Unsupervised Induction of Tree Substitution Grammars for Dependency Parsing

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

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

Abstract

Inducing a grammar directly from text is one of the oldest and most challenging tasks in Computational Linguistics. Significant progress has been made for inducing dependency grammars, however the models employed are overly simplistic, particularly in comparison to supervised parsing models. In this paper we present an approach to dependency grammar induction using tree substitution grammar which is capable of learning large dependency fragments and thereby better modelling the text. We define a hierarchical non-parametric Pitman-Yor Process prior which biases towards a small grammar with simple productions. This approach significantly improves the state-of-the-art, when measured by head attachment accuracy.

1 Introduction

Grammar induction is a central problem in Computational Linguistics, the aim of which is to induce linguistic structures from an unannotated text corpus. Despite considerable research effort this unsupervised problem remains largely unsolved, particularly for traditional phrase-structure parsing approaches (Clark, 2001; Klein and Manning, 2002). Phrase-structure parser induction is made difficult due to two types of ambiguity: the constituent structure and the constituent labels. In particular the constituent labels are highly ambiguous, firstly we don't know *a priori* how many there are, and secondly labels that appear high in a tree (e.g., an *S* category for a clause) rely on the correct inference of all the latent labels below them. However recent work on the induction of dependency grammars has proved more fruitful (Klein and Manning, 2004). Dependency grammars (Mel′čuk, 1988) should be easier to induce from text compared to phrase-structure grammars because the set of labels (heads) are directly observed as the words in the sentence.

Approaches to unsupervised grammar induction, both for phrase-structure and dependency grammars, have typically used very simplistic models (Clark, 2001; Klein and Manning, 2004), especially in comparison to supervised parsing models (Collins, 2003; Clark and Curran, 2004; McDonald, 2006). Simple models are attractive for grammar induction because they have a limited capacity to overfit, however they are incapable of modelling many known linguistic phenomena. We posit that more complex grammars could be used to better model the unsupervised task, provided that active measures are taken to prevent overfitting. In this paper we present an approach to dependency grammar induction using a tree-substitution grammar (TSG) with a Bayesian non-parametric prior. This allows the model to learn large dependency fragments to best describe the text, with the prior biasing the model towards fewer and smaller grammar productions.

We adopt the split-head construction (Eisner, 2000; Johnson, 2007) to map dependency parses to context free grammar (CFG) derivations, over which we apply a model of TSG induction (Cohn et al., 2009). The model uses a hierarchical Pitman-Yor process to encode a backoff path from TSG to CFG rules, and from lexicalised to unlexicalised rules. Our best lexicalised model achieves a head attachment accuracy of of 55.7% on Section 23 of the WSJ data set, which significantly improves over state-of-the-art and far exceeds an EM baseline (Klein and Manning, 2004) which obtains 35.9%.

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1204–1213,
MIT, Massachusetts, USA, 9-11 October 2010.

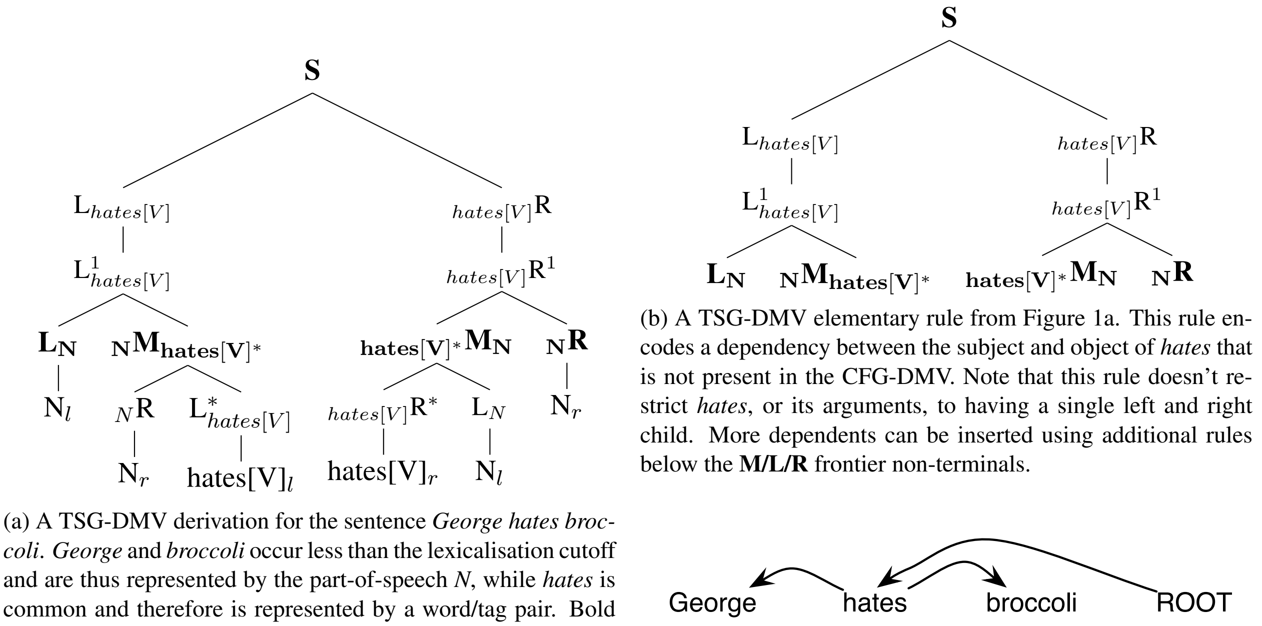
CFG Rule	**DMV Distribution**	**Description**
$\mathrm{S} \rightarrow \mathrm{L}_H\, {}_H\mathrm{R}$	$p(root = H)$	The head of the sentence is H.
$\mathrm{L}_H \rightarrow \mathrm{H}_l$	$p(STOP \mid dir = L, head = H, val = 0)$	H has no left children.
$\mathrm{L}_H \rightarrow \mathrm{L}^1_H$	$p(CONT \mid dir = L, head = H, val = 0)$	H has at least one left child.
$\mathrm{L}^*_H \rightarrow \mathrm{H}_l$	$p(STOP \mid dir = L, head = H, val = 1)$	H has no more left children.
$\mathrm{L}^*_H \rightarrow \mathrm{L}^1_H$	$p(CONT \mid dir = L, head = H, val = 1)$	H has another left child.
${}_H\mathrm{R} \rightarrow \mathrm{H}_r$	$p(STOP \mid dir = R, head = H, val = 0)$	H has no right children.
${}_H\mathrm{R} \rightarrow {}_H\mathrm{R}^1$	$p(CONT \mid dir = R, head = H, val = 0)$	H has at least one right child.
${}_H\mathrm{R}^* \rightarrow \mathrm{H}_r$	$p(STOP \mid dir = R, head = H, val = 1)$	H has no more right children.
${}_H\mathrm{R}^* \rightarrow {}_H\mathrm{R}^1$	$p(CONT \mid dir = R, head = H, val = 1)$	H has another right child.
$\mathrm{L}^1_H \rightarrow \mathrm{L}_C\, {}_C\mathrm{M}_{H^*}$	$p(C \mid dir = L, head = H)$	C is a left child of H.
${}_H\mathrm{R}^1 \rightarrow {}_{H^*}\mathrm{M}_C\, {}_C\mathrm{R}$	$p(C \mid dir = R, head = H)$	C is a right child of H.
${}_C\mathrm{M}_{H^*} \rightarrow {}_C\mathrm{R}\ \mathrm{L}^*_H$	$p = 1$	Unambiguous
${}_{H^*}\mathrm{M}_C \rightarrow {}_H\mathrm{R}^*\ \mathrm{L}_C$	$p = 1$	Unambiguous

