Variational Bayesian Grammar Induction for Natural Language

Kenichi Kurihara and Taisuke Sato

Tokyo Institute of Technology, Tokyo, Japan,
{kurihara,sato}@mi.cs.titech.ac.jp

Abstract. This paper presents a new grammar induction algorithm for probabilistic context-free grammars (PCFGs). There is an approach to PCFG induction that is based on parameter estimation. Following this approach, we apply the variational Bayes to PCFGs. The variational Bayes (VB) is an approximation of Bayesian learning. It has been empirically shown that VB is less likely to cause overfitting. Moreover, the free energy of VB has been successfully used in model selection. Our algorithm can be seen as a generalization of PCFG induction algorithms proposed before. In the experiments, we empirically show that induced grammars achieve better parsing results than those of other PCFG induction algorithms. Based on the better parsing results, we give examples of recursive grammatical structures found by the proposed algorithm.

1 Introduction

Grammar induction is one of the most challenging tasks in natural language processing as Chomsky's "the poverty of the stimulus" says. Nonetheless, applications have already been proposed. For example, van Zaanen [18] applied grammar induction to build treebanks. Bockhorst and Craven [3] improved models of RNA sequences using grammar induction.

There is one practical approach to induce context-free grammars in natural language processing, which exploits parameter estimation of probabilistic context-free grammars (PCFGs) [13, 15, 4, 7]. Although this approach is not optimal, the empirical results showed good performance, e.g. over 90 % bracketing accuracy on the Wall Street Journal [15, 7]. They also utilize bracketed sentences. Brackets explicitly indicate the boundaries of constituents. One may criticize using brackets because making brackets have been expensive in terms of time and cost. However, unsupervised induction algorithms have been proposed to annotate brackets [8, 9].

This paper presents a variational Bayesian PCFG induction algorithm. Parameter-estimation-based induction has mainly two procedures. One is to estimate parameters, and the other is to choose a better grammatical structure based on a criterion. In previous work, parameter estimation is done with the Inside-Outside algorithm [2], which is an EM algorithm for PCFGs, and grammars are chosen by an approximate Bayesian posterior probability [16, 4]. Our algorithm can be seen as a Bayesian extension of parameter-estimation-based grammar induction. Moreover, our criterion to choose a grammar generalizes the approximate Bayesian posterior.

We experimentally show that our algorithm achieves better parsing results than other PCFG induction algorithms. One may be afraid that rule-based grammars do not parse some sentences due to the lack of rules. We propose an engineering approach to achieve 100% parsing coverage based on semi-supervised clustering that mixes labeled and unlabeled data [19]. As the better parsing results supports that induced grammars are well-organized, we give the examples of grammatically meaningful structures in induced grammars.

2 Parameter-Estimation-Based Grammar Induction

Since Lari and Young [11] empirically showed the possibility of statistical induction of PCFGs using the Inside-Outside algorithm [2], parameter-estimation-based grammar induction has received a great deal of attention. Largely speaking, there are two main issues. One is efficiency, and the other is a criterion to choose a grammatical structure.

In terms of efficiency, Pereira and Schabes [13] extended the Inside-Outside algorithm to take advantage of constituent bracketing information in a training corpus. Brackets help the algorithm improve parameter estimation and efficiency. While Pereira and Schabes [13] fixed the number of non-terminals, an incremental grammar induction algorithm has been proposed for more efficiency [7].

Grammar induction can be seen as a search problem whose search space is possible grammars. From this viewpoint, a criterion to choose a grammatical structure plays a critical role. Stolcke and Omohundro [16], Chen [4] use Bayesian posterior probabilities as a criterion where the posterior is approximated by Viterbi parses. This approximation is a special case of the variational Bayes.

Our algorithm exploits the variational Bayes. Therefore, our algorithm is a Bayesian extension of an efficient algorithm proposed by Hogenhout and Matsumoto [7] and the generalization of Bayesian induction studied by Stolcke and Omohundro [16], Chen [4].

3 Variational Bayesian Learning

The variational Bayes (VB) [1, 6] has succeeded in many applications [12, 17]. It is empirically shown that VB is less likely to cause overfitting than the EM algorithm, and the free energy calculated by VB can be exploited as a criterion of model selection[1].

