Collins-LA: Collins' Head-Driven Model with Latent Annotation

Seung-Hoon Na[1] , Meixun Jin[1], In-Su Kang[2] and Jong-Hyeok Lee[1]

[1] Pohang University of Science and Technology (POSTECH), AITrc, Republic of Korea
{nsh1979,meixunj,jhlee}@postech.ac.kr,
[2] Korea Institute of Science and Technology Information (KISTI), Republic of Korea
dbaisk@kisti.re.kr

Abstract. Recent works on parsing have reported that the lexicalization does not have a serious role for parsing accuracy. Latent-annotation methods such as PCFG-LA are one of the most promising un-lexicalized approaches, and reached the-state-of-art performance. However, most works on latent annotation have investigated only PCFG formalism, without considering the Collins' popular head-driven model, though it is a significantly important and interesting issue. To this end, this paper develops Collins-LA, the extension of the Collins' head-driven model to support the latent annotation. We report its basic accuracy, comparing with PCFG-LA. The experimental results show that Collins-LA has potential to improve basic parsing accuracy, resulting in comparable performance with PCFG-LA even in the naive setting.

1 Introduction

Recent works for parsing have consistently shown that the lexicalization does not have serious effects on parsing accuracy. Gildea mentioned that the high performance of a Collins' model is obtained not from the bi-lexical dependency, showing that parsing accuracy is not decreased even when the bi-lexical dependency is not incorporated to the model [1]. Gildea's result has been re-confirmed by Bikel, during his investigation through the re-implementation of the Collins' parsing model [2].

Another direction is opened from a Klein's work, where fully un-lexicalized parsing models are extensively evaluated through an accurate design of tree-bank annotation [3]. Klein's work includes Johnson's parent annotation [4], and external-internal annotation, tag-splitting, and head annotation, etc, resulting in the parsing accuracy of about 86%, which corresponds to the initial performance of the lexicalized parsing accuracy. Motivated from this, Matsuzaki proposed a new generative model PCFG-LA, an extension of PCFG models in which non-terminal symbols are annotated with latent variables [5]. PCFG-LA successfully replaces the manual feature selection used in previous research such as Johnson's work or Klein's work, by showing that PCFG-LA reaches Klein's result. Based on this, Petrov et al. further explored PCFG-LA where hierarchical splitting and merging of latent non-terminals are utilized, thus differentiating the number of latent variables of each non-terminal symbol [6]. Their result is remarkable, showing about 90% accuracy, which is almost comparable to the-state-of-art of Charniak's parser [7]. All these previous results consistently show

Advances in Natural Language Processing and Applications
Research in Computing Science 33, 2008, pp. 43-54

Received 15/10/07
Accepted 07/12/07
Final Version 19/01/08

that the annotation-based parsing strategy, especially the recently developed automatic annotation, is a promising approach which can adequately balance between the naiveness of the original parsing model and the sparseness problem of the complicated lexicalized model.

However, most previous works on the annotation are restricted to PCFG, without investigating other probabilistic parsing models such as a head-driven model, etc. Of course, PCFG has many advantages such as simplicity and clearness to be applied to the parsing problem, but there are many other parsing models which are known as the-state-of-art methods. In this regard, this paper investigates the extension of the popular Collins' head-driven model [8] with the latent annotation, namely *Collins-LA*. The Collins' model deserves to be extended with the latent annotation for following reasons. 1) Collins' model is one of the-state-of-art approaches for parsing, as well as the Charniak's model. 2) Gildea's observation that bi-lexicalization does not play a serous role in the high performance of parsing is revisited through Collins' head-driven parsing model. 3) Previous works on PCFG have adopted the Collins' head-driven idea when binarizing production rules collected from the tree-bank, by using head-driven binarization according to the head-modifier generation principle (e.g. Matsuzaki's center-head and center-parent binarization, Petrov's x-bar binarization). Although their usage of the head-driven approach is limited to the style of merely applying the head-modifier rule, not incorporated with the parsing model, the head-driven idea has been widely accepted in the parsing community.

