Using Left-corner Parsing to Encode Universal Structural Constraints in Grammar Induction

Hiroshi Noji
Graduate School of Information Science
Nara Institute of Science and Technology
noji@is.naist.jp

Yusuke Miyao
National Institute of Informatics
yusuke@nii.ac.jp

Mark Johnson
Department of Computing
Macquarie University
mark.johnson@mq.edu.au

Abstract

Center-embedding is difficult to process and is known as a rare syntactic construction across languages. In this paper we describe a method to incorporate this assumption into the grammar induction tasks by restricting the search space of a model to trees with limited center-embedding. The key idea is the tabulation of *left-corner* parsing, which captures the degree of center-embedding of a parse via its stack depth. We apply the technique to learning of famous generative model, the dependency model with valence (Klein and Manning, 2004). Cross-linguistic experiments on Universal Dependencies show that often our method boosts the performance from the baseline, and competes with the current state-of-the-art model in a number of languages.

1 Introduction

Human languages in the world are divergent, but they also exhibit many striking similarities (Greenberg, 1963; Hawkins, 2014). At the level of syntax, one attractive hypothesis for such regularities is that any grammars of languages have evolved under the pressures, or biases, to avoid structures that are difficult to process. For example it is known that many languages have a preference for shorter dependencies (Gildea and Temperley, 2010; Futrell et al., 2015), which originates from the difficulty in processing longer dependencies (Gibson, 2000).

Such syntactic regularities can also be useful in applications, in particular in unsupervised (Klein and Manning, 2004; Mareček and Žabokrtský, 2012; Bisk and Hockenmaier, 2013) or weakly-supervised (Garrette et al., 2015) grammar induction tasks, where the models try to recover the syntactic structure of language without access to the syntactically annotated data, e.g., from raw or part-of-speech tagged text only. In these settings, finding better syntactic regularities universal across languages is essential, as they work as a small cue to the correct linguistic structures. A preference exploited in many previous works is favoring shorter dependencies, which has been encoded in various ways, e.g., initialization of EM (Klein and Manning, 2004), or model parameters (Smith and Eisner, 2006), and this has been the key to success of learning (Gimpel and Smith, 2012).

In this paper, we explore the utility for another universal syntactic bias that has not yet been exploited in grammar induction: a bias against center-embedding. Center-embedding is a syntactic construction on which a clause is embedded into another one. An example is "*The reporter [who the senator [who Mary met] attacked] ignored the president.*", where "*who Mary met*" is embedded in a larger relative clause. These constructions are known to cause memory overflow (Miller and Chomsky, 1963; Gibson, 2000), and also are rarely observed cross-linguistically (Karlsson, 2007; Noji and Miyao, 2014). Our learning method exploits this universal property of language. Intuitively during learning our models explore the restricted search space, which excludes linguistically implausible trees, i.e., those with deeper levels of center-embedding.

We describe how these constraints can be imposed in EM with the inside-outside algorithm. The central

SHIFT	$\sigma^{d-1} \overset{a}{\mapsto} \sigma^{d-1}\|A^d$	$A \to a \in P$
SCAN	$\sigma^{d-1}\|B/A^d \overset{a}{\mapsto} \sigma^{d-1}\|B^d$	$A \to a \in P$
PRED	$\sigma^{d-1}\|A^d \overset{\varepsilon}{\mapsto} \sigma^{d-1}\|B/C^d$	$B \to A\,C \in P$
COMP	$\sigma^{d-1}\|D/B^d\|A^{d+1} \overset{\varepsilon}{\mapsto} \sigma^{d-1}\|D/C^d$	$B \to A\,C \in P$

Figure 1: A set of transitions in left-corner parsing. The rules on the right side are the side conditions, in which P is the set of rules of a given CFG.

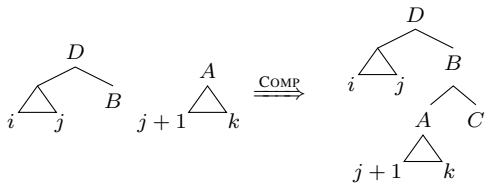


Figure 2: COMP combines two subtrees on the top of the stack. i, j, k are indices of spans.

idea is to tabulate *left-corner* parsing, on which its stack depth captures the degree of center-embedding of a partial parse. Each chart item keeps the current stack depth and we discard all items where the depth exceeds some threshold. The technique is general and can be applicable to any model on PCFG; in this paper, specifically, we describe how to apply the idea on the dependency model with valence (DMV) (Klein and Manning, 2004), a famous generative model for dependency grammar induction.

We focus our evaluation on grammar induction from part-of-speech tagged text, comparing the effect of several biases including the one against longer dependencies. Our main empirical finding is that though two biases, avoiding center-embedding and favoring shorter dependencies, are conceptually similar (both favor simpler grammars), often they capture different aspects of syntax, leading to different grammars. In particular our bias cooperates well with additional small syntactic cue such as the one that the sentence root tends to be a verb or a noun, with which our models compete with the strong baseline relying on a larger number of hand crafted rules on POS tags (Naseem et al., 2010).

Our contributions are: the idea to utilize left-corner parsing for a tool to constrain the models of syntax (Section 3), the formulation of this idea for DMV (Section 4), and cross-linguistic experiments across 25 languages to evaluate the universality of the proposed approach (Sections 5 and 6).