[1] The minimum description length (MDL) and Bayesian information criterion (BIC) are also often used as criteria for model selection. However, they are not proper to non-identifiable probabilistic models, s.t. hidden Markov models, maximum entropy models, PCFGs, etc., due to their singular Fisher information matrices. Although VB is still an approximation of true Bayesian learning, its free energy is free from this problem.

Let $X = (x_1, ..., x_N)$, $Y = (y_1, ..., y_N)$ and $\boldsymbol{\theta}$ be observed data, hidden variables and the set of parameters, respectively. VB estimates the variational posterior distribution $q(\boldsymbol{\theta}, Y)$ which approximates the true posterior distribution $p(\boldsymbol{\theta}, Y|X)$ whereas the EM algorithm conducts the point estimation of parameters. The objective function of $q(\boldsymbol{\theta}, Y)$ is free energy $\mathcal{F}$, which is defined as an upper bound of a negative log likelihood,

$$\begin{aligned} -\log p(X) &= -\log \sum_Y \int d\boldsymbol{\theta}\ q(\boldsymbol{\theta}, Y) \frac{p(X, \boldsymbol{\theta}, Y)}{q(\boldsymbol{\theta}, Y)} \\ &\leq -\sum_Y \int d\boldsymbol{\theta}\ q(\boldsymbol{\theta}, Y) \log \frac{p(X, \boldsymbol{\theta}, Y)}{q(\boldsymbol{\theta}, Y)} \equiv \mathcal{F}(X). \end{aligned} \tag{1}$$

It is easy to see that the difference between the free energy and the negative log marginal likelihood[2] is KL-divergence between the posterior $q(\boldsymbol{\theta}, Y)$ and the true posterior $p(\boldsymbol{\theta}, Y|X)$,

$$\mathcal{F}(X) + \log p(X) = \mathrm{KL}(q(\boldsymbol{\theta}, Y)||p(\boldsymbol{\theta}, Y|X)). \tag{2}$$

Since minimizing the free energy leads to minimizing the distance between $q(\boldsymbol{\theta}, Y)$ and $p(\boldsymbol{\theta}, Y|X)$, the optimal $q(\boldsymbol{\theta}, Y)$ as an approximation of $p(\boldsymbol{\theta}, Y|X)$ is given at the minimum free energy.

Assuming a factorization, $q(\boldsymbol{\theta}, Y) = q(\boldsymbol{\theta})q(Y)$, we find the following equations by taking variation of the free energy with respect to $q(\boldsymbol{\theta}, Y)$ and setting to zero.

$$q(\boldsymbol{\theta}) \propto \exp\left[E[\log p(X, Y, \boldsymbol{\theta})]_{q(Y)}\right], \tag{3}$$

$$q(Y) \propto \exp\left[E[\log p(X, Y, \boldsymbol{\theta})]_{q(\boldsymbol{\theta})}\right], \tag{4}$$

where $E[f(x)]_{q(x)} = \int dx f(x)q(x)$. The optimal $q(\boldsymbol{\theta}, Y)$ is iteratively estimated by updating $q(\boldsymbol{\theta})$ and $q(Y)$ alternately. Note that if we give an constraints on, $q(\boldsymbol{\theta}) = \delta(\boldsymbol{\theta}, \boldsymbol{\theta}^*)$ where $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \exp\left[E[\log p(X, Y, \boldsymbol{\theta})]_{q(Y)}\right]$, the above algorithm will be the EM algorithm. Therefore, the EM algorithm is a special case of VB.

Next, we explain the important property of the free energy, capability of model selection. Let's assume that we have many models which can describe probabilistic events. What we want to do here is to choose the most likely model. In the Bayesian approach, people choose one which maximizes a marginal likelihood, $p(X)$. In VB, we choose one which minimizes the free energy, which is an upper bound of the negative log marginal likelihood, Eqn.1.

[2] When a probabilistic model gives event x with probability $p(x|\boldsymbol{\theta})$, the marginal likelihood is $p(x) = \int d\boldsymbol{\theta} p(x|\boldsymbol{\theta})p(\boldsymbol{\theta})$.

4 Proposed Algorithm

4.1 Variational Bayes for PCFGs

In our previous work, we proposed a VB algorithm for PCFGs, and empirically showed that VB is less likely to cause overfitting than the Inside-Outside algorithm[10]. In this section, we briefly explain the VB for PCFGs, then derive the free energy as a criterion to search for a grammatical structure.