Our goal is to examine whether or not the Collins' model is accepted even for the un-lexicalized model. To this end, we formulate a generation model based on Collin-LA, and derive an EM training algorithm to estimate Collins-LA. To cover the specialized generation for BaseNP, we derive a new EM algorithm using the inside-outside algorithm embedding a forward-backward algorithm, and evaluate it on the standard Penn tree-bank data set. Experimental results show that Collins-LA marginally improves the original Collins' model, and is comparable to Matsuzaki's PCFG-LA. This result is notable in that the Collins' head-driven model works well even within the extension of latent annotation. Further, it implies that the work on the latent annotation should be further separately investigated for other grammars.

Relating to our work, Prescher proposed Head-LA, the latent annotation framework of a head-driven model [9]. However, Prescher's work refers to Charniak's head-driven parsing model [10], not to Collins' parsing model. Collins' head-driven model is different from Chaniak's one, since Chaniak's model involves the production rule within the generative model, while Collins' model does not include the production rule itself. Therefore, Prescher's head-LA cannot be directly applied to Collins' head-driven model, thus we require a new separate algorithm to support the latent annotation for Collins' model.

This paper is organized as follows. First, Section 2 briefly mentions Collins' head-driven model. Section 3 examines the proposed Collins-LA, and a EM algorithm to train it. Section 4 further extends Collins-LA for dealing with a specialized process of BaseNP, and drives a novel EM algorithm where a forward-backward algorithm is embedded to the inside-outside algorithm. Section 5 presents experimental results. Finally, the conclusion will be given in Section 6.

2 Un-lexicalized Collins' Head-Driven Model

In the principle of Collins' model, the head node is first generated for a given parent node, and other modifier nodes are generated from the head node and the parent node. Formally, suppose that the production rule is given by r as follows.

$$r : P \rightarrow L_N \Lambda\ L_1 H R_1 \Lambda\ R_M$$

, where H indicates the head-node of parent node P and, L_i and R_j are left and right child nodes. The generation of Collins' model consists of two separated processes [8]: 1) generation of head-node H from the given parent node P, 2) generation of left and right child nodes L_i or R_j from parent-node P and head-node H. This is formulated as follows.

$$P(r) = P(H \mid P)\prod_i P(L_i \mid P, H)\prod_j P(R_j \mid P, H) \quad \textbf{(1)}$$

However, in this setting, the generation process will not be stopped, and would prefer the production rule with a less number of child nodes. Collins introduced *STOP* node, indicating the stop event on head-modifier generations, and defined the following model.

$$P(r) = P(H \mid P)\prod_i P(L_i \mid P, H)\prod_j P(R_j \mid P, H) \quad \textbf{(2)}$$
$$P_L(STOP \mid P, H)P_R(STOP \mid P, H)$$

where $P_L(STOP|P,H)$ and $P_R(STOP|P,H)$ indicate the probability that generates *STOP* event for the left-child part and the right-child part, respectively. This *STOP*-added model can partially deal with the preference to production rule with the small number of child nodes. In fact, when L_{N+1} and R_{M+1} are inserted to the rule r as a STOP symbol, Eq. (2) is equivalent to Eq. (1). Summing up, Collins' model consists of the following four types of probabilities.

1) Head generation: $P(H|P)$
2) Left modifier generation: $P(L_i|P,H)$
3) Right modifier generation: $P(R_j|P,H)$
4) Lexical generation: $P(w|H)$

For a given parse tree T, the generation probability of T is formulated as follows:

$$P(T) = \prod_{r \in T} P(r)^{c(r;T)} \quad \textbf{(3)}$$

where $c(r;T)$ is the number of counts that applies the rule r in tree T. Parsing is the work to find the best parse tree that maximizes the above generation probability $P(T)$. The above model is the un-lexicalized version of Collins' head-driven. The final version of Collin's model is fully lexicalized where the head-word and the head-tag are annotated to each non-terminal node in the original tree T, using the same head-modifier generation principle.