Table 1: The CFG-DMV grammar schema. Note that the actual CFG is created by instantiating these templates with part-of-speech tags observed in the data for the variables H and C. Valency (*val*) can take the value 0 (no attachment in the direction (*dir*) d) and 1 (one or more attachment). L and R indicates child dependents left or right of the parent; superscripts encode the stopping and valency distributions, X^1 indicates that the head will continue to attach more children and X^* that it has already attached a child.

2 Background

The most successful framework for unsupervised dependency induction is the Dependency Model with Valence (DMV) (Klein and Manning, 2004). This model has been adapted and extended by a number of authors and currently represents the state-of-the-art for dependency induction (Cohen and Smith, 2009; Headden III et al., 2009). Eisner (2000) introduced the *split-head* algorithm which permits efficient $\mathcal{O}(|\mathbf{w}|^3)$ parsing complexity by replicating (splitting) each terminal and processing left and right dependents separately. We employ the related *fold-unfold* representation of Johnson (2007) that defines a CFG equivalent of the split-head parsing algorithm, allowing us to easily adapt CFG-based grammar models to dependency grammar. Table 1 shows the equivalent CFG grammar for the DMV model (CFG-DMV) using the unfold-fold transformation. The key insight to understanding the non-terminals in this grammar is that the subscripts encode the terminals at the boundaries of the span of that non-terminal. For example the non-terminal L_H encodes that the right most terminal spanned by this constituent is H (and the reverse for ${}_H\mathrm{R}$), while ${}_A\mathrm{M}_B$ encodes that A and B are the left-most and right-most terminals of the span. The * and ^{1} superscripts are used to encode the valency of the head, both indicate that the head has at least one attached dependent in the specified direction. This grammar allows $\mathcal{O}(|\mathbf{w}|^3)$ parsing complexity which follows from the terminals of the dependency tree being observed, such that each span of the parse chart uniquely specifies its possible heads (either the leftmost, rightmost or both) and therefore the number of possible non-terminals for each span is constant. The transform is illustrated in figures 1a and 1c which show the CFG tree for an example sentence and the equivalent dependency tree.

Normally DMV based models have been trained on part-of-speech tags of the words in a sentence, rather than the words themselves. Headden III et al. (2009) showed that performance could be improved by including high frequency words as well as tags in their model. In this paper we refer to such models as *lexicalised*; words which occur more than one hundred times in the training corpus are represented by a word/tag pair, while those less frequent are represented simply by their tags. We are also able to show that this basic approach to lexicalisation improves the performance of our models.



(a) A TSG-DMV derivation for the sentence *George hates broccoli*. *George* and *broccoli* occur less than the lexicalisation cutoff and are thus represented by the part-of-speech *N*, while *hates* is common and therefore is represented by a word/tag pair. Bold nodes indicate frontier nodes of elementary trees.

(b) A TSG-DMV elementary rule from Figure 1a. This rule encodes a dependency between the subject and object of *hates* that is not present in the CFG-DMV. Note that this rule doesn't restrict *hates*, or its arguments, to having a single left and right child. More dependents can be inserted using additional rules below the **M/L/R** frontier non-terminals.