Let $G = (V_N, V_T, R, S, \boldsymbol{\theta})$ be a PCFG where V_N, V_T, R and $\boldsymbol{\theta}$ are the sets of non-terminals, terminals, derivation rules and parameters, respectively, and S denotes the start symbol. Assuming the prior of parameters to be a product of Dirichlet distributions, the learning algorithm estimates the hyperparameters of the posterior. Let $\boldsymbol{u} = \{u_r | r \in R\}$ be the hyperparameters of the prior. The hyperparameters of the posterior are the converged values of $\boldsymbol{u}^{(k)}$ $(k = 0, 1, ...)$ defined as,

$$\boldsymbol{u}^{(0)} = \boldsymbol{u} \tag{5}$$

$$u_r^{(k+1)} = u_r + \sum_{n=1}^{N} \sum_{\boldsymbol{r} \in \Phi(x_n)} \frac{\prod_{r \in R} \pi^{(k)}(r)^{c(r;\boldsymbol{r})}}{\sum_{\boldsymbol{r}' \in \Phi(x_n)} \prod_{r \in R} \pi^{(k)}(r)^{c(r;\boldsymbol{r}')}} c(r; \boldsymbol{r}) \tag{6}$$

$$\pi^{(k)}(A \to \alpha) = \exp\left[\psi(u_{A \to \alpha}^{(k)}) - \psi\left(\sum_{\alpha; A \to \alpha \in R} u_{A \to \alpha}^{(k)}\right)\right], \tag{7}$$

where $\Phi(x_n)$ is the set of all the derivations of sentence x_n, $c(r; \boldsymbol{r})$ is the number of occurrences of derivation rule r in derivation $\boldsymbol{r}$ and $\psi(\cdot)$ is the digamma function[3]. The computational complexity of updating k is equal to that of one iteration of the Inside-Outside algorithm, which is $\mathcal{O}(N \max_n(|x_n|)^3)$ where $|x_n|$ is the length of sentence x_n.

As we discussed in section 3, the free energy can be a criterion for model selection. Here, we derive the free energy of PCFGs as a criterion to choose a grammar,

$$\begin{aligned} \mathcal{F}(X, G) = &- \sum_{n=1}^{N} \log \left[\sum_{\boldsymbol{r} \in \Phi(x_n)} \prod_{r \in R} \pi(r)^{c(r;\boldsymbol{r})}\right] + \sum_{A \in V_N} \log \frac{\Gamma\left(\sum_{\alpha; A \to \alpha \in R} u_{A \to \alpha}^*\right)}{\Gamma\left(\sum_{\alpha; A \to \alpha \in R} u_{A \to \alpha}\right)} \\ &- \sum_{r \in R} \log \frac{\Gamma(u_r^*)}{\Gamma(u_r)} + \sum_{A \in V_N} \sum_{\alpha; A \to \alpha \in R} (u_{A \to \alpha}^* - u_{A \to \alpha}) \log \pi(A \to \alpha), \end{aligned} \tag{8}$$

where u_r and u_r^* are the hyperparameters of the prior and the posterior of rule r, respectively. Although the first term of Eqn.8 is the most expensive, it can be

[3] The digamma function is defined as $\psi(x) = \frac{\partial}{\partial x} \log \Gamma(x)$.

$$\left\{\begin{array}{l} S \rightarrow A\ B \\ A \rightarrow A\ B \\ A \rightarrow a \\ B \rightarrow B\ B \\ B \rightarrow a \end{array}\right\} \xrightarrow{\text{merging}} \left\{\begin{array}{l} S \rightarrow A\ A \\ A \rightarrow A\ A \\ A \rightarrow a \end{array}\right\} \qquad \left\{\begin{array}{l} S \rightarrow A\ B \\ A \rightarrow A\ B \\ A \rightarrow a \\ B \rightarrow a \end{array}\right\} \xrightarrow{\text{splitting}} \left\{\begin{array}{l} S \rightarrow A1\ B \\ S \rightarrow A2\ B \\ A1 \rightarrow A1\ B \\ A1 \rightarrow A2\ B \\ A1 \rightarrow a \\ A2 \rightarrow A1\ B \\ A2 \rightarrow A2\ B \\ A2 \rightarrow a \\ B \rightarrow a \end{array}\right\}$$

Fig. 1. The examples of merging and splitting. The left figure is merging, and the right is splitting.

computed efficiently by dynamic programming just as the Inside-Outside algorithm. Therefore, the computational complexity of Eqn.8 is $\mathcal{O}(N \max_n(|x_n|)^3)$.