3 Latent Annotation of Collin's Head-Driven Model

3.1 Brief Background on Tree-Bank Annotation Approaches

Annotation-driven approaches do not use a given tree-bank directly. Instead, it converts the tree-bank into the annotated one by annotating non-terminal symbols with another non-terminal symbol in the tree or an automatically generated symbol. In fact, lexicalization used in the head-driven model belongs to annotation-driven approaches, where the head word and the head tag are attached to each non-terminal node.

The initial work for the annotation is Johnson's parent annotation, where a non-terminal node is annotated with the symbol of its parent node [4]. In spite of the simplicity of Johnson's annotation, PCFG using the annotated tree-bank reaches about 79% parsing accuracy which significantly improves the baseline of 72%. Klein's work further explored Johnson's annotation scheme, by investigating various annotation schemes such as tag split, attachment of pre-nonterminal and BaseNP, reaching the parsing accuracy of about 86% [4]. However, Johnson's and Klein's works applied manually designed annotation, i.e. considering "which non-terminal symbol of nodes should be split?", and "how the symbol should be annotated with another symbol?". To avoid the manual design of annotation, Matsuzaki et al. [5] and Petrov et al. [6] applied the latent annotation where such considerations are automatically decided.

3.2 Collins-LA: Latent Extension of Collins Model

This study extends the original Collins' model to Collins-LA by annotating latent variables to non-terminal symbols. Let k be the number of latent variables. For given non-terminal symbol P, there are k possible latent variables for P - $P[1]$, …, $P[L]$. Given parse tree T, $\varphi(T)$ is a *latent-annotated tree* obtained from T where each non-terminal node is mapped to one of latent variables. Then, we define the generation probability of $\varphi(T)$ as follows.

$$P(\phi(T)) = \prod_{r \in \phi(T)} P(r)^{c(r;\phi(T))} \quad \textbf{(4)}$$

The production rule r for generating $\varphi(T)$ takes the following form.

$$r : P[x] \rightarrow L_{N+1} L_N[z_N] \Lambda\ L_1[z_1] H[y] R_1[w_1] \Lambda\ R_M[w_N] R_{M+1} \quad \textbf{(5)}$$

where x, y, z_i and w_j correspond to the index numbers used for latent variables in $\varphi(T)$: P, H, L_i, and R_j, respectively. L_{N+1} and R_{M+1} indicates *STOP* symbols.

From the totality law of the probability, $P(T)$ is obtained from the summation of generative probabilities of all latent-annotated trees as follows.

$$P(T) = \sum_{\varphi(T) \in \Phi(T)} P(\varphi(T)) \quad \textbf{(6)}$$

where $\Phi(T)$ is the set of all possible latent annotated trees which can be obtained from T. $|\Phi(T)|$ is the number of the set, which follows the exponential function of the num-

ber of non-terminal nodes in T. Similar to the original Collins' model, we should estimate the following four types of probabilities.

1) Head generation: $P(H[y]|P[x])$
2) Left modifier: $P(L_i[z]|P[x],H[y])$
3) Right modifier: $P(R_j[z]|P[x],H[y])$
4) Lexical generation: $P(w|H[x])$

where x, y, and z are the index numbers of latent variables – P, H and L_i (or R_j)

3.3 EM Training Algorithm

Given the training set of parse trees $Train = \{T_1, T_2, ..., T_U\}$, we derive an EM-training algorithm similar to the inside-outside algorithm to estimate the above four types of probabilities for Collins-LA.

Let $\beta(P[x])$ be the inside-probability which generates the sub-tree with P as a root node, providing that P is assigned to the latent index number x. Then, $\beta(P[x])$ can be defined according to the following three different cases.