(c) A traditional dependency tree representation of the parse tree in Figure 1a before applying the lexicalisation cutoff.

Figure 1: TSG-DMV representation of dependency trees.

3 Lexicalised TSG-DMV

The models we investigate in this paper build upon the CFG-DMV by defining a *Tree Substitution Grammar* (TSG) over the space of CFG rules. A 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 (rules). The productions take the form of *elementary trees* – tree fragments of height ≥ 1, 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. We can represent derivations as sequences of elementary trees, $\mathbf{e}$, by specifying that during the generation of the tree each elementary tree is substituted for the left-most frontier non-terminal. Figure 1a shows a TSG derivation for the dependency tree in Figure 1c where bold nonterminal labels denote substitution sites (root/frontier nodes in the elementary trees).

The probability of a derivation, $\mathbf{e}$, is the product of the probabilities of its component rules,

$$P(\mathbf{e}) = \prod_{c \rightarrow e \in \mathbf{e}} P(e|c) \,. \qquad (1)$$

where each rewrite is assumed conditionally independent of all others given its root nonterminal, $c = \text{root}(e)$. The probability of a tree, t, and string of words, $\mathbf{w}$, are

$$P(t) = \sum_{\mathbf{e}:\text{tree}(\mathbf{e})=t} P(\mathbf{e}) \;\text{ and }\; P(\mathbf{w}) = \sum_{t:\text{yield}(t)=\mathbf{w}} P(t)\,,$$

respectively, where tree($\mathbf{e}$) returns the tree for the derivation $\mathbf{e}$ and yield(t) returns the string of terminal symbols at the leaves of t.

A *Probabilistic Tree Substitution Grammar* (PTSG), like a PCFG, assigns a probability to each rule in the grammar, denoted $P(e|c)$. The probability of a derivation, $\mathbf{e}$, is the product of the probabilities of its component rules. Estimating a PTSG requires learning the sufficient statistics for $P(e|c)$ in (1) based on a training sample. Parsing involves finding the most probable tree for a given string ($\arg\max_t P(t|\mathbf{w})$). This is typically approximated by finding the most probable derivation which can be done efficiently using the CYK algorithm.

3.1 Model

In this work we propose the Tree Substitution Grammar Dependency Model with Valence (TSG-DMV). We define a hierarchical non-parametric TSG model on the space of parse trees licensed by the CFG grammar in Table 1. Our model is a generalisation of that of Cohn et al. (2009) and Cohn et al. (2011). We extend those works by moving from a single level Dirichlet Process (DP) distribution over rules to a multi-level Pitman-Yor Process (PYP), and including lexicalisation. The PYP has been shown to generate distributions particularly well suited to modelling language (Teh, 2006; Goldwater et al., 2006). Teh (2006) used a hierarchical PYP to model backoff in language models, we leverage this same capability to model backoff in TSG rules. This effectively allows smoothing from lexicalised to unlexicalised grammars, and from TSG to CFG rules.

Here we describe our deepest model which has a four level hierarchy, depicted graphically in Table 2. In Section 5 we evaluate different subsets of this hierarchy. The topmost level of our model describes lexicalised elementary elementary fragments (e) as produced by a PYP,

$$\begin{aligned} e|c &\sim G_c \\ G_c|a_c, b_c, \mathrm{P}^{\mathrm{lcfg}} &\sim \mathrm{PYP}(a_c, b_c, \mathrm{P}^{\mathrm{lcfg}}(\cdot|c))\,, \end{aligned}$$

where a_c and b_c control the strength of the backoff distribution $\mathrm{P}^{\mathrm{lcfg}}$. The space of lexicalised TSG rules will inevitably be very sparse, so the base distribution $\mathrm{P}^{\mathrm{lcfg}}$ backs-off to calculating the probability of a TSG rules as the product of the CFG rules it contains, multiplied by a geometric distribution over the size of the rule.

$$\begin{aligned} \mathrm{P}^{\mathrm{lcfg}}(e|c) &= \prod_{f\in\mathrm{F}(e)} s_{f_c} \prod_{i\in\mathrm{I}(e)} (1 - s_{i_c}) \\ &\quad \times A(\text{lex-cfg-rules}(e|c)) \\ \alpha|c &\sim A_c \\ A_c|a_c^{\mathrm{lcfg}}, b_c^{\mathrm{lcfg}}, \mathrm{P}^{\mathrm{cfg}} &\sim \mathrm{PYP}(a_c^{\mathrm{lcfg}}, b_c^{\mathrm{lcfg}}, \mathrm{P}^{\mathrm{cfg}}(\cdot|c)), \end{aligned}$$

where $I(e)$ are the set of internal nodes in e excluding the root, $F(e)$ are the set of frontier non-terminal nodes, and c_i is the non-terminal symbol for node i and s_c is the probability of stopping expanding a node labelled c. The function lex-cfg-rules$(e|c)$ returns the CFG rules internal to e, each of the form $c' \rightarrow \alpha$; each CFG rule is drawn from the backoff distribution, $A_{c'}$. We treat s_c as a parameter which is estimated during training, as described in Section 4.2.