Note that the free energy is reduced to the approximate posterior which Stolcke and Omohundro [16] and Chen [4] used, provided that $q(\boldsymbol{\theta}) = \delta(\boldsymbol{\theta}, \boldsymbol{\theta}^*)$ where $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|X)$, and the Viterbi approximation is taken. Therefore, their approximate posterior is a special case of the free energy, Eqn.8.

In the following section, we will show three procedures to search for a more likely grammar in the sense of this free energy.

4.2 Grammar Induction Algorithm

Grammar induction can be seen as a search problem whose search space is possible grammars. Several heuristics to search for a grammar have been proposed [4, 7, 3].

Our grammar induction algorithm has three procedures to search for a better grammar, which are merging non-terminals, splitting a non-terminal and deletion of a derivation rule. Merging was studied in [16], and split was proposed in [7]. Every time applying one of these procedures, we calculate the free energy, then we accept the modified grammar if the free energy is decreased.

Merging non-terminals generalizes a grammar, and Splitting a non-terminal specializes a grammar. Figure.1 is an example. The left hand side figure shows merging non-terminal A and B to A, and the right hand side shows splitting non-terminal A into $A1$ and $A2$.

The number of pairs of non-terminals to be merged is $\frac{1}{2}(|V_N|-1)|V_N|$. Since it is not tractable to try all the pairs, we restrict the number of candidates to be merged. The measure to choose candidates is the cosine between the parameter vectors of two non-terminals,

$$cos(A, B) = \frac{\hat{\boldsymbol{\theta}}_A^T \hat{\boldsymbol{\theta}}_B}{||\hat{\boldsymbol{\theta}}_A||\ ||\hat{\boldsymbol{\theta}}_B||}, \tag{9}$$

where $\hat{\boldsymbol{\theta}}_A$ is a parameter vector consisting of all available rules whose left hand side is A. Letting $\boldsymbol{\theta}(A \to \alpha)$ be the parameter of rule $A \to \alpha$, $\hat{\boldsymbol{\theta}}_A$ becomes[4]

$$\hat{\boldsymbol{\theta}}_A = (\hat{\boldsymbol{\theta}}(A \to \alpha), \hat{\boldsymbol{\theta}}(A \to \beta), ...)^T \tag{10}$$

$$\hat{\boldsymbol{\theta}}(A \to \alpha) = \begin{cases} \int d\boldsymbol{\theta}\ q(\boldsymbol{\theta})\boldsymbol{\theta}(A \to \alpha) & \text{if } A \to \alpha \in R \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

The larger $cos(A, B)$ suggests that the roles of non-terminals A and B are closer.

We also restrict the number of candidates to be split based on $\#_{V_N}(\cdot)$,

$$\#_{V_N}(A) = \sum_{\alpha; A \to \alpha \in R} u^*_{A \to \alpha} - u_{A \to \alpha}. \tag{12}$$

Since $\#_{V_N}(A)$ is the expected number of occurrences of non-terminal A in observed sentences, non-terminal A which has large $\#_{V_N}(A)$ may have overloaded syntactic roles.

We make a grammar compact by deleting redundant derivation rules. The candidates to be deleted are chosen based on $\#_R(\cdot)$,

$$\#_R(r) = u^*_r - u_r. \tag{13}$$

The less $\#_R(r)$ is, the fewer derivation rule r is used.

The input of our algorithm is an initial grammar and hyperparameters of the prior, $\boldsymbol{u}$. Although they are arbitrary, in the experiments, we use initial grammar $G = (V_N, V_T, R, S)$ where $V_N = \{S\}$, $V_T = \{\text{terminals in a training corpus}\}$ and $R = \{S \to SS\} \cup \{S \to a | a \in V_T\}$.

Finally, we summarize our grammar induction algorithm in Fig.2. C_{split} and C_{merge} in step 4 and 6 are the maximum number of trials of merging and splitting, respectively. Although the total computational complexity depends on the number of iteration, the complexity in one iteration is equal to $\mathcal{O}(N \max_n(|x_n|)^3)$ as we see in section 4.1.