1) P is a pre-nonterminal node that contains w_i as a unique child word.

$$\beta(P[x]) = P(w_i \mid P[x]) \tag{7}$$

2) For the stop symbol STOP, $\beta(STOP[z])$ is 1.0.
3) Otherwise, let H be the head child node of P, and L_i and R_j be i-th left modifier and j-th right modifier, respectively. Then, $\beta(P[x])$ is recursively defined as follows.

$$\beta(P[x]) = \sum_y P(H[y] \mid P[x])\beta(H[y]) \tag{8}$$
$$\prod_i \gamma^i_{Left}(P[x], H[y]) \prod_j \gamma^j_{Right}(P[x], H[y])$$

where $\gamma_{Left}^{i}(P[x],H[y])$ is the probability that generates the sub-tree with L_i as a root node (including the generation of L_i) as follows.

$$\gamma^i_{Left}(P[x], H[y]) = \sum_z P(L_i[z] \mid P[x], H[y])\beta(L_i[z]) \tag{9}$$

Note that $\gamma_{Left}^{i}(P[x],H[y])$ contains the generation probability for sub-trees with roots as $L_i[1]$ … $L_i[k]$ of each latent node for node L_i, thus it is independent to the index number of the latent variable. Similar to $\gamma_{Left}^{i}(P[x],H[y])$, $\gamma_{Righj}^{i}(P[x],H[y])$ is defined as the probability that generates the sub-tree with R_j as a root node, for given $P[x]$ and $H[y]$ (including the generation of the root node R_j).

$$\gamma^j_{Right}(P[x], H[y]) = \sum_z P(R_j[z] \mid P[x], H_i[y])\beta(R_j[z]) \tag{10}$$

Let $\alpha(P[x])$ be the outside probability that generates all other sub-trees in T, except for the generation of the subtree of $P[x]$ (but including the generation of $P[x]$). For parent node P, let $\alpha(H[y])$, $\alpha(L_i[z])$ and $\alpha(R_j[z])$ be outside probabilities of the head-node H, and child nodes L_i and R_j, respectively. Then, all these outside probabilities are recursively defined using $\alpha(P[x])$, in a style of the top-down form.

$$\alpha(H[y]) = \sum_x \alpha(P[x])P(H[y] \mid P[x]) \prod_i \gamma^i_{Left}(P[x], H[y]) \prod_j \gamma^j_{Right}(P[x], H[y]) \quad (11)$$

For i-th left child node L_i,

$$\alpha(L_i[z]) = \sum_x \sum_y \alpha(P[x])P(H[y] \mid P[x])\beta(H[y]) P(L_i[z] \mid P[x], H[y]) \left(\prod_{k \neq i} \gamma^k_{Left}(P[x], H[y]) \right) \prod_j \gamma^j_{Right}(P[x], H[y]) \quad (12)$$

For j-th right child node R_j,

$$\alpha(R_j[z]) = \sum_x \sum_y \alpha(P[x])P(H[y] \mid P[x])\beta(H[y]) P(R_j[z] \mid P[x], H[y]) \left(\prod_{k \neq i} \gamma^k_{Left}(P[x], H[y]) \right) \prod_j \gamma^j_{Right}(P[x], H[y]) \quad (13)$$

Based on inside and outside probabilities of $\alpha(P[x])$ and $\beta(P[x])$, the Expectation step in EM algorithm calculates the expected count of events which correspond to four type of rules.

$$c(P[x], w \mid T) \propto \alpha(P[x])P(w \mid P[x]) \quad (14)$$

$$c(P[x], H[y] \mid T) \propto \alpha(P[x])P(H[y] \mid P[x])\beta(H[y]) \prod_i \gamma^i_{Left}(P[x], H[y]) \prod_j \gamma^j_{Right}(P[x], H[y])$$

$$c(P[x], H[y], L_i[z] \mid T) \propto \alpha(P[x])P(H[y] \mid P[x])\beta(H[y]) \prod_{k \neq i} \gamma^k_{Left}(P[x], H[y]) \prod_j \gamma^j_{Right}(P[x], H[y]) P(L_i[z] \mid P[x], H[y])\beta(L_i[z])$$

