Inducing Compact but Accurate Tree-Substitution Grammars

Trevor Cohn and **Sharon Goldwater** and **Phil Blunsom**
School of Informatics
University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
Scotland, United Kingdom
{tcohn,sgwater,pblunsom}@inf.ed.ac.uk

Abstract

Tree substitution grammars (TSGs) are a compelling alternative to context-free grammars for modelling syntax. However, many popular techniques for estimating weighted TSGs (under the moniker of Data Oriented Parsing) suffer from the problems of inconsistency and overfitting. We present a theoretically principled model which solves these problems using a Bayesian non-parametric formulation. Our model learns compact and simple grammars, uncovering latent linguistic structures (e.g., verb subcategorisation), and in doing so far out-performs a standard PCFG.

1 Introduction

Many successful models of syntax are based on *Probabilistic Context Free Grammars* (PCFGs) (e.g., Collins (1999), Charniak (2000)). However, directly learning a PCFG from a treebank results in poor parsing performance, due largely to the unrealistic independence assumptions imposed by the context-free assumption. Considerable effort is required to coax good results from a PCFG, in the form of grammar engineering, feature selection and clever smoothing (Collins, 1999; Charniak, 2000; Charniak and Johnson, 2005; Johnson, 1998). This effort must be repeated when moving to different languages, grammar formalisms or treebanks. We propose that much of this hand-coded knowledge can be obtained automatically as an emergent property of the treebanked data, thereby reducing the need for human input in crafting the grammar.

We present a model for automatically learning a *Probabilistic Tree Substitution Grammar* (PTSG), an extension to the PCFG in which non-terminals can rewrite as entire tree fragments (*elementary trees*), not just immediate children. These large fragments can be used to encode non-local context, such as head-lexicalisation and verb subcategorisation. Since no annotated data is available providing TSG derivations of strings, we must induce the PTSG productions and their probabilities in an unsupervised way from an ordinary treebank. This is the same problem addressed by Data Oriented Parsing (DOP, Bod et al. (2003)), a method which uses as productions all sub-trees of the training corpus. However, many of the DOP estimation methods have serious shortcomings (Johnson, 2002), namely inconsistency for DOP1 (Bod, 2003) and overfitting of the maximum likelihood estimate (Prescher et al., 2004).

In this paper we develop an alternative means of learning a PTSG from a treebanked corpus, with the twin objectives of a) finding a grammar which accurately models the data and b) keeping the grammar as simple as possible, with few, compact, elementary trees. This is achieved using a prior to encourage sparsity and simplicity in a Bayesian non-parametric formulation. The framework allows us to perform inference over an infinite space of grammar productions in an elegant and efficient manner. The net result is a grammar which only uses the increased context afforded by the TSG when necessary to model the data, and otherwise uses context-free rules.[1] That is, our model learns to use larger rules when the CFG's independence assumptions do not hold. This contrasts with DOP, which seeks to use all elementary trees from the training set. While our model is able, in theory, to use all such trees, in practice the data does not justify such a large grammar. Grammars that are only about twice the size of a treebank PCFG provide large gains in accuracy. We obtain additional improvements with

[1]While TSGs and CFGs describe the same string languages, TSGs can describe context-sensitive tree-languages, which CFGs cannot.

grammars that are somewhat larger, but still much smaller than the DOP all-subtrees grammar. The rules in these grammars are intuitive, potentially offering insights into grammatical structure which could be used in, e.g., the development of syntactic ontologies and guidelines for future treebanking projects.

2 Background and related work

A *Tree Substitution Grammar*[2] (TSG) is a 4-tuple, $G = (T, N, S, R)$, where T is a set of *terminal symbols*, N is a set of *non-terminal symbols*, $S \in N$ is the distinguished *root non-terminal* and R is a set of productions (a.k.a. rules). The productions take the form of *elementary trees* – tree fragments of depth ≥ 2,[3] where each internal node is labelled with a non-terminal and each leaf is labelled with either a terminal or a non-terminal. Non-terminal leaves are called *frontier non-terminals* and form the substitution sites in the generative process of creating trees with the grammar.

A *derivation* creates a tree by starting with the root symbol and rewriting (substituting) it with an elementary tree, then continuing to rewrite frontier non-terminals with elementary trees until there are no remaining frontier non-terminals. Unlike Context Free Grammars (CFGs) a syntax tree may not uniquely specify the derivation, as illustrated in Figure 1 which shows two derivations using different elementary trees to produce the same tree.