5 Experiments: Parsing Results

We conducted parsing experiments. First, we compared our algorithm with other grammar induction algorithms. We also conducted an experiment of semi-supervised induction, that achieved 100% parsing coverage.

In every experiment, the Wall Street Journal (WSJ) in Penn Treebank is used for training and test. Training and test corpora consist of part-of-speech (POS) tag sequences. We fix the hyperparameter of the prior, u_r, to 1.0 for all derivation rule $r \in R$. This hyperparameter is known as an uninformative prior.

[4] To evaluate the cosine of any two non-terminals, the dimensionality of $\hat{\boldsymbol{\theta}}_A$ must be the same for any non-terminal A. Therefore, $\hat{\boldsymbol{\theta}}_A$ consists of any possible rules in Chomsky normal form regardless of their existence in R. Therefore, the dimensionality of $\hat{\boldsymbol{\theta}}_A$ is $|V_T| + |V_N|^2$ for all A.

```
1. Input: an initial grammar and the hyperparameters, $u$, of the prior.
2. Estimate hyperparameters $u^*$ of posteriors.
3. Sort non-terminals in descending order of $\#_{V_N}(\cdot)$.
4. for $i$ in $1...C_{\mathrm{split}}$
   (a) Split the $i$th candidate.
   (b) Estimate hyperparameters $u^*$ of posteriors.
   (c) Delete derivation rules while the free energy decreases.
   (d) If the free energy is smaller than that in step 4a, accept the grammar, and go
       to step 3.
5. Sort pairs of non-terminals in descending order of their cosines.
6. for $i$ in $1...C_{\mathrm{merge}}$
   (a) Merge the $i$th candidate.
   (b) Estimate hyperparameters $u^*$ of posteriors.
   (c) Delete derivation rules while the free energy decreases.
   (d) If the free energy is smaller than that in step 6a, accept the grammar, and go
       to step 3.
7. Output: an induced grammar.
```

Fig. 2. Grammar Induction Algorithm

Since our algorithm belongs to Bayesian learning, the most likely parse, $\boldsymbol{r}^*$, is given by summing out parameters,

$$\boldsymbol{r}^* = \arg\max_{\boldsymbol{r}\in\Phi(x)} = \int d\boldsymbol{\theta}\ p(\boldsymbol{r}, x|\boldsymbol{\theta})q(\boldsymbol{\theta}|u^*). \tag{14}$$

Note that it is impossible to apply Viterbi-style parsing to Eqn.14. We therefore exploit reranking [5]. First, 10 Viterbi parses with $\hat{\boldsymbol{\theta}}$ are collected, then the most likely derivation is chosen by calculating Eqn.14 of each derivation.

5.1 Comparison with Other Grammar Induction

We compared our algorithm with Schabes et al. [15] and Hogenhout and Matsumoto [7]. We followed their experimental setting. The training and test corpora are subsets of WSJ. The training corpus had 1,000 sentences of 0-15 words. Test were done on different sentence length, 0-10, 0-15, 10-19 and 20-30 words. Each test corpus had 250 sentences[5]. We conducted these experiments five times.

[5] Schabes et al. used 1042 sentences of 0-15 words for training, and 84 sentences for test. Hogenhout and Matsumoto used 1000 sentences for training and 100 sentences for test. Hogenhout and Matsumoto used 31 POS tags after merging some rare POS tags for larger parsing coverage while Penn Treebank has 46 POS tags. In experiment "Hogenhout & Matsumoto (15)" and "Hogenhout & Matsumoto (18)" in Table 1, the training corpus contained sentences of 0-15 and 0-20 words, respectively.

	bracketing accuracy			
#words	0-10	0-15	10-19	20-30
proposed algorithm	98.1	95.9	93.7	89.4
Hogenhout & Matsumoto (15)	92.0	91.7	83.8	72.0
Hogenhout & Matsumoto (18)	94.1	91.5	86.9	81.8
Schabes et al.	94.4	90.2	82.5	71.5
Right Linear	76	70	63	50
Treebank grammar	46	31	25	N/A

Table 1. Comparison of bracketing accuracies with other methods on various sentence lengths.