(Parse tree T should be inserted as a conditional term in the right-hand part in Eq. (14), but we skip.) Now in the Maximization step, the four types of probabilities which we pursue are calculated as follows, using the above expected counts.

$$P(w \mid P[x]) \propto \sum_{T \in Train} P(T)c(P[x], w \mid T) \quad (15)$$

$$P(H[y] \mid P[x]) \propto \sum_{T \in Train} P(T)c(P[x], H[y] \mid T)$$

$$P(L_i[z] \mid P[x], H[y]) \propto \sum_{T \in Train} P(T)c(P[x], H[y], L_i[z] \mid T)$$

where *Train* is the training set of parse trees in a tree-bank. *P(T)* can be rewritten by using these inside and outside probabilities.

$$P(T) = \sum_{z} \alpha(X[z] \mid T)\beta(X[z] \mid T) \tag{16}$$

where X is a non-terminal symbol in parse tree T.

3.4 Decoding

The decoding problem (i.e. parsing problem) is to find the best parse tree that maximizes $P(T)$. However, the exact decoding problem is NP-complete [5]. Instead, we adopted n-best re-ranking strategy, which is one of the approximation methods taken by Matsuzaki et al [5]. In n-best re-ranking, we first generate top n candidate parse trees using a non-latent original parsing model (PCFG or other probabilistic grammar), and then re-rank them according to the generation probabilities $P(T)$ from Collins-LA.

4 Further Extension for BaseNP Generation

4.1 Specialized Generation Model for BaseNP

BaseNP is a non-recursive noun phrase which does not contain any other noun phrases as its descendents. Due to a linguistic reason, Collins dealt with BaseNP, using a different generation model from the generation of other non-terminals in his original head-driven model. While other non-terminals are generated from parent node and head child node (i.e. $P(L_i|P,H)$), BaseNP is generated from parent node and previous left modifier ($P(L_i|P,L_{i-1})$). To discuss this specialized generation in more details, let us suppose that BaseNP P is generated by production rule r:

$$r : P \rightarrow L_N \Lambda\ L_1 H$$

Then, the generation model of BaseNP is given as follows.

$$P(r) = P(H \mid P)\prod_{i} P(L_i \mid P, L_{i-1}) \tag{17}$$

where L_0 indicates H.

4.2 New EM: Inside-Outside Algorithm Embedding Forward Backward Algorithm

The specialized dependency for generating BaseNP is characterized to Markov dependency of Hidden Markov Model (HMM). Thus, a forward-backward algorithm for learning HMM should be embedded into the original inside-outside algorithm. For BaseNP, inside-outside probabilities are further reformulated as follows.

Inside Probability. First, the inside probability is reformulated as follows.

$$\beta(P[x]) = \sum_{y} P(H[y] \mid P[x])\beta(H[y])f(P[x], L_0[y]) \quad (18)$$

Again, we regard L_0 as H. $f(P[x],L_i[y])$ indicates the forward probability, which is the probability that generates all next left modifiers for given $P[x]$ and $L_i[y]$. This is recursively defined as follows.

$$f(P[x], L_i[y]) = \sum_{z} P(L_{i+1}[z] \mid P[x], L_i[y])\beta(L_{i+1}[z])f(P[x], L_{i+1}[z]) \quad (19)$$
$$\text{if } i < Ml$$
$$= \sum_{z} P(STOP[z] \mid P[x], L_i[y]) \quad \text{otherwise}$$

Outside Probability.