2 Left-corner parsing

We first describe (arc-eager) left-corner (LC) parsing as a push-down automaton (PDA), and then reformulate it as a grammar transform. In previous work this algorithm has been called *right-corner* parsing (e.g., Schuler et al. (2010)); we avoid this term and instead treat it as a variant of LC parsing following more recent studies, e.g., van Schijndel and Schuler (2013). The central motivation for this technique is to detect center-embedding in a parse efficiently. We describe this mechanism after providing the algorithm itself. We then give historical notes on LC parsing at the end of this section.

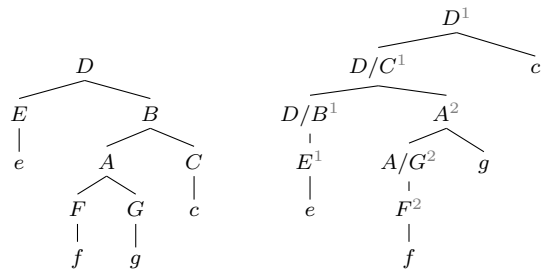
PDA Let us assume a CFG is given, and it is in CNF. We formulate LC parsing as a set of transitions between configurations, each of which is a pair of the stack and the input position (next input symbol). In Figure 1 a transition $\sigma_1 \overset{a}{\mapsto} \sigma_2$ means that the stack is changed from σ_1 to σ_2 by reading the next input symbol a. We use a vertical bar to signify the append operation, e.g., $\sigma = \sigma'|\sigma_1$ denotes σ_1 is the topmost symbol of σ. Each stack symbol is either a nonterminal, or a pair of nonterminals, e.g., A/B, which represents a subtree rooted at A and is awaiting symbol B. We also decorate each symbol with depth; for example, $\sigma^{d-1}|A^d$ means the current stack depth is d, and the depth of the topmost symbol in σ is $d-1$. The bottom symbol on the stack is always the empty symbol ε^0 with depth 0. Parsing begins with ε^0. Given the start symbol of CFG S, it finishes when S^1 is found on the stack.

The key transition here is COMP (Figure 2).[1] Basically the algorithm builds a tree by expanding the hypothesis from left to right. In COMP, a subtree rooted at A is combined with the second top subtree (D/B) on the stack. This can be done by first *predicting* that A's parent symbol is B and its sibling is C; then it unifies two different Bs to combine them. PRED is simpler, and it just predicts the parent and sibling symbols of A. The input symbols are read by SHIFT and SCAN: SHIFT addes a new element on the stack while SCAN fills in the predicted sibling symbol. For an example, Figure 3 shows how

[1] van Schijndel and Schuler (2013) employ different transition names, e.g., L- and L+; we avoid them as they are less informative.

Step	Transition	Stack	Next input symbol
0		ε	e
1	SHIFT	E^1	f
2	PRED	D/B^1	f
3	SHIFT	$D/B^1\ F^2$	g
4	PRED	$D/B^1\ A/G^2$	g
5	SCAN	$D/B^1\ A^2$	c
6	COMP	D/C^1	c
7	SCAN	D^1	

Figure 3: Sequence of transitions in LC PDA to parse the tree in Figure 4(a).



(a) (b)

Figure 4: An example of LC transform: (a) the original parse; and (b) the transformed parse.

this PDA works for parsing a tree in Figure 4(a).

Grammar transform The algorithm above can be reformulated as a grammar transform, which becomes the starting point for our application to grammar induction. This can be done by extracting the operated top symbols on the stack in each transition:

SHIFT: $A^d \rightarrow a \quad (A \rightarrow a \in P)$;
SCAN: $B^d \rightarrow B/A^d\ a \quad (A \rightarrow a \in P)$;
PRED: $B/C^d \rightarrow A^d \quad (B \rightarrow A\ C \in P)$;
COMP: $D/C^d \rightarrow D/B^d\ A^{d+1} \quad (B \rightarrow A\ C \in P)$.

where a rule on the right side is a condition given the set of rules P in the CFG.

Figure 4 shows an example of this transform. The essential point is that each CFG rule in the transformed parse (b) corresponds to a transition in the original algorithm (Figure 1). For example a rule $D/C^1 \rightarrow D/B^1\ A^2$ in the parse indicates that the stack configuration $D/B^1|A^2$ occurs during parsing (just corresponding to the step 5 in Figure 3) and COMP is then applied. This can also be seen as an instantiation of Figure 2.

Stack depth and center-embedding We use the term *center-embedding* to distinguish just the tree structures, i.e., ignoring symbols. That is, the tree in Figure 4(a) is the minimal, one degree of center-embedded tree, where the constituent rooted at A is embedded into a larger constituent rooted at D. Multiple, or degree ≥ 2 of center-embedding occurs if this constituent is also embedded into another larger constituent.

Note that it is only COMP that consumes the top *two* symbols on the stack. This means that a larger stack depth occurs only when COMP is needed. Furthermore, from Figure 2 COMP always induces a subtree involving new center-embedding, and this is the underlying mechanism that the stack depth of the algorithm captures the degree of center-embedding.

One thing to note is that to precisely associate the stack depth and the degree of center-embedding the depth calculation in COMP should be revised as:

$$\text{COMP: } D/C^d \rightarrow D/B^d\ A^{d'} \quad (B \rightarrow A\ C \in P)$$
$$d' = \begin{cases} d & (\textsc{SpanLen}(A) = 1) \\ d+1 & (\text{otherwise}), \end{cases} \quad (1)$$

where $\textsc{SpanLen}(A)$ calculates the span length of the constituent rooted at A, which is 2 in Figure 4(b). This modification is necessary since COMP for a single token occurs for building purely right-branching structures.[2] Formally, then, given a tree with degree λ of center-embedding the largest stack depth d^* during parsing this tree is: $d^* = \lambda + 1$.