Table 1 compares the results. “Hogenhout & Matsumoto (15)” and “Hogenhout & Matsumoto (18)” are induced grammars which have 15 and 18 nonterminals, respectively. The results are taken from [7]. “Schabes et al.”, “Right Linear” and “Treebank grammar” mean Schabes's grammar induction, a systematic right linear branching except for the last punctuation and a grammar extracted from tree labels in WSJ, respectively. These results are from Schabes et al. [15]. Bracketing accuracy in Table 1 is the ration of predicted brackets which is consistent with correct brackets [13].

As Table 1 shows, the proposed algorithm achieved the best score in every test corpus. The parsing coverage of our method was over 99% (see Table 2). Induced grammars had average 22.4 non-terminals and 509.8 derivation rules. The algorithm converged in less than one hour on Pentium 4 3.8 GHz (SuSE 10.0).

One possible reason why our parsing results were better than others might be because the number of terminals in our induced grammars was larger than that of others. The larger number of non-terminals could capture language more appropriately. Hogenhout and Matsumoto [7] fixed the number of non-terminals to 15 or 18. Although our algorithm stops when the free energy converged, their algorithm does not have any criteria to stop. Moreover, it is straightforward to combine our algorithm with search algorithms, e.g. beam search, but it is unclear for their algorithm due to lack of the criteria of model selection. Although we did not compare our free energy with an approximate Bayesian posterior [16, 4], it is well known that the free energy of VB is a tighter bound to the true posterior than their approximate posterior.

5.2 Semi-Supervised Induction as Robust Grammar Induction

In machine learning, semi-supervised learning has received considerable attention. Semi-supervised clustering [19] in grammar induction is to combine a bracketed training corpus and an unbracketed test corpus. Since the proposed algorithm accepts unbracketed sentences also, such a combined corpus leads to 100% parsing coverage on a test corpus even when a training corpus does not have some terminals which occur in a test corpus. We conducted an experiment to show

	#words	0-10	0-15	10-19	20-30
proposed algorithm	0-CB	85.6	66.6	45.6	12.1
	BA	98.1	95.9	93.7	89.4
	coverage	99.8	99.4	100.0	99.6
	failure	3/1250	8/1250	0/1250	5/1250
proposed algorithm + semi-supervised	0-CB	86.0	65.9	44.0	11.4
		(100)	(87.5)	N/A	(0.0)
	BA	98.2	95.8	93.6	89.6
		(100)	(98.1)	N/A	(79.5)
	coverage	100.0	100.0	100.0	100.0
	failure	0/1250	0/1250	0/1250	0/1250

Table 2. Grammar Induction and Semi-Supervised Induction.

how semi-supervised induction works. We used the same training and test corpora as Section 5.1. Our semi-supervised induction has two steps. First, we run the proposed algorithm only on a training corpus, then we add the following derivation rules to make the induced grammar redundant,

$$\{S \rightarrow SS\} \cup \{S \rightarrow a | a \in \{\text{terminals in a test corpus}\}\}$$

where S is a start symbol. After that, we run the algorithm on training and test corpora. Table 2 shows the results. "failure" is the number of sentences which were not parsed by the induced grammar. In semi-supervised part, digits in parentheses show the results of the failed sentences in bracketed induction[6]. The digits in parentheses are comparable with the average of all. This demonstrates grammars learned by semi-supervised induction work well also on sentences failed before. Although semi-supervised induction does not improve zero crossing brackets (0-CB)[7] and bracketing accuracy (BA), this would be because failed sentences are few, i.e. 3, 8, 0 and 5 sentences in 0-10, 0-15, 10-19 and 20-30 words test corpora, respectively.

We have proposed an approach to combine training and test corpora. However, this might be untractable when test corpora are very large. On-line learning is another approach. It is straightforward to apply on-line learning based on the variational Bayes [14]. This would turn the proposed algorithm into an efficient incremental semi-supervised induction algorithm.

6 Discussion: Induced Grammars

So far, we have shown the parsing results of our grammar induction algorithm. However, we believe strongly that grammar induction is not only for obtaining a parser but also can be a tool for grammatical structure understanding.

[6] The corpus of length 10-19 does not have parentheses because the coverage was originally 100%.

[7] 0-CB is the ratio of sentences whose brackets are completely consistent with correct brackets.