Outside probabilities are reformulated case-by-case as follows. For head-child node H,

$$\alpha(H[y]) = \sum_{x} \alpha(P[x])P(H[y] \mid P[x])f(P[x], H[y])b(P[x], H[y]) \quad (20)$$

For i-th left-child node L_i,

$$\alpha(L_i[y]) = \sum_{x}\sum_{y} \alpha(P[x])P(H[y] \mid P[x])f(P[x], L_i[y])b(P[x], L_i[y]) \quad (21)$$

, where $b(P[x],L_i[y])$ indicates the backward probability which is the probability that i-th left child node is generated with $L_i[y]$ for given $P[x]$. The backward probability is defined as the summation of probabilities of all possible right stop events.

$$b(P[x], L_i[z]) = \sum_{y} \beta(L_{i-1}[y])b(P[x], L_{i-1}[y])P(L_i[z] \mid P[x], L_{i-1}[y]) \quad (22)$$

As a special case, the backward probability for head child node - $b(P[x],H[y])$ is defined as the summation of probabilities of all possible right stop events.

$$b(P[x], H[y]) = \sum_{z} P(STOP[z] \mid P[x], H[y]) \quad (23)$$

EM Algorithm of the Specialized Collins-LA for BaseNP.

In the Expectation step, the expected counts are calculated as follows.

$$c(P[x], H[y] \mid T) \propto \alpha(P[x])P(H[y] \mid P[x])\beta(H[y]) \quad (24)$$
$$f(P[x], H[y])b(P[x], H[y])$$

$$c(P[x], L_{i-1}[z], L_i[w] \mid T) \propto \sum_{y} \alpha(P[x])P(H[y] \mid P[x])\beta(H[y])$$
$$f(P[x], L_i[w])b(P[x], L_{i-1}[z])\beta(L_{i-1}[z])\beta(L_{i-1}[w])$$
$$P(L_i[w] \mid P[x], L_{i-1}[z])$$

In the Maximization step, Eq. (15) is used to estimate each type of probability.

5 Experimentation

5.1 Experimental Setting

We used WSJ sections (1 ~ 21) as a training data, and the first 420 sentences in WSJ section 22 as a test set. Through our preliminary experiments, we found that the parsing accuracy between the test set and the standard test set (using WSJ section 23) does not show a difference of more than 0.5%.

5.2 Generating N-best Candidate Parse Trees

Table 1. Oracle Performances of *N*-best Parse Trees

	N = 1	*N = 50*	*N = 100*	*N = 200*	*N = 300*	*N = 500*
PCFG	70.25	85.67	87.24	88.78	89.1	89.66
Klein's Markovization (v=2, h=1)	77.57	89.68	90.93	92.02	92.45	92.82

For generating *N*-best candidate parse trees, we adopted Klein's vertical and horizontal markovization [3]. We simply call it *Klein's markovization*. As for vertical markovization level (v) and horizontal level (h), we selected as $v = 2$ and $h = 2$. Table 1 shows the oracle performance for *N*-best parse trees of Klein's markovization for different *N*s *(1, 50, 100, 200, 300,* and *500),* compared with the result of PCFG. Klein's markovization showed a large difference from PCFG's result when *N* is 1. As *N* is larger, the performance difference becomes smaller.

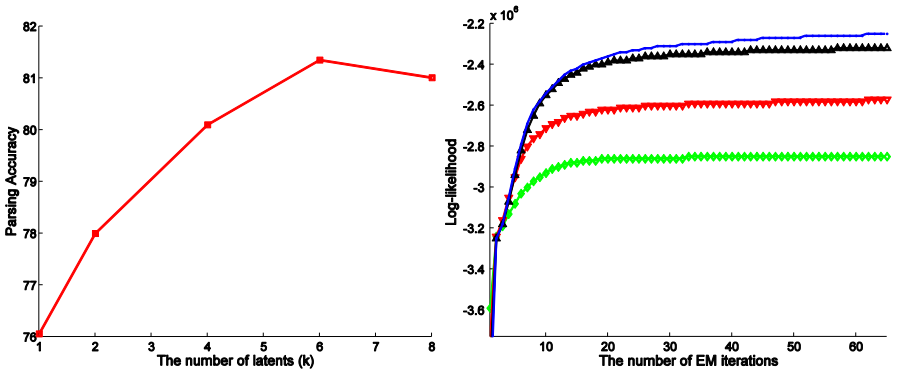
5.3 Performance of Collins-LA: Effects of the Number of Latent Variables