Schuler et al. (2010) found that on English treebanks larger stack depth such as 3 or 4 rarely occurs while Noji and Miyao (2014) validated the language universality of this observation through cross-linguistic experiments. These suggest we may utilize LC parsing as a tool for exploiting universal syntactic biases as we discuss in Section 3.

Historical notes Rosenkrantz and Lewis (1970) first presented the idea of LC parsing as a grammar transform. This is *arc-standard*, and has no relevance to center-embedding; Resnik (1992) and Johnson (1998) formulated an *arc-eager* variant by extending this algorithm. The presented algorithm here is the same as Schuler et al. (2010), and is slightly different from Johnson (1998). The difference is in the start and end conditions: while

[2] Schuler et al. (2010) skip this subtlety by only concerning stack depth after PRED or COMP. We do not take this approach since ours allows a flexible extension described in Section 3.

our parser begins with an empty symbol, Johnson's parser begins with the predicted start symbol, and finishes with an empty symbol.

3 Learning with structural constraints

Now we discuss how to utilize LC parsing for grammar induction in general. An important observation in the above transform is that if we perform chart parsing, e.g., CKY, we can detect center-embedded trees efficiently in a chart. For example, by setting a threshold of stack depth δ, we can eliminate any parses involving center-embedding up to degree $\delta-1$. Note that in a probabilistic setting, each weight of a transformed rule comes from the corresponding underlying CFG rule (i.e., the condition).

For learning, our goal is to estimate θ of a generative model $p(z, x|\theta)$ for parse z and its yields (words) x. We take an EM-based simplest approach, and multiply the original model by a constraint factor $f(z, x) \in [0, 1]$ to obtain a new model:

$$p'(z, x|\theta) \propto p(z, x|\theta) f(z, x), \tag{2}$$

and then optimize θ based on $p'(z, x|\theta)$. This is essentially the same approach as Smith and Eisner (2006). As shown in Smith (2006), when training with EM we can increase the likelihood of $p'(z, x|\theta)$ by just using the expected counts from an E-step on the unnormalized distribution $p(z, x|\theta) f(z, x)$.

We investigate the following constraints in our experiments:

$$f(z, x) = \begin{cases} 0 & (d_z^* > \delta) \\ 1 & (\text{otherwise}), \end{cases} \tag{3}$$

where d_z^* is the largest stack depth for z in LC parsing and δ is the threshold. This is a hard constraint, and can easily be achieved by removing all chart items (of LC transformed grammar) on which the depth of the symbol exceeds δ. For example, when $\delta = 1$ the model only explores trees without center-embedding, i.e., right- or left-linear trees.

Length-based constraints By $\delta = 2$, the model is allowed to explore trees with one degree of center-embedding. Besides these simple ones, we also investigate relaxing $\delta = 1$ that results in an intermediate between $\delta = 1$ and 2. Specifically, we relax the depth calculation in COMP (Eq. 1) as follows:

$$d' = \begin{cases} d & (\textsc{SpanLen}(A) \leq \xi) \\ d + 1 & (\text{otherwise}), \end{cases} \tag{4}$$

where $\xi \geq 1$ controls the minimal length of a span regarded as *embedded* into another one. For example, when $\xi = 2$, the parse in Figure 4(a) is *not* regarded as center-embedded because the span length of the constituent reduced by COMP (i.e., A) is 2.

This modification is motivated with our observation that in many cases center-embedded constructions arise due to embedding of small chunks, rather than clauses. An example is "*... prepared [the cat 's] dinner*", where "*the cat 's*" is center-embedded in our definition. For this sentence, by relaxing the condition with, e.g., $\xi = 3$, we can suppress the increase of stack depth. We treat ξ as a hyperparameter in our experiments, and in practice, we find that this relaxed constraint leads to higher performance.

4 Dependency grammar induction

In this section we discuss how we can formulate the dependency model with valence (DMV) (Klein and Manning, 2004), a famous generative model for *dependency* grammar induction, on LC parsing. Though as we will see, applying LC parsing for a dependency model is a little involved compared to simple PCFG models, dependency models have been the central for the grammar induction tasks, and we consider it is most appropriate for assessing the effectiveness of our approach.

DMV is a head-outward generative model of a dependency tree, controlled by two types of multinomial distributions. For *stop* $\in \{\textsc{stop}, \neg\textsc{stop}\}$, $\theta_{\text{S}}(\textit{stop}|h, \textit{dir}, \textit{adj})$ is a Bernoulli random variable to decide whether or not to attach further dependents in *dir* $\in \{\leftarrow, \rightarrow\}$ direction. The adjacency *adj* $\in \{\textsc{true}, \textsc{false}\}$ is the key factor to distinguish the distributions of the first and the other dependents, which is TRUE if h has no dependent yet in *dir* direction. Another type of parameter is $\theta_{\text{A}}(a|h, \textit{dir})$, a probability that h takes a as a dependent in *dir* direction.

