximum probability derivation and tree were considerably worse than a tree constructed to maximise the expected number of correct CFG rules (MER), based on Goodman's (2003) algorithm for maximising labelled recall. For this reason we the MER parsing algorithm using sampled Monte Carlo estimates for the marginals over CFG rules at each sentence span.

4 Experiments

We tested our model on the Penn treebank using the same data setup as Cohn et al. (2009). Specifically, we used only section 2 for training and section 22 (devel) for reporting results. Our models were all sampled for 5k iterations with hyperparameter inference for α_c and s_c $\forall$ $c \in N$, but in contrast to our previous approach we did not use annealing which we did not find to help generalisation accuracy. The MH acceptance rates were in excess of 99% across both training and parsing. All results are averages over three runs.

For training the blocked MH sampler exhibits faster convergence than the local Gibbs sampler, as shown in Figure 4. Irrespective of the initialisation the blocked sampler finds higher likelihood states in many fewer iterations (the same trend continues until iteration 5k). To be fair, the blocked sampler is slower per iteration (roughly 50% worse) due to the higher overheads of the grammar transform and performing dynamic programming (despite nominal optimisation).[4] Even after accounting for the time differ-

[4]The speed difference diminishes with corpus size: on sections 2–22 the blocked sampler is only 19% slower per

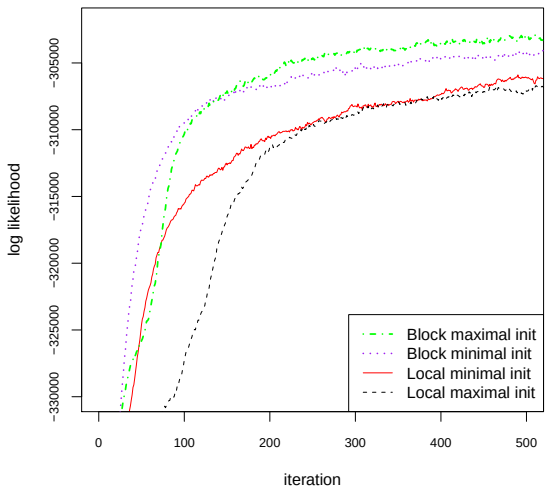

Figure 4: Training likelihood vs. iteration. Each sampling method was initialised with both minimal and maximal elementary trees.

Training	truncated	transform
Local minimal init	77.63	77.98
Local maximal init	77.19	77.71
Blocked minimal init	77.98	78.40
Blocked maximal init	77.67	78.24

Table 2: Development F1 scores using the truncated parsing algorithm and the novel grammar transform algorithm for four different training configurations.

ence the blocked sampler is more effective than the local Gibbs sampler. Training likelihood is highly correlated with generalisation F1 (Pearson's correlation efficient of 0.95), and therefore improving the sampler convergence will have immediate effects on performance.

Parsing results are shown in Table 2.[5] The blocked sampler results in better generalisation F1 scores than the local Gibbs sampler, irrespective of the initialisation condition or parsing method used. The use of the grammar transform in parsing also yields better scores irrespective of the underlying model. Together these results strongly advocate the use of the grammar transform for inference in infinite TSGs.

We also trained the model on the standard Penn treebank training set (sections 2–21). We initialised the model with the final sample from a run on the small training set, and used the blocked sampler for 6500 iterations. Averaged over three runs, the test F1 (section 23) was 85.3 an improvement over our earlier 84.0 (Cohn et al., 2009) although still well below state-of-the-art parsers. We conjecture that the performance gap is due to the model using an overly simplistic treatment of unknown words, and also a further mixing problems with the sampler. For the full data set the counts are much larger in magnitude which leads to stronger modes. The sampler has difficulty escaping such modes and therefore is slower to mix. One way to solve the mixing problem is for the sampler to make more global moves, e.g., with table label resampling (Johnson and Goldwater, 2009) or split-merge (Jain and Neal, 2000). Another way is to use a variational approximation instead of MCMC sampling (Wainwright and Jordan, 2008).

iteration than the local sampler.

[5]Our baseline 'Local maximal init' slightly exceeds previously reported score of 76.89% (Cohn et al., 2009).

5 Discussion

We have demonstrated how our grammar transformation can implicitly represent an exponential space of tree fragments efficiently, allowing us to build a sampler with considerably better mixing properties than a local Gibbs sampler. The same technique was also shown to improve the parsing algorithm. These improvements are in no way limited to our particular choice of a TSG parsing model, many hierarchical Bayesian models have been proposed which would also permit similar optimised samplers. In particular models which induce segmentations of complex structures stand to benefit from this work; Examples include the word segmentation model of Goldwater et al. (2006) for which it would be trivial to adapt our technique to develop a blocked sampler. Hierarchical Bayesian segmentation models have also become popular in statistical machine translation where there is a need to learn phrasal translation structures that can be decomposed at the word level (DeNero et al., 2008; Blunsom et al., 2009; Cohn and Blunsom, 2009). We envisage similar representations being applied to these models to improve their mixing properties.

A particularly interesting avenue for further research is to employ our blocked sampler for unsupervised grammar induction. While it is difficult to extend the local Gibbs sampler to the case where the tree is not observed, the dynamic program for our blocked sampler can be easily used for unsupervised inference by omitting the tree matching constraints.

References

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A Gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 782–790, Suntec, Singapore, August.

Rens Bod, Remko Scha, and Khalil Sima'an, editors. 2003. *Data-oriented parsing*. Center for the Study of Language and Information - Studies in Computational Linguistics. University of Chicago Press.

Trevor Cohn and Phil Blunsom. 2009. A Bayesian model of syntax-directed tree to string grammar induction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 352–361, Singapore, August.

Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing compact but accurate tree-substitution 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 548–556, Boulder, Colorado, June.

John DeNero, Alexandre Bouchard-Côté, and Dan Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 314–323, Honolulu, Hawaii, October.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 673–680, Sydney, Australia, July.

Joshua Goodman. 2003. Efficient parsing of DOP with PCFG-reductions. In Bod et al. (Bod et al., 2003), chapter 8.

Sonia Jain and Radford M. Neal. 2000. A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13:158–182.

Mark Johnson and Sharon 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*, pages 317–325, Boulder, Colorado, June.

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

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

Khalil Sima'an and Luciano Buratto. 2003. Backoff parameter estimation for the dop model. In Nada Lavrac, Dragan Gamberger, Ljupco Todorovski, and Hendrik Blockeel, editors, *ECML*, volume 2837 of *Lecture Notes in Computer Science*, pages 373–384. Springer.

Martin J Wainwright and Michael I Jordan. 2008. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc., Hanover, MA, USA.