A *Probabilistic Tree Substitution Grammar* (PTSG), like a PCFG, assigns a probability to each rule in the grammar. The probability of a derivation is the product of the probabilities of its component rules, and the probability of a tree is the sum of the probabilities of its derivations.

As we mentioned in the introduction, work within the DOP framework seeks to induce PTSGs from treebanks by using all possible subtrees as rules, and one of a variety of methods for estimating rule probabilities.[4] Our aim of inducing compact grammars contrasts with that of DOP; moreover, we develop a probabilistic estimator which avoids the shortcomings of DOP1 and the maximum likelihood estimate (Bod, 2000; Bod, 2003; Johnson, 2002). Recent work on DOP estimation also seeks to address these problems, drawing from estimation theory to solve the consistency problem (Prescher et al., 2004; Zollmann and Sima'an, 2005), or incorporating a grammar brevity term into the learning objective (Zuidema, 2007). Our work differs from these previous approaches in that we explicitly model a prior over grammars within a Bayesian framework.[5]

Models of grammar refinement (Petrov et al., 2006; Liang et al., 2007; Finkel et al., 2007). also aim to automatically learn latent structure underlying treebanked data. These models allow each non-terminal to be split into a number of subcategories. Theoretically the grammar space of our model is a sub-space of theirs (projecting the TSG's elementary trees into CFG rules). However, the number of non-terminals required to recreate our TSG grammars in a PCFG would be exorbitant. Consequently, our model should be better able to learn specific lexical patterns, such as full noun-phrases and verbs with their sub-categorisation frames, while theirs are better suited to learning subcategories with larger membership, such as the terminals for days of the week and noun-adjective agreement. The approaches are orthogonal, and we expect that combining a category refinement model with our TSG model would provide better performance than either approach alone.

Our model is similar to the Adaptor Grammar model of Johnson et al. (2007a), which is also a kind of tree-substitution grammar based on non-parametric Bayesian techniques. However, Adaptor Grammars require that each sub-tree expands completely, with only terminal symbols as leaves, while our own model permits non-terminal frontier nodes. In addition, they disallow recursive containment of adapted non-terminals; we impose no such constraint. These constraints on Adaptor Grammars are a result of their application for inducing grammars from strings, rather than from trees, as in this work.

3 Model

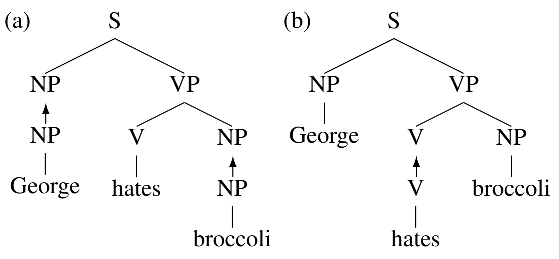
Recall the nature of our task: we are given a corpus of parse trees $\mathbf{t}$ and wish to infer a tree-substitution grammar G that we can use to parse new data. Rather than inferring a grammar directly, we go

[2]A TSG is a *Tree Adjoining Grammar* (TAG; Joshi (2003)) without the adjunction operator.

[3]Elementary trees of depth two correspond to productions in a context free grammar.

[4]TAG induction (Chiang and Bikel, 2002; Xia, 2002) also tackles a similar learning problem.

[5]A similar Bayesian model of TSG induction has been developed independently to this work (O'Donnell et al., 2009b; O'Donnell et al., 2009a).



$$
\begin{aligned}
\text{S} &\rightarrow \text{NP (VP (V hates) NP)}\\
\text{NP} &\rightarrow \text{George}\\
\text{NP} &\rightarrow \text{broccoli}\\
\text{S} &\rightarrow \text{(NP George) (VP V (NP broccoli))}\\
\text{V} &\rightarrow \text{hates}
\end{aligned}
$$

Figure 1: Example derivations for the same tree, where arrows indicate substitution sites. The elementary trees used in (a) and (b) are shown below as grammar productions in bracketed tree notation.

through an intermediate step of inferring a distribution over the derivations used to produce $\mathbf{t}$, i.e., a distribution over sequences of elementary trees $\mathbf{e}$ that compose to form $\mathbf{t}$. We will then essentially read the grammar off the elementary trees, as described in Section 5. Our problem therefore becomes one of identifying the posterior distribution of $\mathbf{e}$ given $\mathbf{t}$, which we can do using Bayes' Rule:

$$P(\mathbf{e}|\mathbf{t}) \propto P(\mathbf{t}|\mathbf{e})P(\mathbf{e}) \qquad (1)$$