For this particular model, we take the following approach to formulate it in LC parsing: 1) converting a dependency tree into a binary CFG parse; 2) applying LC transform on it; and 3) encoding DMV

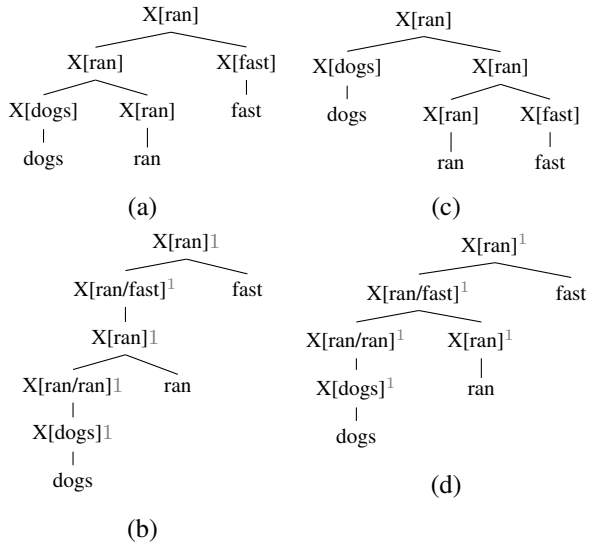


Figure 5: Two CFG parses for "dogs ran fast" and the results of LC transform ((a) → (b); (c) → (d)). X[a/b] is an abbreviation for X[a]/X[b].

parameters into each CFG rule of the transformed grammar.[3] Below we discuss a problem for (1) and (2), and then consider parameterization.[4]

Spurious ambiguity The central issue for applying LC parsing is the *spurious ambiguity* in dependency grammars. That is, there are more than one (binary) CFG parses corresponding to a given dependency tree. This is problematic mainly for two reasons: 1) we cannot specify the degree of center-embedding in a dependency tree uniquely; and 2) this one-to-many mapping prevents the inside-outside algorithm to work correctly (Eisner, 2000).

As a concrete example, Figures 5(a) and 5(c) show two CFG parses corresponding to the dependency tree *dogs*↶*ran*↷*fast*. We approach this problem by first providing a grammar transform, which generates all valid LC transformed parses (e.g., Figures 5(b) and 5(d)) and then restricting the grammar

[3]Another approach might be just applying the technique in Section 3 to some PCFG that encodes DMV, e.g., Headden III et al. (2009). The problem with this approach, in particular with split-head grammars (Johnson, 2007), is that the calculated stack depth no longer reflects the degree of center-embedding in the original parse correctly. As we discuss later, instead, we can speed up inference by applying head-splitting *after* obtaining the LC transformed grammar.

[4]Technical details including the chart algorithm for split-head grammars can be found in the Ph.D. thesis of the first author (Noji, 2016).

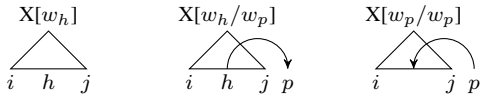


Figure 6: The senses of the symbols as a chart item. X[w_h/w_p] *predicts* the next dependent outside of the span while X[w_p/w_p] predicts the head.

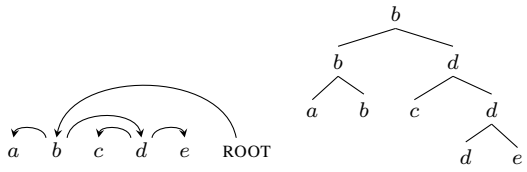


Figure 7: Implicit binarization of the restricted grammar. For each token, if its parent is in the right side (e.g., b), it attaches all left children first. The behavior is opposed when the parent is in its left (e.g., d). A dummy root token is placed at the end.

for generating particular parses only.

Naive method Let us begin with the grammar below, which suffers from the spurious ambiguity:

SHIFT: $\mathrm{X}[w_h]^d \rightarrow w_h$
SCAN: $\mathrm{X}[w_h]^d \rightarrow \mathrm{X}[w_h/w_p]^d\ w_p$
L-PRED: $\mathrm{X}[w_p/w_p]^d \rightarrow \mathrm{X}[w_h]^d\ (w_h \overset{\frown}{} w_p)$;
R-PRED: $\mathrm{X}[w_h/w_p]^d \rightarrow \mathrm{X}[w_h]^d\ (w_h \overset{\frown}{} w_p)$;
L-COMP: $\mathrm{X}[w_h/w_p]^d \rightarrow \mathrm{X}[w_h/w_p]^d\mathrm{X}[w_a]^{d'}\ (w_a \overset{\frown}{} w_p)$;
R-COMP: $\mathrm{X}[w_h/w_a]^d \rightarrow \mathrm{X}[w_h/w_p]^d\mathrm{X}[w_p]^{d'}\ (w_p \overset{\frown}{} w_a)$.

Here X[a/b] denotes X[a]/X[b] while w_h denotes the h-th word in the sentence w. We can interpret these rules as the operations on chart items (Figure 6). Note that only PRED and COMP create new dependency arcs and we divide them depending on the direction of the created arcs (L and R). d' is calculated by Eq. 4. Note also that for L-COMP and R-COMP h might equal p; X[ran/fast]^1 → X[ran/ran]^1 X[ran]^2 in Figure 5(d) is such a case for R-COMP.

Removing spurious ambiguity We can show that by restricting conditions for some rules, the spurious ambiguity can be eliminated (the proof is omitted).

1. Prohibit R-COMP when $h = p$;
2. Assume the span of $\mathrm{X}[w_p]^{d'}$ is (i, j) ($i \leq p \leq j$). Then allow R-COMP only when $i = p$.