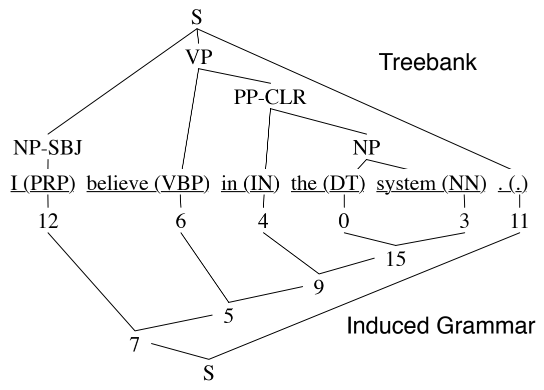

Fig. 3. A parse tree from Treebank and a parse tree predicted by an induced grammar.

Fig.3 is an illustrative example of a parse tree predicted by an induced grammar. Fig.4 lists a subset of the induced grammar used in Fig.3. In this example, non-terminal 15 derives "DT JJ^* NN" where DT is determiner, JJ is adjective and NN is noun. Therefore, non-terminal 15 can be interpreted as a noun phrase (NP). We can also find that start symbol S derives the following grammatical structure (see Appendix A),

$$\mathrm{S} \Rightarrow \underbrace{(\mathrm{PRP}|\mathrm{DT}\ \ \mathrm{JJ}^*\ \ \mathrm{NN})}_{\text{subject}}\ \underbrace{(\mathrm{VBD}|\mathrm{VBZ}|\mathrm{VBP})}_{\text{verb}}\ \underbrace{\mathrm{DT}\ \ \mathrm{JJ}^*\ \ \mathrm{NN}}_{\text{noun-phrase}}\ \ \mathrm{IN}\ \ \underbrace{\mathrm{DT}\ \ \mathrm{JJ}^*\ \ \mathrm{NN}}_{\text{noun-phrase}}\ \ .$$

where $(\cdot|\cdot)$, $*$, ? are the usual notations of regular expressions, PRP is personal pronoun, VBD is verb (past tense), VBZ is verb (third person singular present), VBP is verb (non-third person singular present) and IN is preposition or subordinating conjunction. Clearly, this structure captures typical English sentences.

The above discussion is based on selected examples, and nonsense structures can also be found due to redundant derivation rules. Actually, it is not trivial how to find meaningful grammatical structures. But, the parsing results support that induced grammars have meaningful structures.

7 Conclusion

We have proposed a variational Bayesian PCFG induction algorithm in the context of parameter-estimation-based grammar induction. Experimental results showed that the proposed algorithm induced more well-structured grammars in terms of parsing results than other PCFG induction algorithms. We also showed that induced PCFGs in fact had meaningful recursive structures.

As future work, we may need to exploit a dependency model to improve parsing results. Since our algorithm is greedy, we might combine our algorithm

left hand side non-terminal	$\hat{\boldsymbol{\theta}}$ derivation rule
15	0.28 15 $\rightarrow$ 0 3 $\vdots$
0	0.62 0 $\rightarrow$ DT 0.11 0 $\rightarrow$ 0 10 $\vdots$
10	0.47 10 $\rightarrow$ JJ $\vdots$
3	0.81 3 $\rightarrow$ NN $\vdots$

Fig. 4. The induced grammar in Fig.3. Using shown derivation rules, non-terminal 15 derives "DT JJ* NN", where DT is determiner, JJ is adjective and NN is noun.

with a search algorithm such as beam search. Although we constrained ourselves to PCFGs, we should explore other types of grammars as well. Moreover, how to find meaningful structures from induced grammars remains to be investigated.

Acknowledgments

Special thanks to Dr. Kameya for helpful comments. This research was funded by the 21st Century COE Program "Framework for Systematization and Application of Large-scale Knowledge Resources."

Bibliography

[1] Hagai Attias. A variational Bayesian framework for graphical models. In *Advances in Neural Information Processing Systems*, volume 12, 2000.

[2] J. K. Baker. Trainable grammars for speech recognition. In D. H. Klatt and J. J. Wolf, editors, *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550, 1979.

[3] Joseph Bockhorst and Mark Craven. Refining the structure of a stochastic context-free grammar. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 2001.

[4] Stanley F. Chen. Bayesian grammar induction for language modeling. In *Meeting of the Association for Computational Linguistics*, pages 228–235, 1995.

[5] Michael Collins. Discriminative reranking for natural language parsing. In *Proc. 17th International Conf. on Machine Learning*, pages 175–182, 2000.