Since the sequence of elementary trees can be split into derivations, each of which completely specifies a tree, $P(\mathbf{t}|\mathbf{e})$ is either equal to 1 (when $\mathbf{t}$ and $\mathbf{e}$ are consistent) or 0 (otherwise). Therefore, the work in our model is done by the prior distribution over elementary trees. Note that this is analogous to the Bayesian model of word segmentation presented by Goldwater et al. (2006); indeed, the problem of inferring $\mathbf{e}$ from $\mathbf{t}$ can be viewed as a segmentation problem, where each full tree must be segmented into one or more elementary trees. As in Goldwater et al. (2006), we wish to favour solutions employing a relatively small number of elementary units (here, elementary trees). This can be done using a Dirichlet process (DP) prior. Specifically, we define the distribution of elementary tree e with root non-terminal symbol c as

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

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 (and consequently, the tendency to infer larger or smaller sets of elementary trees from the observed data). We discuss the base distribution in more detail below.

Rather than representing the distribution G_c explicitly, we integrate over all possible values of G_c. This leads to the following distribution over e_i, conditioned on $\mathbf{e}_{<i} = e_1 \ldots e_{i-1}$ and the root category c:

$$p(e_i|\mathbf{e}_{<i}, c, \alpha_c, P_0) = \frac{n^{<i}_{e_i,c} + \alpha_c P_0(e_i|c)}{n^{<i}_{\cdot,c} + \alpha_c} \qquad (2)$$

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.

As with other DP models, ours can be viewed as a *cache model*, where e_i can be generated in one of two ways: by drawing it from the base distribution, where the probability of any particular tree is proportional to $\alpha_c P_0(e_i|c)$, or by drawing it 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. This view makes it clear that the model embodies a "rich-get-richer" dynamic in which a few expansions will occur with high probability, but many will occur only once or twice, as is typical of natural language. Our model is similar in this way to the Adaptor Grammar model of Johnson et al. (2007b).

We still need to define P_0, the base distribution over tree fragments. We use two such distributions. The first, P_0^M generates each elementary tree by a series of random decisions: whether to expand a non-terminal, how many children to produce and their identities. The probability of expanding a non-terminal node labelled c is parameterised via a binomial distribution, $\text{Bin}(\beta_c)$, while all other decisions are chosen uniformly at random. The second base distribution, P_0^C, has a similar generative process but draws non-terminal expansions from a treebank-trained PCFG instead of a uniform distribution.

Both choices of P_0 have the net effect of biasing the model towards simple rules, comprised of a small number of internal nodes. The geometric increase in cost discourages the model from using larger rules; for this to occur these rules must yield

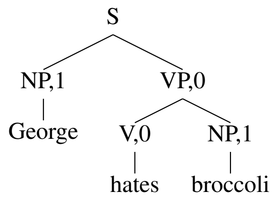


Figure 2: Gibbs state e specifying the derivation in Figure 1a. Each node is labelled with its substitution indicator variable.

a large increase in the data likelihood. As P_0^C incorporates PCFG probabilities, it assigns higher relative probability to larger rules, compared to the more draconian P_0^M.

4 Training

To train our model we use Gibbs sampling (Geman and Geman, 1984), a Markov chain Monte Carlo method (Gilks et al., 1996) in which variables are repeatedly sampled conditioned on the values of all other variables in the model. After a period of burn-in, each sampler state (set of variable assignments) is a sample from the posterior distribution of the model. In our case, we wish to sample from $P(\mathbf{e}|\mathbf{t}, \alpha, \beta)$, where $(\alpha, \beta) = \{\alpha_c, \beta_c\}$ for all categories c. To do so, 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. Each substitution point forms the root of some elementary tree, as well as a frontier non-terminal of an ancestor node's elementary tree. Collectively, the training trees and substitution variables specify the sequence of elementary trees $\mathbf{e}$ that is the current state of the sampler. Figure 2 shows an example tree with its substitution variables, corresponding to the TSG derivation in Figure 1a.

Our Gibbs sampler works by sampling the value of each substitution variable, one at a time, in random order. If d is the node associated with the substitution variable s under consideration, then the two possible values of s define two options for $\mathbf{e}$: one in which d is internal to some elementary tree e_M, and one in which d is the substitution site connecting two smaller trees, e_A and e_B. In the example in Figure 2, when sampling the VP node, e_M = (S NP (VP (V hates) NP)), e_A = (S NP VP), and e_B = (VP (V hates) NP). To sample a value for s, we compute the probabilities of e_M and (e_A, e_B), conditioned on $\mathbf{e}^-$: all other elementary trees in the training set that share at most a root or frontier non-terminal with e_M, e_A, or e_B. This is easy to do because the DP is *exchangeable*, meaning that the probability of a set of outcomes does not depend on their ordering. Therefore, we can treat the elementary trees under consideration as the last ones to be sampled, and apply Equation 2, giving us