The next level of backoff ($\mathrm{P}^{\mathrm{cfg}}$) removes the lexicalisation from the CFG rules, describing the generation of a lexicalised rule by first generating an unlexicalised rule from a PYP, then generating the lexicalisaton from a uniform distribution over words:[1]

$$\begin{aligned} \mathrm{P}^{\mathrm{cfg}}(\alpha|c) &= B(\text{unlex}(\alpha)|\text{unlex}(c)) \\ &\quad \times \frac{1}{|\mathbf{w}|^{|\alpha|}} \\ \alpha'|c' &\sim B_{c'} \\ B_{c'}|a_{c'}^{\mathrm{cfg}}, b_{c'}^{\mathrm{cfg}}, \mathrm{P}^{\mathrm{sh}} &\sim \mathrm{PYP}(a_{c'}^{\mathrm{cfg}}, b_{c'}^{\mathrm{cfg}}, \mathrm{P}^{\mathrm{sh}}(\cdot|c')), \end{aligned}$$

where unlex$(\cdot)$ removes the lexicalisation from non-terminals leaving only the tags.

The final base distribution over CFG-DMV rules (P^{sh}) is inspired by the *skip-head* smoothing model of Headden III et al. (2009). This model showed that smoothing the DMV by removing the heads from the CFG rules significantly improved performance. We replicate this behavior through a final level in our hierarchy which generates the CFG rules without their heads, then generates the heads from a uniform distribution:

$$\begin{aligned} \mathrm{P}^{\mathrm{sh}}(\alpha|c) &= C(\text{drop-head}(c \rightarrow \alpha)) \times \frac{1}{|P|} \\ \alpha|c &\sim C_c \\ C_c|a_c^{\mathrm{sh}}, b_c^{\mathrm{sh}} &\sim \mathrm{PYP}(a_c^{\mathrm{sh}}, b_c^{\mathrm{sh}}, \text{Uniform}(\cdot|c)), \end{aligned}$$

where drop-head$(\cdot)$ removes the symbols that mark the head on the CFG rules, and P is the set of part-of-speech tags. Each stage of backoff is illustrated in Table 2, showing the rules generated from the TSG elementary tree in Figure 1b.

Note that while the supervised model of Cohn et al. (2009) used a fixed back-off PCFG distribution, this model implicitly infers this distribution within

[1] All unlexicalised words are actually given the generic UNK symbol as their lexicalisation.

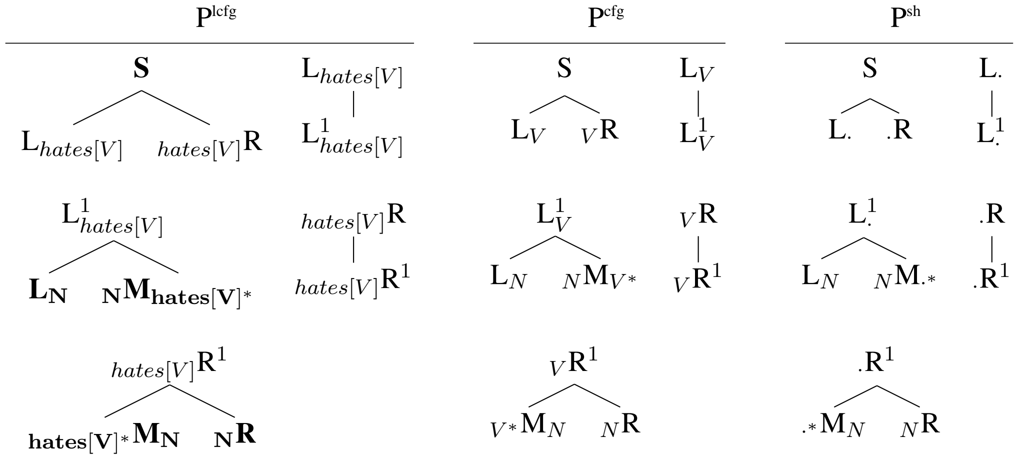


Table 2: Backoff trees for the elementary tree in Figure 1b.

its hierarchy, essentially learning the DMV model embedded in the TSG.

In this application to dependency grammar our model is capable of learning tree fragments which group CFG parameters. As such the model can learn to condition dependency links on the valence, e.g. by combining $\mathrm{L}_H \rightarrow \mathrm{L}^1_H$ and $\mathrm{L}^1_H \rightarrow \mathrm{L}_C\ {}_C\mathrm{M}_{H*}$ rules into a single fragment the model can learn a parameter that the leftmost child of H is C. By linking together multiple L^1_H or ${}_H\mathrm{R}^1$ non-terminals the model can learn groups of dependencies that occur together, e.g. tree fragments representing the complete preferred argument frame of a verb.

4 Inference

4.1 Training

To train our model we use Markov Chain Monte Carlo sampling (Geman and Geman, 1984). Where previous supervised TSG models (Cohn et al., 2009) permit an efficient local sampler, the lack of an observed parse tree in our unsupervised model makes this sampler not applicable. Instead we use a recently proposed blocked Metroplis-Hastings (MH) sampler (Cohn and Blunsom, 2010) which exploits a factorisation of the derivation probabilities such that whole trees can be sampled efficiently. See Cohn and Blunsom (2010) for details. That algorithm is applied using a dynamic program over an observed tree, the generalisation to our situation of an inside pass over the space of all trees is straightforward.

A final consideration is the initialisation of the sampler. Klein and Manning (2004) emphasised the importance of the initialiser for achieving good performance with their model. We employ the same *harmonic* initialiser as described in that work. The initial derivations for our sampler are the Viterbi derivations under the CFG parameterised according to this initialiser.