Intuitively, these conditions constraint the order that each word collects its left and right children. For

example, by the condition 1, this grammar is prohibited to generate the parse of Figure 5(d).

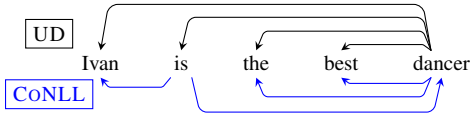
Binarization Note that two CFG parses in Figures 5(a) and 5(c) differ in how we *binarize* a given dependency tree. This observation indicates that our restricted grammar *implicitly* binarizes a dependency tree, and the incurred stack depth (or the degree of center-embedding) is determined based on the structure of the binarized tree. Specifically, we can show that the presented grammar performs *optimal* binarization; i.e., it *minimizes* the incurred stack depth. Figure 7 shows an example, which is not regarded as center-embedded in our procedure. In summary, our method detects center-embedding for a dependency tree, but the degree is determined based on the structure of the binarized CFG parse.

Parameterization We can encode DMV parameters into each rule. A new arc is introduced by one of {L/R}-{PRED/COMP}, and the stop probabilities can be assigned appropriately in each rule by calculating the valence from indices in the rule. For example, after L-PRED, w_h does not take any right dependents so $\theta_{\mathrm{S}}(\mathit{stop}|w_h, \rightarrow, h = j)$, where j is the right span index of $\mathrm{X}[w_h]$, is multiplied.

Improvement Though we omit the details, we can improve the time complexity of the above grammar from $O(n^6)$ to $O(n^4)$ applying the technique similar to Eisner and Satta (1999) without changing the binarization mechanism mentioned above. We implemented this improved grammar.

5 Experimental setup

A sound evaluation metric in grammar induction is known as an open problem (Schwartz et al., 2011; Bisk and Hockenmaier, 2013), which essentially arises from the ambiguity in the notion of head. For example, Universal dependencies (UD) is the recent standard in annotation and prefers content words to be heads, but as shown below this is very different from the conventional style, e.g., the one in CoNLL shared tasks (Johansson and Nugues, 2007):



The problem is that both trees are *correct* under some linguistic theories but the standard metric, unlabeled attachment score (UAS), only takes into account the annotation of the current gold data.

Our goal in this experiment is to assess the effect of our structural constraints. To this end, we try to eliminate such arbitrariness in our evaluation as much as possible in the following way:

- We experiment on UD, in which every treebank follows the consistent UD style annotation.
- We restrict the model to explore only trees that follow the UD style annotation during learning[5], by prohibiting every function word[6] in a sentence to have any dependents.
- We calculate UAS in a standard way.

We use UD of version 1.2. Some treebanks are very small, so we select the top 25 largest languages. The input to the model is coarse universal POS tags. Punctuations are stripped off. All models are trained on sentences of length ≤ 15 and tested on ≤ 40.

Initialization Much previous work of dependency grammar induction relies on the technique called harmonic initialization, which also biases the model towards shorter dependencies (Klein and Manning, 2004). Since our focus is to see the effect of structural constraints, we do not try this and initialize models uniformly. However, we add a baseline model with this initialization in our comparison to see the relative strength of our approach.

Models For the baseline, we employ a variant of DMV with *features* (Berg-Kirkpatrick et al., 2010), which is simple yet known to boost the performance well. The feature templates are almost the same; the only change is that we add backoff features for STOP probabilities that ignore both direction and adjacency, which we found slightly improves the performance in a preliminary experiment. We set the regularization parameter to 10 though in practice we found the model is less sensitive to this value. We run 100 iterations of EM for each setting. The dif-

[5] We remove the restriction at test time though we found it does not affect the performance.

[6] A word with one of the following POS tags: ADP, AUX, CONJ, DET, PART, and SCONJ.

ference of each model is then the type of constraints imposed during the E-step[7], or initialization:

- Baseline (FUNC): Function word constraints;
- HARM: FUNC with harmonic initialization;
- DEP: FUNC + stack depth constraints (Eq. 3);
- LEN: FUNC + soft dependency length bias, which we describe below.

For DEP, we use $\delta = 1.\xi$ to denote the relaxed maximum depth allowing span length up to ξ (Eq. 4).

LEN is the previously explored structural bias (Smith and Eisner, 2006), which penalizes longer dependencies by modifying each attachment score:

$$\theta'_{\text{A}}(a|h, dir) = \theta_{\text{A}}(a|h, dir) \cdot e^{-\gamma \cdot (|h-a|-1)}, \quad (5)$$

where γ (≥ 0) determines the strength of the bias and $|h-a|$ is (string) distance between h and a.

Note that DEP and LEN are closely related; generally center-embedded constructions are accompanied by longer dependencies so LEN also penalizes center-embedding implicitly. However, the opposite is not true and there exist many constructions with longer dependencies without center-embedding. By comparing these two settings, we discuss the worth of focusing on constraining center-embedding relative to the simpler bias on dependency length.

Finally we also add the system of Naseem et al. (2010) in our comparison. This system encodes many manually crafted rules between POS tags with the posterior regularization technique. For example, the model is encouraged to find NOUN → ADJ relationship. Our systems cannot access to these core grammatical rules so it is our strongest baseline.[8]

Constraining root word We also see the effects of the constraints when a small amount of grammatical rule is provided. In particular, we restrict the candidate root words of the sentence to a noun or a verb; similar rules have been encoded in past work such as Gimpel and Smith (2012) and the CCG induction system of Bisk and Hockenmaier (2013).