$$P(e_M|c_M) = \frac{n^-_{e_M,c_M} + \alpha_{c_M} P_0(e_M|c_M)}{n^-_{\cdot,c_M} + \alpha_{c_M}} \quad (3)$$

$$P(e_A, e_B|c_A) = \frac{n^-_{e_A,c_A} + \alpha_{c_A} P_0(e_A|c_A)}{n^-_{\cdot,c_A} + \alpha_{c_A}} \quad (4)$$
$$\times \frac{n^-_{e_B,c_B} + \delta(e_A, e_B) + \alpha_{c_B} P_0(e_B|c_B)}{n^-_{\cdot,c_B} + \delta(c_A, c_B) + \alpha_{c_B}}$$

where c_x is the root label of $e_x, x \in \{A, B, M\}$, the counts n^- are with respect to $\mathbf{e}^-$, and $\delta(\cdot,\cdot)$ is the Kronecker delta function, which returns 1 when its arguments are identical and 0 otherwise. We have omitted $\mathbf{e}^-, \mathbf{t}, \alpha$ and β from the conditioning context. The δ terms in the second factor of (4) account the changes to n^- that would occur after observing e_A, which forms part of the conditioning context for e_B. If the trees e_A and e_B are identical, then the count $n^-_{e_B,c_B}$ would increase by one, and if the trees share the same root non-terminal, then $n^-_{\cdot,c_B}$ would increase by one.

In the previous discussion, we have assumed that the model hyperparameters, (α, β), are known. However, selecting their values by hand is extremely difficult and fitting their values on heldout data is often very time consuming. For this reason we treat the hyper-parameters as variables in our model and infer their values during training. We choose vague priors for each hyper-parameter, encoding our lack of information about their values. We treat the concentration parameters, α, as being generated by a vague gamma prior, $\alpha_c \sim \text{Gamma}(0.001, 1000)$. We sample a new value α'_c using a log-normal distribution with mean α_c and variance 0.3, which is then accepted into the distribution $p(\alpha_c|\mathbf{e}, \mathbf{t}, \alpha^-, \beta)$ using the Metropolis-Hastings algorithm. We use a Beta prior for the binomial specification parameters, $\beta_c \sim \text{Beta}(1, 1)$. As the Beta distribution is conjugate to the binomial, we can directly resample the β parameters from the posterior, $p(\beta_c|\mathbf{e}, \mathbf{t}, \alpha, \beta^-)$. Both the concentration and substitution parameters are resampled after every full Gibbs sampling iteration over the training trees.

5 Parsing

We now turn to the problem of using the model to parse novel sentences. This requires finding the maximiser of

$$p(t|w, \mathbf{t}) = \int p(t|w, \mathbf{e}, \alpha, \beta)\, p(\mathbf{e}, \alpha, \beta|\mathbf{t})\; d\mathbf{e}\, d\alpha\, d\beta \tag{5}$$

where w is the sequence of words being parsed and t the resulting tree, $\mathbf{t}$ are the training trees and $\mathbf{e}$ their segmentation into elementary trees.

Unfortunately solving for the maximising parse tree in (5) is intractable. However, it can approximated using Monte Carlo techniques. Given a sample of $(\mathbf{e}, \alpha, \beta)$[6] we can reason over the space of possible trees using a Metropolis-Hastings sampler (Johnson et al., 2007b) coupled with a Monte Carlo integral (Bod, 2003). The first step is to sample from the posterior over derivations, $p(d|w, \mathbf{e}, \alpha, \beta)$. This is achieved by drawing samples from an approximation grammar, $\tilde{p}(d|w)$, which are then accepted to the true distribution using the Metropolis-Hastings algorithm. The second step records for each sampled derivation the CFG tree. The counts of such trees constitute an approximation to $p(t|w, \mathbf{e}, \alpha, \beta)$, from which we can recover the maximum probability tree.