4.2 Sampling hyperparameters

We treat the hyper-parameters $\{(a^x_c, b^x_c, s_c), c \in N\}$ as random variables in our model and infer their values during training. We choose quite vague priors for each hyper-parameter, encoding our lack of information about their values.

We place prior distributions on the PYP discount a_c and concentration b_c hyperparamters and sample their values using a slice sampler. We use the range doubling slice sampling technique of (Neal, 2003) to draw a new sample of a'_c from its conditional distribution.[2] For the discount parameters a_c we employ a uniform Beta distribution, as we have no strong prior knowledge of what its value should be ($a_c \sim \mathrm{Beta}(1,1)$). Similarly, we treat the concentration parameters, b_c, as being generated by a vague gamma prior, $b_c \sim \mathrm{Gamma}(1,1)$, and sample a new value b'_c using the same slice-sampling approach as for a_c:

$$P(b_c|\mathbf{z}) \propto P(\mathbf{z}|b_c) \times \mathrm{Gamma}(b_c|1,1).$$

[2] We made use of the slice sampler included in Mark Johnson's Adaptor Grammar implementation http://www.cog.brown.edu/~mj/Software.htm.

Corpus	Words	Sentences
Sections 2-21 ($\|\mathbf{x}\| \leq 10$)	42505	6007
Section 22 ($\|\mathbf{x}\| \leq 10$)	1805	258
Section 23 ($\|\mathbf{x}\| \leq 10$)	2649	398
Section 23 ($\|\mathbf{x}\| \leq \infty$)	49368	2416

Table 3: Corpus statistics for the training and testing data for the TSG-DMV model. All models are trained on the gold standard part-of-speech tags after removing punctuation.

We use a vague Beta prior for the stopping probabilities in $\mathrm{P}^{\mathrm{lcfg}}$, $s_c \sim \mathrm{Beta}(1, 1)$.

All the hyper-parameters are resampled after every 10th sample of the corpus derivations.

4.3 Parsing

Unfortunately finding the maximising parse tree for a string under our TSG-DMV model is intractable due to the inter-rule dependencies created by the PYP formulation. Previous work has used Monte Carlo techniques to sample for one of the maximum probability parse (MPP), maximum probability derivation (MPD) or maximum marginal parse (MMP) (Cohn et al., 2009; Bod, 2006). We take a simpler approach and use the Viterbi algorithm to calculate the MPD under an approximating TSG defined by the last set of derivations sampled for the corpus during training. Our results indicate that this is a reasonable approximation, though the experience of other researchers suggests that calculating the MMP under the approximating TSG may also be beneficial for DMV (Cohen et al., 2008).

5 Experiments

We follow the standard evaluation regime for DMV style models by performing experiments on the text of the WSJ section of the Penn. Treebank (Marcus et al., 1993) and reporting head attachment accuracy. Like previous work we pre-process the training and test data to remove punctuation, training our unlexicalised models on the gold-standard part-of-speech tags, and including words occurring more than 100 times in our lexicalised models (Headden III et al., 2009). It is very difficult for an unsupervised model to learn from long training sentences as they contain a great deal of ambiguity, therefore the majority of DMV based models have been trained on sentences restricted in length to $\leq$ 10 tokens.[3] This has the added benefit of decreasing the runtime for experiments. We present experiments with this training scenario. The training data comes from sections 2-21, while section 23 is used for evaluation. An advantage of our sampling based approach over previous work is that we infer all the hyperparameters, as such we don't require the use of section 22 for tuning the model.

The models are evaluated in terms of head attachment accuracy (the percentage of correctly predicted head indexes for each token in the test data), on two subsets of the testing data. Although we can argue that unsupervised models are better learnt from short sentences, it is much harder to argue that we don't then need to be able to parse long sentences with a trained model. The most commonly employed test set mirrors the training data by only including sentences $\leq$ 10. In this work we focus on the accuracy of our models on the whole of section 23, without any pruning for length. The training and testing corpora statistics are presented in Table 3. Subsequent to the evaluation reported in Table 4 we use section 22 to report the correlation between heldout accuracy and the model log-likelihood (LLH) for analytic purposes.

As we are using a sampler during training, the result of any single run is non-deterministic and will exhibit a degree of variance. All our reported results are the mean and standard deviation (σ) from forty sampling runs.

5.1 Discussion

Table 4 shows the head attachment accuracy results for our TSG-DMV, plus many other significant previously proposed models. The subset of hierarchical priors used by each model is noted in brackets.

The performance of our models is extremely encouraging, particularly the fact that it achieves the highest reported accuracy on the full test set by a considerable margin. On the $|\mathbf{w}| \leq 10$ test set all the TSG-DMVs are second only to the L-EVG model of Headden III et al. (2009). The L-EVG model extends DMV by adding additional lexicalisation,

[3] See Spitkovsky et al. (2010a) for an exception to this rule.