[6] Zoubin Ghahramani and Matthew J. Beal. Variational inference for Bayesian mixtures of factor analysers. In *Advances in Neural Information Processing Systems*, volume 12, 2000.

[7] Wide R. Hogenhout and Yuji Matsumoto. A fast method for statistical grammar induction. *Natural Language Engineering*, 4(3):191–209, 1998.

[8] Dan Klein and Christopher D. Manning. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the ACL*, 2002.

[9] Dan Klein and Christopher D. Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the ACL*, 2004.

[10] Kenichi Kurihara and Taisuke Sato. An application of the variational Bayesian approach to probabilistic context-free grammars, 2004. IJCNLP-04 Workshop beyond shallow analyses.

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

[12] David J.C. MacKay. em ensemble learning for hidden markov models. Technical report, 1997.

[13] Fernando C. N. Pereira and Yves Schabes. Inside-outside reestimation from partially bracketed corpora. In *Meeting of the Association for Computational Linguistics*, pages 128–135, 1992.

[14] Masaaki Sato. Online model selection based on the variational bayes. In *Neural Computation*, volume 13, pages 1649–1681, 2001.

[15] Yves Schabes, Michal Roth, and Randy Osborne. Parsing the wall street journal with the inside-outside algorithm. In *ACL*, pages 341–347, 1993.

[16] Andreas Stolcke and Stephen Omohundro. Inducing probabilistic grammars by Bayesian model merging. In *International Conference on Grammatical Inference*, 1994.

[17] Naonori Ueda and Zoubin Ghahramani. Bayesian model search for mixture models based on optimizing variational bounds. *Neural Networks*, 15(10):1223–1241, 2002.

[18] Menno van Zaanen. Abl: Alighment-based learning. In *COLING*, volume 18, pages 961–967, 2000.

$\hat{\boldsymbol{\theta}}$	derivation rule
0.80	S $\rightarrow$ 7 11
0.64	7 $\rightarrow$ 12 5
0.23	12 $\rightarrow$ PRP
0.19	12 $\rightarrow$ 0 3
0.62	0 $\rightarrow$ DT
0.11	0 $\rightarrow$ 0 10
0.47	10 $\rightarrow$ JJ
0.81	3 $\rightarrow$ NN
0.79	5 $\rightarrow$ 6 9
0.30	6 $\rightarrow$ VBD
0.26	6 $\rightarrow$ VBZ
0.15	6 $\rightarrow$ VBP
0.26	9 $\rightarrow$ 8 1
0.24	8 $\rightarrow$ 0 3
0.48	1 $\rightarrow$ 4 15
0.55	4 $\rightarrow$ IN
0.28	15 $\rightarrow$ 0 3
0.85	11 $\rightarrow$.

Fig. 5. A subset of the induced grammar in Fig.3.

[19] Kiri Wagsta, Claire Cardie, Seth Rogers, and Stefan Schroedl. Constrained k-means clustering with background knowledge. In *Proceedings of 18th International Conference on Machine Learning*, pages 577–584, 2001.

Appendix

A A Grammatical Structure in an Induced Grammar

Fig.5 lists a subset of the induced grammar used in Fig.3. These derivation rules leads a grammatical structure,

$$\mathrm{S} \Rightarrow \underbrace{(\mathrm{PRP}|\mathrm{DT}\ \ \mathrm{JJ}^*\ \ \mathrm{NN})}_{\text{subject}}\ \ \underbrace{(\mathrm{VBD}|\mathrm{VBZ}|\mathrm{VBP})}_{\text{verb}}\ \ \underbrace{\mathrm{DT}\ \ \mathrm{JJ}^*\ \ \mathrm{NN}}_{\text{noun-phrase}}\ \ \mathrm{IN}\ \ \underbrace{\mathrm{DT}\ \ \mathrm{JJ}^*\ \ \mathrm{NN}}_{\text{noun-phrase}}\ \ .$$

where PRP is personal noun, DT is determiner, JJ is adjective, NN is noun, VBD is verb (past tense), VBZ is verb (third person singular present), VBP is (non-third person singular present) and IN is preposition or subordinating conjunction. "DT JJ* NN" can be interpreted as a noun phrase. Therefore, (PRP|DT JJ* NN) is also a noun phrase. In this example, (PRP|DT JJ* NN) makes the subject of the sentence.