A natural proposal distribution, $\tilde{p}(d|w)$, is the maximum a posterior (MAP) grammar given the elementary tree analysis of our training set (analgous to the PCFG approximation used in Johnson et al. (2007b)). This is not practical because the approximation grammar is infinite: elementary trees with zero count in $\mathbf{e}$ still have some residual probability under P_0. In the absence of a better alternative, we discard (most of) the zero-count rules from MAP grammar. This results in a tractable grammar representing the majority of the probability mass, from which we can sample derivations. We specifically retain all zero-count PCFG productions observed in the training set in order to provide greater robustness on unseen data.[7]

In addition to finding the maximum probability parse (MPP), we also report results using the maximum probability derivation (MPD). While this could be calculated in the same manner as described above, we found that using the CYK algorithm (Cocke, 1969) to find the Viterbi derivation for $\tilde{p}$ yielded consistently better results. This algorithm maximises an approximated model, as opposed to approximately optimising the true model. We also present results using the tree with the maximum expected count of CFG rules (MER). This uses counts of the CFG rules applied at each span (compiled from the derivation samples) followed by a maximisation step to find the best tree. This is similar to the MAX-RULE-SUM algorithm of Petrov and Klein (2007) and maximum expected recall parsing (Goodman, 2003).

$$\begin{aligned}
&S \rightarrow A \mid B\\
&A \rightarrow A\ A \mid B\ B \mid (A\ a)\ (A\ a) \mid (B\ a)\ (B\ a)\\
&B \rightarrow A\ A \mid B\ B \mid (A\ b)\ (A\ b) \mid (B\ b)\ (B\ b)
\end{aligned}$$

Figure 3: TSG used to generate synthetic data. All production probabilities are uniform.

6 Experiments

Synthetic data Before applying the model to natural language, we first create a synthetic problem to confirm that the model is capable of recovering a known tree-substitution grammar. We created 50 random trees from the TSG shown in Figure 3. This produces binary trees with A and B internal nodes and 'a' and 'b' as terminals, such that the terminals correspond to their grand-parent non-terminal (A and a or B and b). These trees cannot be modelled accurately with a CFG because expanding A and B nodes into terminal strings requires knowing their parent's non-terminal.

We train the model for 100 iterations of Gibbs sampling using annealing to speed convergence. Annealing amounts to smoothing the distributions in (3) and (4) by raising them to the power of $\frac{1}{T}$. Our annealing schedule begins at $T = 3$ and linearly decreases to reach $T = 1$ in the final iteration. The sampler converges to the correct grammar, with the 10 rules from Figure 3.

Penn-treebank parsing We ran our natural language experiments on the Penn treebank, using the standard data splits (sections 2–21 for training, 22 for development and 23 for testing). As our model is parameter free (the α and β parame-

[6]Using many samples of $(\mathbf{e}, \alpha, \beta)$ in a Monte Carlo integral is a straight-forward extension to our parsing algorithm. We did not observe a significant improvement in parsing accuracy when using a multiple samples compared to a single sample, and therefore just presnt results for a single sample.

[7]Experiments with additional zero-count rules (e.g., DOP fragments up to a certain depth) lead to an indistinguishable change in the parsing accuracy, but at the cost of a much larger approximation grammar.

ters are learnt in training), we do not use the development set for parameter tuning. We expect that fitting these parameters to maximise performance on the development set would lead to a small increase in generalisation performance, but at a significant cost in runtime. We replace tokens with count ≤ 1 in the training sample with one of roughly 50 generic unknown word markers which convey the token's lexical features and position in the sentence, following Petrov et al. (2006). We also right-binarise the trees to reduce the branching factor in the same manner as Petrov et al. (2006). The predicted trees are evaluated using EVALB[8] and we report the F1 score over labelled constituents and exact match accuracy over all sentences in the testing sets.

In our experiments, we initialized the sampler by setting all substitution variables to 0, thus treating every full tree in the training set as an elementary tree. Starting with all the variables set to 1 (corresponding to CFG expansions) or a random mix of 0s and 1s considerably increases time until convergence. We hypothesise that this is due to the sampler getting stuck in modes, from which a series of locally bad decisions are required to escape. The CFG solution seems to be a mode and therefore starting the sampler with maximal trees helps the model to avoid this mode. Other techniques could be used to improve training efficiency such as split-merge MCMC sampling (Jain and Neal, 2000) or replacing the sampler with a truncated variational approximation (Liang et al., 2007).

Small data sample For our first treebank experiments, we train on a small data sample by using only section 2 of the treebank. Bayesian methods tend to do well with small data samples, while for larger samples the benefits diminish relative to point estimates. The models were trained using Gibbs sampling for 4000 iterations with annealing linearly decreasing from $T = 5$ to $T = 1$, after which the model performed another 1000 iterations with $T = 1$. The final training sample was used in the parsing algorithm, which used 1000 derivation samples for each test sentence. All results are the average of five independent runs.

Table 1 presents the prediction results on the development set. The baseline is a maximum likelihood PCFG. The TSG model significantly outperforms the baseline with either base distribution P_0^M or P_0^C. This confirms our hypothesis that

[8]See `http://nlp.cs.nyu.edu/evalb/`.