	Directed Attachment Accuracy on WSJ23	
Model	$\lvert\mathbf{w}\rvert \leq 10$	$\lvert\mathbf{w}\rvert \leq \infty$
Attach-Right	38.4	31.7
EM (Klein and Manning, 2004)	46.1	35.9
Dirichlet (Cohen et al., 2008)	46.1	36.9
LN (Cohen et al., 2008)	59.4	40.5
SLN, TIE V&N (Cohen and Smith, 2009)	61.3	41.4
DMV (Headden III et al., 2009)	$55.7_{\sigma=8.0}$	-
DMV smoothed (Headden III et al., 2009)	$61.2_{\sigma=1.2}$	-
EVG smoothed (Headden III et al., 2009)	$65.0_{\sigma=5.7}$	-
L-EVG smoothed (Headden III et al., 2009)	$\mathbf{68.8}_{\sigma=4.5}$	-
Less is More (Spitkovsky et al., 2010a)	56.2	44.1
Leap Frog (Spitkovsky et al., 2010a)	57.1	45.0
Viterbi EM (Spitkovsky et al., 2010b)	65.3	47.9
Hypertext Markup (Spitkovsky et al., 2010c)	69.3	50.4
Adaptor Grammar (Cohen et al., 2010)	50.2	-
TSG-DMV ($\mathrm{P}^{\mathrm{cfg}}$)	$65.9_{\sigma=2.4}$	$53.1_{\sigma=2.4}$
TSG-DMV ($\mathrm{P}^{\mathrm{cfg}}$, P^{sh})	$65.1_{\sigma=2.2}$	$51.5_{\sigma=2.0}$
LexTSG-DMV ($\mathrm{P}^{\mathrm{lcfg}}$, $\mathrm{P}^{\mathrm{cfg}}$)	$67.2_{\sigma=1.4}$	$55.2_{\sigma=2.2}$
LexTSG-DMV ($\mathrm{P}^{\mathrm{lcfg}}$, $\mathrm{P}^{\mathrm{cfg}}$, P^{sh})	$67.7_{\sigma=1.5}$	$\mathbf{55.7}_{\sigma=2.0}$
Supervised MLE (Cohen and Smith, 2009)	84.5	68.8

Table 4: Mean and variance for the head attachment accuracy of our TSG-DMV models (highlighted) with varying backoff paths, and many other high performing models. Citations indicate where the model and result were reported. Our models labelled TSG used an unlexicalised top level G_c PYP, while those labelled LexTSG used the full lexicalised G_c.

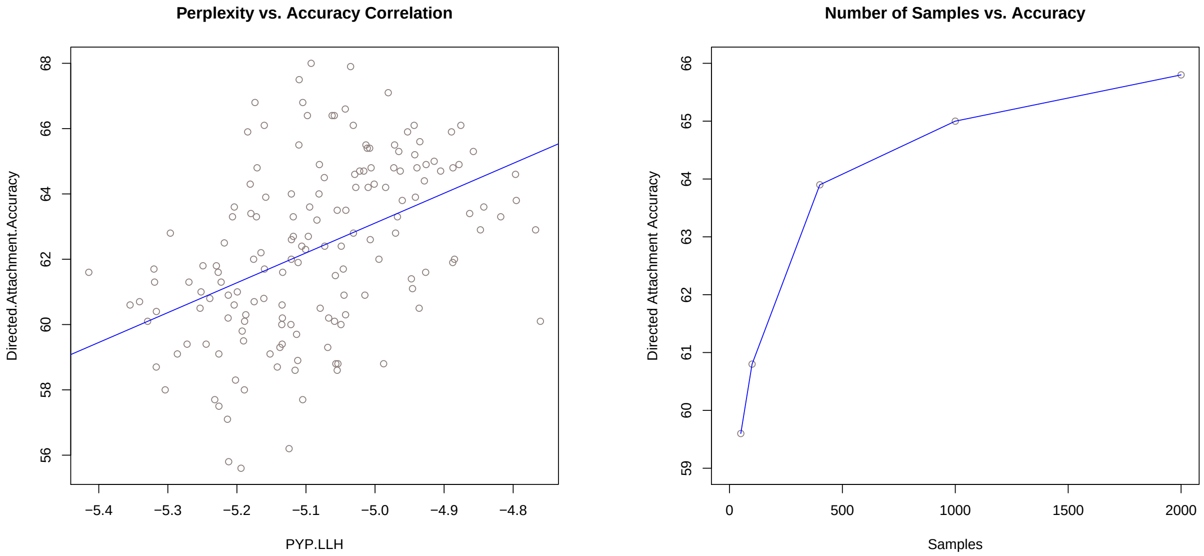
valency conditioning, interpolated back-off smoothing and a random initialiser. In particular Headden III et al. (2009) shows that the random initialiser is crucial for good performance, however this initialiser requires training 1000 models to select a single best model for evaluation and results in considerable variance in test set performance. Note also that our model exhibits considerably less variance than those induced using this random initialiser, suggesting that the combination of the harmonic initialiser and blocked-MH sampling may be a more practicable training regime.

The recently proposed Adaptor Grammar DMV model of Cohen et al. (2010) is similar in many way to our TSG model, incorporating a Pitman Yor prior over units larger than CFG rules. As such it is surprising that our model is performing significantly better than this model. We can identify a number of differences that may impact these results: the Adaptor Grammar model is trained using variational inference with the space of tree fragments truncated, while we employ a sampler which can nominally explore the full space of tree fragments; and the adapted tree fragments must be complete subtrees (i.e. they don't contain variables), whereas our model can make use of arbitrary tree fragments. An interesting avenue for further research would be to extend the variational algorithm of Cohen et al. (2010) to our TSG model, possibly speeding inference and allowing easier parallelisation.