[7]We again remove the restrictions at decoding as we observed that the effects are very small.

[8]We encode the customized rules that follow UD scheme. The following 13 rules are used: ROOT → VERB, ROOT → NOUN, VERB → NOUN, VERB → ADV, VERB → VERB, VERB → AUX, NOUN → ADJ, NOUN → DET, NOUN → NUM, NOUN → NOUN, NOUN → CONJ, NOUN → ADP, ADJ → ADV.

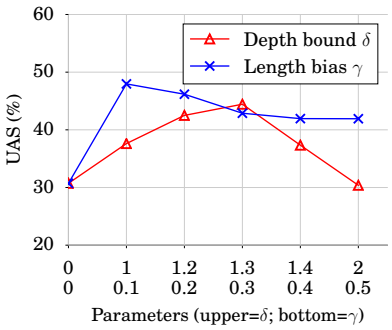


Figure 8: UAS for various settings on (UD) WSJ.

Hyperparameters Selecting hyperparameters in multilingual grammar induction is difficult; some works tune values for each language based on the development set (Smith and Eisner, 2006; Bisk et al., 2015), but this violates the assumption of unsupervised learning. We instead follow many works (Mareček and Žabokrtský, 2012; Naseem et al., 2010) and select the values with the English data. For this, we use the WSJ data, which we obtain in UD style from the Stanford CoreNLP (ver. 3.6.0).[9]

6 Experiments

WSJ Figure 8 shows the result on WSJ. Both DEP and LEN have one parameter: the maximum depth δ, and γ (Eq. 5), and the figure shows the sensitivity on them. Note that x-axis = 0 represents FUNC.

For LEN, we can see the optimal parameter γ is 0.1, and degrades the performance when increasing the value; i.e., the small bias is the best. For DEP, we find the best setting is 1.3, i.e., allowing embedded constituents of length 3 or less ($\xi = 3$ in Eq. 4). We can see that allowing depth 2 degrades the performance, indicating that depth 2 allows too many trees and does not reduce the search space effectively.[10]

Multilingual results Table 1 shows the main multilingual results. When we see "No root constraint" block, we notice that our DEP boosts the performance in many languages (e.g., Bulgarian, French,

[9]Note that the English data in UD is Web Treebank (Silveira et al., 2014), not the standard WSJ Penn treebank.

[10]We see the same effects when training with longer sentences (e.g., length ≤ 20). This is probably because a looser constraint does nothing for shorter sentences. In other words, the model can restrict the search space only for longer sentences, which are relatively small in the data.

	No root constraint				+ root constraint				
	FUNC	DEP	LEN	HARM	FUNC	DEP	LEN	HARM	N10
A-Greek	35.9	31.6	34.7	**37.8**	37.9	**45.0**	34.4	37.7	40.1
Arabic	48.6	38.7	**49.8**	42.8	45.9	44.3	**49.6**	31.4	37.8
Basque	41.7	**46.1**	45.0	24.9	42.5	44.8	**44.8**	25.3	50.1
Bulgarian	45.6	**69.0**	64.8	66.4	69.1	**71.1**	61.9	68.0	58.6
Croatian	40.8	32.2	**50.7**	47.8	40.7	42.2	47.6	**47.7**	41.0
Czech	56.0	**62.0**	52.7	53.7	47.2	**62.2**	56.0	52.2	52.0
Danish	42.5	42.7	42.3	**47.2**	42.6	42.8	42.3	**46.6**	42.8
Dutch	25.7	26.6	**28.0**	26.2	25.7	27.5	**28.7**	26.4	40.6
English	37.2	39.8	**52.1**	37.5	37.5	**40.0**	38.4	38.2	51.4
Estonian	68.5	67.4	68.0	**68.6**	68.0	67.8	65.1	**68.5**	67.3
Finnish	26.2	24.5	**27.9**	25.7	25.7	27.3	**27.9**	20.5	44.6
French	36.7	**48.0**	36.8	36.5	36.5	**54.6**	36.3	36.7	53.3
German	44.6	**48.0**	46.3	43.6	43.9	**50.4**	47.9	43.9	53.5
Hebrew	58.4	54.4	58.5	**59.1**	55.4	**59.7**	59.4	59.0	56.9
Hindi	54.7	52.6	16.0	**55.8**	**55.8**	52.6	48.8	55.7	55.8
Indonesian	36.0	**52.9**	45.6	40.1	30.4	**53.1**	40.5	40.0	51.1
Italian	63.8	67.8	**68.4**	65.0	63.1	65.7	**68.8**	62.9	56.3
Japanese	46.8	44.5	**73.8**	47.9	47.6	46.7	**72.3**	47.9	51.3
Latin-ITT	42.3	**43.8**	42.1	41.0	42.4	**43.7**	38.4	41.6	38.4
Norwegian	44.7	45.3	45.1	**51.9**	44.8	45.4	45.2	**45.7**	55.4
Persian	**44.9**	39.0	37.3	36.6	44.1	**46.6**	37.2	43.6	55.2
Portuguese	48.4	61.1	**61.6**	55.9	49.2	61.1	**61.4**	44.6	47.1
Slovenian	**65.6**	61.0	50.1	62.7	**65.1**	60.7	49.4	63.6	53.1
Spanish	52.2	54.6	**62.5**	49.1	44.4	53.8	**60.0**	48.4	55.3
Swedish	42.7	48.1	**51.4**	48.1	43.1	42.8	42.7	**47.6**	46.7
Avg	46.0	48.1	**48.5**	46.9	45.9	**50.1**	48.2	45.8	50.2

Table 1: Attachment scores on UD with or without root POS constraints. A-Greek = Ancient Greek. N10 = Naseem et al. (2010) with modified rules.