		F1	EX	# rules
PCFG		60.20	4.29	3500
TSG P_0^M:	MPD	72.17	11.92	6609
	MPP	71.27	12.33	6609
	MER	74.25	12.30	6609
TSG P_0^C:	MPD	75.24	15.18	14923
	MPP	75.30	15.74	14923
	MER	76.89	15.76	14923
SM$_{\tau=2}$:	MPD	71.93	11.30	16168
	MER	74.32	11.77	16168
SM$_{\tau=5}$:	MPD	75.33	15.64	39758
	MER	77.93	16.94	39758

Table 1: Development results for models trained on section 2 of the Penn tree-bank, showing labelled constituent F1 and exact match accuracy. Grammar sizes are shown as the number of rules with count ≥ 1.

CFGs are not sufficiently powerful to model syntax, but that the increased context afforded to the TSG can make a large difference. This result is even more impressive when considering the difference in the sizes of grammar in the PCFG versus TSG models. The TSG using P_0^M achieves its improvements with only double as many rules, as a consequence of the prior which encourages sparse solutions. The TSG results with the CFG base distribution, P_0^C, are more accurate but with larger grammars.[9] This base distribution assigns proportionally higher probability to larger rules than P_0^M, and consequently the model makes use of these additional rules in a larger grammar.

Surprisingly, the MPP technique is not systematically better than the MPD approach, with mixed results under the F1 metric. We conjecture that this is due to sampling variance for long sentences, where repeated samples of the same tree are exceedingly rare. The MER technique results in considerably better F1 scores than either MPD or MPP, with a margin of 1.5 to 3 points. This method is less affected by sampling variance due to its use of smaller tree fragments (PCFG productions at each span).

For comparison, we trained the Berkeley split-merge (SM) parser (Petrov et al., 2006) on the same data and decoded using the Viterbi algorithm (MPD) and expected rule count (MER a.k.a. MAX-RULE-SUM). We ran two iterations of

[9]The grammar is nevertheless far smaller than the full DOP grammar induced from this data set, which has approximately 700K rules.

split-merge training, after which the development F1 dropped substantially (in contrast, our model is not fit to the development data). The result is an accuracy slightly below that of our model ($\mathrm{SM}_{\tau=2}$). To be fairer to their model, we adjusted the unknown word threshold to their default setting, i.e., to apply to word types occurring fewer than five times ($\mathrm{SM}_{\tau=5}$). We expect that tuning the treatment of unknown words in our model would also yield further gains. The grammar sizes are not strictly comparable, as the Berkeley binarised grammars prohibit non-binary rules, and are therefore forced to decompose each of these rules into many child rules. But the trend is clear – our model produces similar results to a state-of-the-art parser, and can do so using a small grammar. With additional rounds of split-merge training, the Berkeley grammar grows exponentially larger (200K rules after six iterations).

Full treebank We now train the model using P_0^M on the full training partition of the Penn treebank, using sections 2–21. We run the Gibbs sampler for 15,000 iterations while annealing from $T = 5$ to $T = 1$, after which we finish with 5,000 iterations at $T = 1$. We repeat this three times, giving an average F1 of 84.0% on the testing partition using the maximum expected rule algorithm and 83.0% using the Viterbi algorithm. This far surpasses the ML-PCFG (F1 of 70.7%), and is similar to Zuidema's (2007) DOP result of 83.8%. However, it still well below state-of-the art parsers (e.g., the Berkeley parser trained using the same data representation scores 87.7%). But we must bear in mind that these parsers have had the benefit of years of tuning to the Penn-treebank, where our model is fundamentally much simpler and is largely untuned. We anticipate that careful data preparation and model tuning could greatly improve our model's performance.

7 Discussion

So what kinds of non-CFG rules is the model learning? Figure 4 shows the grammar statistics for a TSG model trained on the small data sample. This model has 5611 CFG rules and 1008 TSG rules. The TSG rules vary in depth from two to nine levels with the majority between two and four. Most rules combine a small degree of lexicalisation and a variable or two. This confirms our intuition that the model is learning local structures to encode, e.g., multi-word units, subcategorisation frames and lexical agreement. The few very large rules specify full parses for sentences which were repeated in the training corpus. These complete trees are also evident in the long tail of node counts (up to 27; not shown in the figure) and counts for highly lexicalised rules (up to 8).