In Figure 2a we graph the model LLH on the training data versus the head attachment accuracy on the heldout set. The graph was generated by running 160 models for varying numbers of samples and evaluating their accuracy. This graph indicates that the improvements in the posterior probability of the model are correlated with the evaluation, though the correlation is not as high as we might require in order to use LLH as a model selection criteria similar to Headden III et al. (2009). Further refinements to the model could improve this correlation.

The scaling perfomance of the model as the number of samples is increased is shown in Figure 2b. Performance improves as the training data is sampled for longer, and continues to trend upwards beyond 1000 samples (the point for which we've reported results in Table 4). This suggests that longer sampling runs – and better inference techniques – could yield further improvements.

For further analysis Table 5 shows the accuracy of the model at predicting the head for frequent types, while Table 6 shows the performance on dependencies of various lengths. We emphasise that these results are for the single best performing sampler run on the heldout corpus and there is considerable variation in the analyses produced by each sampler. Unsurprisingly, the model appears to be more accurate when predicting short dependencies, a result that is also reflected in the per type accuracies. The model is relatively good at identifying the root verb in each sentence, especially those headed by past tense verbs (VBD, *was*), and to a lesser degree VBPs (*are*). Conjunctions such as *and* pose a particular difficulty when evaluating dependency models as the correct modelling of these remains a



(a) Correlation ($R^2 = 0.2$) between the training LLH of the PYP Model and heldout directed head attachment accuracy (WSJ Section 22, $|\mathbf{w}| \leq 10$) for LexTSG-DMV ($\mathrm{P}^{\mathrm{lcfg}}$, $\mathrm{P}^{\mathrm{cfg}}$, P^{sh}).

(b) Mean heldout directed head attachment accuracy (WSJ Section 22, $|\mathbf{w}| \leq 10$) versus the number of samples used during training for LexTSG-DMV ($\mathrm{P}^{\mathrm{lcfg}}$, $\mathrm{P}^{\mathrm{cfg}}$, P^{sh}).

Figure 2

contentious linguistic issue and it's not clear what the 'correct' analysis should be. Our model gets a respectable 75% accuracy for *and* conjunctions, but for conjunctions (CC) as a whole, the model performs poorly (39%).

Table 7 list the most frequent TSG rules lexicalised with *has*. The most frequent rule is simply the single level equivalent of the DMV terminal rule for *has*. Almost as frequent is rule 3, here the grammar incorporates the terminal into a larger elementary fragment, encoding that it is the head of the past participle occuring immediately to it's right. This shows the model's ability to learn the verb's argument position conditioned on both the head and child type, something lacking in DMV. Rule 7 further refines this preferred analysis for *has been* by lexicalising both the head and child. Rules (4,5,8,10) employ similar conditioning for proper and ordinary nouns heading noun phrases to the left of *has*. We believe that it is the ability of the TSG to encode stronger constraints on argument positions that leads to the model's higher accuracy on longer sentences, while other models do well on shorter sentences but relatively poorly on longer ones (Spitkovsky et al., 2010c).

6 Conclusion

In this paper we have made two significant contributions to probabilistic modelling and grammar induction. We have shown that it is possible to successfully learn hierarchical Pitman-Yor models that encode deep and complex backoff paths over highly structured latent spaces. By applying these models to the induction of dependency grammars we have also been able to advance the state-of-the-art, increasing the head attachment accuracy on section 23 of the Wall Street Journal Corpus by more than 5%.

Further gains in performance may come from an exploration of the backoff paths employed within the model. In particular more extensive experimentation with alternate priors and larger training data may allow the removal of the lexicalisation cutoff which is currently in place to counter sparsity.

We envisage that in future many grammar formalisms that have been shown to be effective in supervised parsing, such as categorial, unification and tree adjoining grammars, will prove amenable to unsupervised induction using the hierarchical nonparametric modelling approaches we have demonstrated in this paper.

	Count	LexTSG-DMV Rules		
1	94	$\mathrm{L}^*{}_{has-VBZ}$	→	$(\mathrm{L}^*{}_{has-VBZ}\ \text{has-VBZ}_l)$
2	74	$\mathrm{L}^1_{has-VBZ}$	→	$(\mathrm{L}^1_{has-VBZ}\ (\mathrm{L}_{NN}\ \mathrm{L}^1_{NN})\ {}_{NN}\mathrm{M}_{has-VBZ^*})$
3	71	${}_{has-VBZ^*}\mathrm{M}_{VBN}$	→	$({}_{has-VBZ^*}\mathrm{M}_{VBN}\ ({}_{has-VBZ}\mathrm{R}^*\ \text{has-VBZ}_r)\ \mathrm{L}_{VBN})$
4	54	${}_{NN}\mathrm{M}_{has-VBZ^*}$	→	$({}_{NN}\mathrm{M}_{has-VBZ^*}\ {}_{NN}\mathrm{R}\ (\mathrm{L}^*{}_{has-VBZ}\ \text{has-VBZ}_l))$
5	36	${}_{NN}\mathrm{M}_{has-VBZ^*}$	→	$({}_{NN}\mathrm{M}_{has-VBZ^*}\ {}_{NN}\mathrm{R}\ \mathrm{L}^*{}_{has-VBZ})$
6	36	${}_{has-VBZ}\mathrm{R}^*$	→	$({}_{has-VBZ}\mathrm{R}^*\ ({}_{has-VBZ}\mathrm{R}^1\ {}_{has-VBZ^*}\mathrm{M}_{VBN}\ ({}_{VBN}\mathrm{R}\ \text{VBN}_r)))$
7	30	${}_{has-VBZ^*}\mathrm{M}_{been-VBN}$	→	$({}_{has-VBZ^*}\mathrm{M}_{been-VBN}\ ({}_{has-VBZ}\mathrm{R}^*\ \text{has-VBZ}_r)\ \mathrm{L}_{been-VBN})$
8	27	${}_{NNP}\mathrm{M}_{has-VBZ^*}$	→	$({}_{NNP}\mathrm{M}_{has-VBZ^*}\ {}_{NNP}\mathrm{R}\ (\mathrm{L}^*{}_{has-VBZ}\ \text{has-VBZ}_l))$
9	25	${}_{has-VBZ}\mathrm{R}$	→	$({}_{has-VBZ}\mathrm{R}\ ({}_{has-VBZ}\mathrm{R}^1\ {}_{has-VBZ^*}\mathrm{M}_{NNS}\ ({}_{NNS}\mathrm{R}\ {}_{NNS}\mathrm{R}^1)))$
10	18	$\mathrm{L}^1_{has-VBZ}$	→	$(\mathrm{L}^1_{has-VBZ}\ \mathrm{L}_{NNP}\ {}_{NNP}\mathrm{M}_{has-VBZ^*})$