Indonesian, and Portuguese), though LEN performs equally well and in average, LEN performs slightly better. Harmonic initialization does not work well.

We then move on to the settings with the constraint on root tags. Interestingly, in these settings DEP performs the best. The model competes with Naseem et al.'s system in average, and outperforms it in many languages, e.g., Bulgarian, Czech, etc. LEN, on the other hand, decreases the average score.

Analysis Why does DEP perform well in particular with the restriction on root candidates? To shed light on this, we inspected the output parses of English with no root constraints, and found that the types of errors are very different across constraints.

Figure 9 shows a typical example of the difference. One difference between trees is in the constructions of phrase "On ... pictures". LEN predicts that "On the next two" comprises a constituent, which modifies "pictures" while DEP predicts that "the ... pictures" comprises a constituent, which is correct, although the head of the determiner is incorrectly predicted. On the other hand, LEN works well to find more primitive dependency arcs between POS tags, such as arcs from verbs to nouns, which are often incorrectly recognized by DEP.

These observations may partially answer the

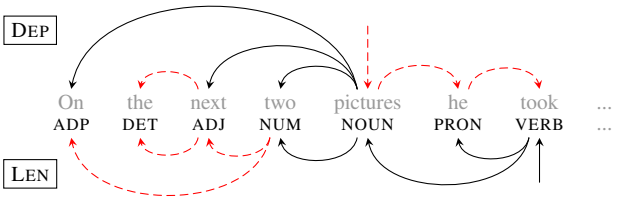


Figure 9: A comparison of output parses by DEP and LEN (with no root constraints). Dashed arcs are misclassified ones.

	Prec.	Recall	F1
FUNC (English)	11.6	18.4	14.1
DEP (English)	**22.4**	**37.1**	**27.9**
LEN (English)	21.6	31.0	25.5
FUNC (Avg.)	22.5	30.0	25.6
DEP (Avg.)	**27.8**	**34.5**	**30.5**
LEN (Avg.)	24.0	33.7	27.9
FUNC + ROOT (Avg.)	22.0	29.4	25.0
DEP + ROOT (Avg.)	**28.1**	**35.2**	**31.0**
LEN + ROOT (Avg.)	21.8	31.2	25.6

Table 2: Unlabeled bracket scores in various settings. Avg. is the average score across languages.

question above. The main source of improvements by DEP is detections of constituents, but this constraint itself does not help to resolve some core dependency relationships, e.g., arcs from verbs to nouns. The constraint on root POS tags is thus orthogonal to this approach, and it may help to find such core dependencies. On the other hand, the dependency length bias is the most effective to find basic dependency relationships between POS tags while the resulting tree may involve implausible constituents. Thus the effect of the length bias seems somewhat overlapped with the root POS constraints, which may be the reason why they do not well collaborate with each other.

Bracket scores We verify the above intuition quantitatively. To this end, we convert both the predicted and gold dependency trees into the unlabeled bracket structures, and then compare them on the standard PARSEVAL metrics. This bracket tree is not binarized; for example, we extract (X a b (X c d)) from the tree $a^\frown b^\frown c^\frown d$. Table 2 shows the results, and we can see that DEP always performs the best, showing that DEP leads to the models that find better constituent structures. Of particular note

	UAS	F1
DEP	48.1	**30.5**
LEN	48.5	27.9
DEP+LEN	**49.2**	27.0

Table 3: Average scores of DEP, LEN, and the combination.

is in Enlgish the bracket and dependency scores are only loosely correlated. In Table 1, UASs for FUNC, DEP, and LEN are 37.2, 39.8, and 52.1, respectively, though F1 of DEP is substantially higher. This suggests that DEP often finds more linguistically plausible structures even when the improvement in UAS is modest. We conjecture that this performance change between constraints essentially arise due to the nature of DEP, which eliminates center-embedding, i.e., implausible *constituent* structures, rather than dependency arcs.

Combining DEP and LEN These results suggest DEP and LEN capture different aspects of syntax. To furuther understand this difference, we now evaluate the models with *both* constraints. Table 3 shows the average scores across languages (without root constraints). Interestingly, the combination (DEP+LEN) performs the best in UAS while the worst in bracket F1. This indicates the ability of DEP to find good constituent boundaries is diminished by combining LEN. We feel the results are expected observing that center-embedded constructions are a special case of longer dependency constructions. In other words, LEN is a stronger constraint than DEP in that the structures penalized by DEP are only a subset of structures penalized by LEN. Thus when LEN and DEP are combined LEN overwhelms, and the advantage of DEP is weakened. This also suggests not penalizing all longer dependencies is important for learning accurate grammars. The improvement of UAS suggests there are also collaborative effects in some aspect.