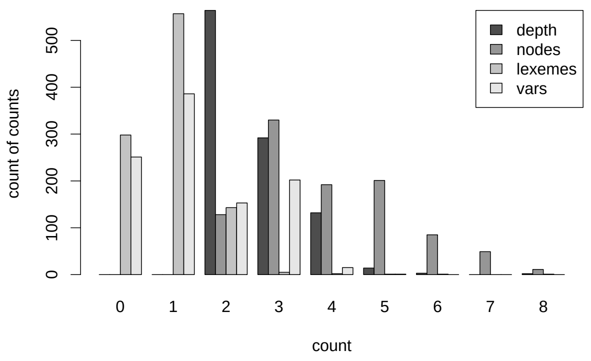


Figure 4: Grammar statistics for a TSG P_0^M model trained on section 2 of the Penn treebank, showing a histogram over elementary tree depth, number of nodes, terminals (lexemes) and frontier non-terminals (vars).

To get a better feel for the types of rules being learnt, it is instructive to examine the rules in the resultant grammar. Table 2 shows the top twenty rules for four phrasal categories for the model trained on the full Penn treebank. We can see that many of these rules are larger than CFG rules, showing that the CFG rules alone are inadequate to model the treebank. Two of the NP rules encode the prevalence of preposition phrases headed by 'of' within a noun phrase, as opposed to other prepositions. Also noteworthy is the lexicalisation of the determiner, which can affect the type of NP expansion. For instance, the indefinite article is more likely to have an adjectival modifier, while the definite article appears more frequently unmodified. Highly specific tokens are also incorporated into lexicalised rules, such as 'years' and 'old' in the last ADJP rule.

Many of the verb phrase expansions have been lexicalised, encoding the verb's subcategorisation, as shown in Table 3. Notice that each verb here accepts only one or a small set of argument frames, indicating that by lexicalising the verb in the VP expansion the model can find a less ambiguous and more parsimonious grammar.

The model also learns to use large rules to describe the majority of root node expansions (we add a distinguished TOP node to all trees). These

NP →	PP →	ADJP →
DT N̄P	IN NP	JJ
NNS	(IN in) NP	RB JJ
DT NN	(TO to) NP	JJ (AD̄JP CC JJ)
(DT the) N̄P	TO NP	JJ PP
JJ NNS	(IN with) NP	(RB very) JJ
NP (PP (IN of) NP)	(IN of) NP	RB AD̄JP
NP PP	(IN by) NP	(RBR more) JJ
NP (N̄P (CC and) NP)	(IN at) NP	JJ AD̄JP
JJ N̄P	IN (NP (DT the) N̄P)	ADJP (AD̄JP CC ADJP)
NN NNS	(IN on) NP	RB VBN
(DT the) NNS	(IN from) NP	RB (AD̄JP JJ PP)
DT (N̄P JJ NN)	IN (S (VP VBG NP))	JJ (PP (TO to) NP)
NN	IN (NP NP PP)	ADJP (PP (IN than) NP)
JJ NN	(IN into) NP	(RB too) JJ
(NP DT NN) (PP (IN of) NP)	(IN for) NP	(RB much) JJR
NP N̄P	IN (NP NP (N̄P CC NP))	(JJ UNK-LC)
NNP	(IN in) (NP (DT the) N̄P)	(ADJP JJR) (PP (IN than) NP)
NNP N̄P	IN (NP (NP DT NN) (PP (IN of) NP))	(RB so) JJ
PRP$ N̄P	(IN through) NP	VBN
(DT a) (N̄P JJ NN)	(IN as) NP	(NP CD (NNS years)) (JJ old)

Table 2: Top twenty expansions sorted by frequency (most frequent at top), taken from the final sample of a model trained on the full Penn treebank. Non-terminals shown with an over-bar denote a binarised sub span of the given phrase type.

NP →
(NNP Mr.) NNP
CD (NN %)
(NP CD (NN %)) (PP (IN of) NP)
(NP ($ $) CD) (NP (DT a) (NN share))
(NP (DT the) (N̄P (NN company) POS)) N̄P
(NP QP (NN %)) (PP (IN of) NP)
(NP CD (NNS cents)) (NP (DT a) (NN share))
(NP (NNP Mr.) (N̄P NNP (POS ’s))) NN
QP (NN %)
(NP (NN president)) (PP (IN of) NP)
(NP (NNP Mr.) (N̄P NNP (POS ’s))) N̄P
NNP (N̄P NNP (NNP Corp.))
NNP (N̄P NNP (NNP Inc.))
(NP (NN chairman)) (PP (IN of) NP)