We set *N* to 200, thus generated top-200 candidate parse trees and re-ranked them according to the generation probability of Collins-LA (BaseNP version in section 4). Figure 1 shows the parsing accuracy of Collins-LA, and the log-likelihood curve on the number of EM iterations, for different *k*s (1, 2, 4, 6 and 8). As shown in Figure 1, the parsing accuracy increases as *k* is larger, but decreases when *k* is more than 6. This result is probably caused by the data sparseness problem that seriously arises when *k* is 8. Log-likelihood at the final EM iterations is in proportional to *k*.

5.4 Comparison with Matsuzaki's PCFG-LA

To compare Collins-LA with the performance of Matsuzaki's PCFG-LA, we re-implemented PCFG-LA. The full binarization of PCFG obtained from the tree-bank is not desirable since a serious data sparseness problem is involved. To this end, we used a light backbone grammar based on the binarization with center-head technique which Matsuzaki used [5], and further with Klein's vertical and horizontal markovizations [3]. To see this binarization scheme in more detail, let us consider the original production rule(Figure 2).

Then, figure 3 shows the types of rules of our backbone grammar obtained by binarizing the original rule of figure 2.



(a) Parsing accuracy (b) Log-likelihood curve

Figure 1. Parsing accuracy of Collins-LA for different *k*s, and log-likelihood curve.

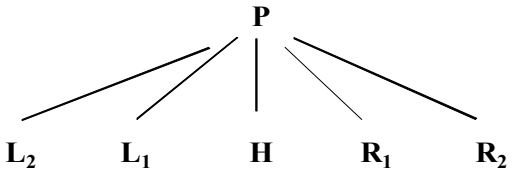
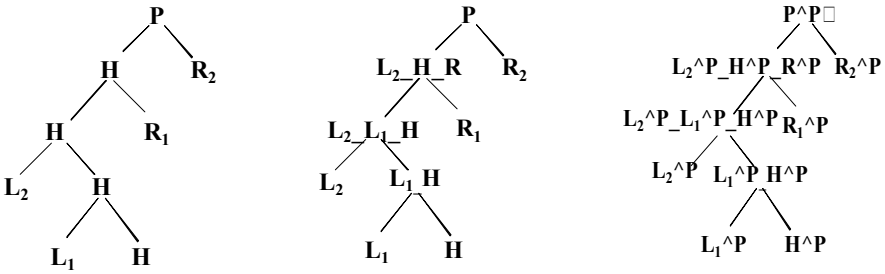


Figure 2. A given production rule.



a) v = 1, h = 0 b) v = 1, h = 2 c) v = 2, h = 2

Figure 3. Three center-head binarizations are presented using a) Klein's markovization, b) Klein's markovization with more horizontal markovization, and c) Klein's markovizatoin with Johnson's parent annotation. We used b) and c) for evaluating PCFG-LA.

Through several experiments, we found that PCFG-LA becomes the best when h is 2. If h is more than 2, then the parsing accuracy decreased. Thus, we selected this horizontal option (i.e. h =2). We optimized the number of latent variables for each nonterminal node. We assigned the number for latent variables differently according to their relative frequencies in the training corpus. During this, we assumed that the maximum number of latent variables is 8. Once our assigned the number, the nonterminal symbols related to noun phrases and verb phrases have 8 different latent variables, due to their large relative frequencies, while a coordination symbol (CC) has only one latent variable. For generating top N parse trees, we used the same grammar (Klein's markovization) described in section 5.2.

Table 2 summarizes the best performances of Collins-LA (k = 6) and PCFG-LA (k = 8 in maximum).

Table 2. The final parsing accuracy of Collins-LA and PCFG-LA.