7 Conclusion

We have shown that a syntactic constraint that eliminates center-embedding is helpful in dependency grammar induction. In particular, we found that our method facilitates to find linguistically correct constituent structures, and given an additional cue on dependency, the models compete with the system relying on a significant amount of prior linguistic knowledge. Future work includes applying our DEP constraint into other PCFG-based grammar induction tasks beyond dependency grammars. In particular, it would be fruitful to apply our idea into constituent structure induction for which, to our knowledge, there has been no successful PCFG-based learning algorithm. As discussed in de Marcken (1999) one reason for the failures of previous work is the lack of necessary syntactic biases, and our approach could be useful to alleviate this issue. Finally, though we have focused on unsupervised learning for simplicity, we believe our syntactic bias also leads to better learning in more practical scenarios, e.g., weakly supervised learning (Garrette et al., 2015).

Acknowledgements

We would like to thank John Pate for the help in preliminary work, as well as Taylor Berg-Kirkpatric for sharing his code. We are also grateful to Edson Miyamoto and Makoto Kanazawa for the valuable feedbacks. The first author was supported by JSPS KAKENHI Gran-in-Aid for JSPS Fellows (Grant Numbers 15J07986), and MOU Grant in National Institute of Informatics.

References

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590, Los Angeles, California, June. Association for Computational Linguistics.

Yonatan Bisk and Julia Hockenmaier. 2013. An hdp model for inducing combinatory categorial grammars. *Transactions of the Association for Computational Linguistics*, 1:75–88.

Yonatan Bisk, Christos Christodoulopoulos, and Julia Hockenmaier. 2015. Labeled grammar induction with minimal supervision. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 870–876, Beijing,China, July.

C. de Marcken. 1999. On the unsupervised induction of phrase-structure grammars. In Susan Armstrong,

Kenneth Church, Pierre Isabelle, Sandra Manzi, Evelyne Tzoukermann, and David Yarowsky, editors, *Natural Language Processing Using Very Large Corpora*, volume 11 of *Text, Speech and Language Technology*, pages 191–208. Springer Netherlands.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 457–464, Stroudsburg, PA, USA. Association for Computational Linguistics.

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.

Richard Futrell, Kyle Mahowald, and Edward Gibson. 2015. Large-scale evidence of dependency length minimization in 37 languages. *Proceedings of the National Academy of Sciences*, 112(33):10336–10341.

Dan Garrette, Chris Dyer, Jason Baldridge, and Noah Smith. 2015. Weakly-supervised grammar-informed bayesian ccg parser learning.

E. Gibson. 2000. The dependency locality theory: A distance-based theory of linguistic complexity. In *Image, language, brain: Papers from the first mind articulation project symposium*, pages 95–126.

Daniel Gildea and David Temperley. 2010. Do grammars minimize dependency length? *Cognitive Science*, 34(2):286–310.

Kevin Gimpel and Noah A. Smith. 2012. Concavity and initialization for unsupervised dependency parsing. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 577–581, Montréal, Canada, June. Association for Computational Linguistics.

Joseph H. Greenberg. 1963. Some universals of grammar with particular reference to the order of meaningful elements. In Joseph H. Greenberg, editor, *Universals of Human Language*, pages 73–113. MIT Press, Cambridge, Mass.

John A Hawkins. 2014. *Cross-linguistic variatoin and efficiency*. Oxford University Press, jan.

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. Association for Computational Linguistics.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, Tartu, Estonia, May.

Mark Johnson. 1998. Finite-state approximation of constraint-based grammars using left-corner grammar transforms. In Christian Boitet and Pete Whitelock, editors, *COLING-ACL*, pages 619–623. Morgan Kaufmann Publishers / ACL.

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.

Fred Karlsson. 2007. Constraints on multiple center-embedding of clauses. *Journal of Linguistics*, 43(2):365–392.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 478–485, Barcelona, Spain, July.

David Mareček and Zdeněk Žabokrtský. 2012. Exploiting reducibility in unsupervised dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 297–307, Jeju Island, Korea, July. Association for Computational Linguistics.

George A. Miller and Noam Chomsky. 1963. Finitary models of language users. In D. Luce, editor, *Handbook of Mathematical Psychology*, pages 2–419. John Wiley & Sons.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244, Cambridge, MA, October. Association for Computational Linguistics.

Hiroshi Noji and Yusuke Miyao. 2014. Left-corner transitions on dependency parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2140–2150, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Hiroshi Noji. 2016. *Left-corner Methods for Syntactic Modeling with Universal Structural Constraints*. Ph.D. thesis, Graduate University for Advanced Studies, Tokyo, Japan, March.

Philip Resnik. 1992. Left-corner parsing and psychological plausibility. In *COLING*, pages 191–197.

D.J. Rosenkrantz and P.M. Lewis. 1970. Deterministic left corner parsing. In *Switching and Automata Theory, 1970., IEEE Conference Record of 11th Annual Symposium on*, pages 139–152, Oct.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2010. Broad-coverage parsing using human-like memory constraints. *Computational Linguistics*, 36(1):1–30.

Roy Schwartz, Omri Abend, Roi Reichart, and Ari Rappoport. 2011. Neutralizing linguistically problematic annotations in unsupervised dependency parsing evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 663–672, Portland, Oregon, USA, June. Association for Computational Linguistics.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Noah A. Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of the International Conference on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL)*, pages 569–576, Sydney, July.

Noah A. Smith. 2006. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. Ph.D. thesis, Johns Hopkins University, Baltimore, MD, October.

Marten van Schijndel and William Schuler. 2013. An analysis of frequency- and memory-based processing costs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 95–105, Atlanta, Georgia, June. Association for Computational Linguistics.