VP →
(VBD said) (SBAR (S (NP (PRP it)) VP))
(VBD said) (SBAR (S NP VP))
(VBD rose) (V̄P (NP CD (NN %)) V̄P)
(VBP want) S
(VBD said) (SBAR (S (NP (PRP he)) VP))
(VBZ plans) S
(VBD said) (SBAR S)
(VBZ says) (SBAR (S NP VP))
(VBP think) (SBAR S)
(VBD agreed) (S (VP (TO to) (VP VB V̄P)))
(VBZ includes) NP
(VBZ says) (SBAR (S (NP (PRP he)) VP))
(VBZ wants) S
(VBD closed) (V̄P (PP (IN at) NP) (V̄P , ADVP))
(VBZ expects) S
(VBZ owns) NP
(VBP say) (SBAR S)
(VBD took) V̄P
(VBD failed) S
(VBD noted) (SBAR (IN that) S)

Table 3: Most frequent lexicalised expansions for noun and verb phrases, excluding auxiliary verbs.

rules mostly describe cases when the S category is used for a full sentence, which most often include punctuation such as the full stop and quotation marks. In contrast, the majority of expansions for the S category do not include any punctuation. The model has learnt to differentiate between the two different classes of S – full sentence versus internal clause – due to their different expansions.

8 Conclusion

In this work we have presented a non-parametric Bayesian model for inducing tree substitution grammars. By incorporating a structured prior over elementary rules our model is able to reason over the infinite space of all such rules, producing compact and simple grammars. In doing so our model learns local structures for latent linguistic phenomena, such as verb subcategorisation and lexical agreement. Our experimental results show that the induced grammars strongly out-perform standard PCFGs, and are comparable to a state-of-the-art parser on small data samples. While our results on the full treebank are well shy of the best available parsers, we have proposed a number of improvements to the model and the parsing algorithm that could lead to state-of-the-art performance in the future.

References

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

R. Bod. 2000. Combining semantic and syntactic structure for language modeling. In *Proceedings of the 6th International Conference on Spoken Language Processing*, Beijing, China.

R. Bod. 2003. An efficient implementation of a new DOP model. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary.

E. Charniak, M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 173–180, Ann Arbor, Michigan.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, 132–139.

D. Chiang, D. M. Bikel. 2002. Recovering latent information in treebanks. In *Proceedings of the 19th International Conference on Computational Linguistics*, 183–189, Taipei, Taiwan.

J. Cocke. 1969. *Programming languages and their compilers: Preliminary notes*. Courant Institute of Mathematical Sciences, New York University.

M. J. Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

J. R. Finkel, T. Grenager, C. D. Manning. 2007. The infinite tree. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 272–279, Prague, Czech Republic.

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

W. Gilks, S. Richardson, D. J. Spiegelhalter, eds. 1996. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, Suffolk.

S. Goldwater, T. Griffiths, M. Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of COLING/ACL*, Sydney.

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

S. Jain, R. 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.

M. Johnson, T. L. Griffiths, S. Goldwater. 2007a. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *Advances in Neural Information Processing Systems 19*.

M. Johnson, T. Griffiths, S. Goldwater. 2007b. 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*, 139–146, Rochester, New York.

M. Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4).

M. Johnson. 2002. The DOP estimation method is biased and inconsistent. *Computational Lingusitics*, 28(1):71–76.

A. Joshi. 2003. Tree adjoining grammars. In R. Mikkov, ed., *The Oxford Handbook of Computational Linguistics*, 483–501. Oxford University Press, Oxford, England.

P. Liang, S. Petrov, M. Jordan, D. Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 688–697, Prague, Czech Republic.

T. J. O'Donnell, N. D. Goodman, J. Snedeker, J. B. Tenenbaum. 2009a. Computation and reuse in language. In *31st Annual Conference of the Cognitive Science Society*, Amsterdam, The Netherlands. To appear.

T. J. O'Donnell, N. D. Goodman, J. B. Tenenbaum. 2009b. Fragment grammar: Exploring reuse in hierarchical generative processes. Technical Report MIT-CSAIL-TR-2009-013, MIT, 2009.

S. Petrov, D. Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, 404–411, Rochester, New York. Association for Computational Linguistics.

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

D. Prescher, R. Scha, K. Sima'an, A. Zollmann. 2004. On the statistical consistency of dop estimators. In *Proceedings of the 14th Meeting of Computational Linguistics in the Netherlands*, Antwerp, Belgium.

F. Xia. 2002. *Automatic grammar generation from two different perspectives*. Ph.D. thesis, University of Pennsylvania.

A. Zollmann, K. Sima'an. 2005. A consistent and efficient estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics*, 10(2):367–388.

W. Zuidema. 2007. Parsimonious data-oriented parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 551–560, Prague, Czech Republic.