	All length			Length ≤ 40		
	LP	LR	F1	LP	LR	F1
Baseline	77.78	77.35	77.57	79.21	78.66	78.9
Collins-LA	80.59	79.63	80.11	82.02	80.68	81.35
PCFG-LA(v=1,h=2)	81.58	80.07	80.82	83.23	81.43	82.32
PCFG-LA(v=2,h=2)	82.62	81.46	**82.04**	84.29	82.86	**83.57**

As shown in Table 2, Collins-LA is comparable to PCFG-LA (v=1, h=2), without showing much difference. Our result of PCFG-LA is less-performed than the result of Matsuzaki's one. The reason for this is that the initial quality of top-N parse trees is different. In Matsuzaki's case, top-1 parsing accuracy of the initial parse trees reaches about 80% performance, which is much more superior to about 77% of our top-1 parse tree. We further evaluated PCFG-LA using the backbone grammar based on the binarization with Johnson's parent annotation (v=2, h=2). Clearly, PCFG-LA with parent annotation (v=2, h=2) is superior to Collins-LA and PCFG-LA (v=1, h=2). Note that PCFG-LA with parent annotation uses a more complex backbone grammar, but Collins-LA uses simple PCFG grammar of (v=1, h=0). The grammar used in Collins-LA is more simple than even PCFG-LA (v=1, h=2). Regarding this, the current result of Collins-LA can be more improved when adopting the elaborated backbone grammar. Although some additional experiments are necessary, this experiment identifies that the formalism of Collins' head-driven model provides a useful framework as PCFG does for latent annotation. This is a sufficiently notable result in the way of the research of latent annotation.

6 Conclusion

This paper proposed the extension of Collins' head-driven model with the latent annotation, namely Collins-LA. We provided a novel training algorithm of Collins-LA, based on an inside-outside algorithm, and further extended it to consider the special consideration for BaseNP by embedding a forward-backward algorithm within the inside-outside algorithm. Experimental results are inspiring, showing that Collins-LA is comparable to PCFG-LA which is equipped with a more elaborated backbone

grammar. Regarding that our current setting of Collins-LA is at an initial stage, its performances could be much improved if more accurate grammars and formalizations are adopted. The work for improving and extending Collins' head-driven model is important since it can be more flexibly applied to non-English languages than PCFG. In the future, we will continue to explore Collins-LA on a more complicated backbone grammar, and use a more elaborated linguistic considerations, without losing the elegance and principle of the original model.

Acknowledgements. This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center (AITrc), also in part by the BK 21 Project and MIC & IITA through IT Leading R&D Support Project in 2007.

References

1. Gildea, D.: Corpus variation and parser performance. In: EMNLP '01. (2001), 167–172
2. Bikel, D.M.: Intricacies of collins' parsing model. Computational Linguististics 30(4) (2004) 479–511
3. Klein, D., Manning, C.D.: Accurate unlexicalized parsing. In: ACL '03. (2003) 423–430
4. Johnson, M.: PCFG models of linguistic tree representations. Computational Linguistics 24(4) (1998) 613–632
5. Matsuzaki, T., Miyao, Y., Tsujii, J.: Probabilistic CFG with latent annotations. In: ACL '05. (2005) 75–82
6. Petrov, S., Barrett, L., Thibaus, R., Klein, D.: Learning accurate, compact, and interpretable tree annotation. In: COLING-ACL '06. (2006) 433–440
7. Charniak, E., Johnson, M.: Coarse-to-fine n-best parsing and maxent discriminative reranking. In: ACL 2005. (2005) 173–180
8. Collins, M.: Head-driven statistical models for natural language parsing. Computational Linguististics 29(4) (2003) 589–637
9. Prescher, D.: Head-driven PCFGs with latent-head statistics. In: IWPT '05. (2005) 115–124
10. Charniak, E.: Statistical parsing with a context-free grammar and word statistics. In: AAAI/IAAI 1997. (2005) 598–603