Table 7: The ten most frequent LexTSG-DMV rules in a final training sample that contain *has*.

References

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Proc. of the 44th Annual Meeting of the ACL and 21st International Conference on Computational Linguistics (COLING/ACL-2006)*, pages 865–872, Sydney, Australia, July.

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proc. of the 42nd Annual Meeting of the ACL (ACL-2004)*, pages 103–110, Barcelona, Spain.

Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *ConLL '01: Proceedings of the 2001 workshop on Computational Natural Language Learning*, pages 1–8. Association for Computational Linguistics.

Shay B. Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 74–82, Morristown, NJ, USA. Association for Computational Linguistics.

Shay B. Cohen, Kevin Gimpel, and Noah A. Smith. 2008. Logistic normal priors for unsupervised probabilistic grammar induction. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Lon Bottou, editors, *NIPS*, pages 321–328. MIT Press.

Shay B. Cohen, David M. Blei, and Noah A. Smith. 2010. Variational inference for adaptor grammars. In *Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Trevor Cohn and Phil Blunsom. 2010. Blocked inference in Bayesian tree substitution grammars. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, page To Appear, Uppsala, Sweden.

Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics on ZZZ*, pages 548–556, Morristown, NJ, USA. Association for Computational Linguistics.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2011. Inducing tree-substitution grammars. *Journal of Machine Learning Research*. To Appear.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers, 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, Tom Griffiths, and Mark Johnson. 2006. Interpolating between types and tokens by estimating power-law generators. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 459–466. MIT Press, Cambridge, MA.

William P. Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, Boulder, Colorado, June.

Child Tag	Predicted Head Correct	Accuracy (%)
NN	181	0.64
NNP	130	0.71
DT	127	0.87
NNS	108	0.72
VBD	108	0.81
JJ	106	0.80
IN	81	0.55
RB	65	0.61
PRP	64	0.97
VBZ	47	0.80
VBN	36	0.86
VBP	30	0.77
CD	26	0.23
VB	25	0.68
the	42	0.88
was	29	0.97
The	25	0.83
of	18	0.78
a	18	0.90
to	17	0.50
in	16	0.89
is	15	0.79
n't	15	0.83
were	12	0.86
are	11	0.92
It	11	1.00
for	9	0.64
and	9	0.75
's	9	1.00

Table 5: Per tag type predicted count and accuracy, for the most frequent 15 un/lexicalised tokens on the WSJ Section 22 $|\mathbf{w}| \leq 10$ heldout set (LexTSG-DMV $(\mathrm{P}^{\mathrm{lcfg}}, \mathrm{P}^{\mathrm{cfg}}, \mathrm{P}^{\mathrm{sh}})$).

Distance	Precision	Recall	F1
1	0.70	0.75	0.72
2	0.70	0.62	0.65
3	0.66	0.62	0.64
4	0.56	0.56	0.56
5	0.53	0.49	0.51
6	0.59	0.66	0.62
7	0.50	0.44	0.47
8	0.57	0.33	0.42
9	0.67	0.40	0.50
10	1.00	0.17	0.29

Table 6: Link distance precision, recall and f-score, on the WSJ Section 22 $|\mathbf{w}| \leq 10$ heldout set.

Mark Johnson. 2007. Transforming projective bilexical dependency grammars into efficiently-parsable CFGs with unfold-fold. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 168–175, Prague, Czech Republic, June. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: models of dependency and constituency. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 478.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Igor′ A. Mel′čuk. 1988. *Dependency Syntax: theory and practice*. State University of New York Press, Albany.

Radford Neal. 2003. Slice sampling. *Annals of Statistics*, 31:705–767.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010a. From Baby Steps to Leapfrog: How "Less is More" in unsupervised dependency parsing. In *Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Valentin I. Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D. Manning. 2010b. Viterbi training improves unsupervised dependency parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL-2010)*.

Valentin I. Spitkovsky, Daniel Jurafsky, and Hiyan Alshawi. 2010c. Profiting from mark-up: Hyper-text annotations for guided parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*.

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