Third-order Variational Reranking on Packed-Shared Dependency Forests

Katsuhiko Hayashi[†], **Taro Watanabe**[‡], **Masayuki Asahara**[†], **Yuji Matsumoto**[†]
[†]Nara Insutitute of Science and Technology
Ikoma, Nara, 630-0192, Japan
[‡]National Institute of Information and Communications Technology
Sorakugun, Kyoto, 619-0289, Japan
{katsuhiko-h,masayu-a,matsu}@is.naist.jp
taro.watanabe@nict.go.jp

Abstract

We propose a novel *forest reranking* algorithm for discriminative dependency parsing based on a variant of Eisner's generative model. In our framework, we define two kinds of generative model for reranking. One is learned from training data offline and the other from a forest generated by a baseline parser on the fly. The final prediction in the reranking stage is performed using linear interpolation of these models and discriminative model. In order to efficiently train the model from and decode on a hypergraph data structure representing a forest, we apply extended *inside/outside* and *Viterbi* algorithms. Experimental results show that our proposed forest reranking algorithm achieves significant improvement when compared with conventional approaches.

1 Introduction

Recently, much of research on statistical parsing has been focused on k-best (or forest) reranking (Collins, 2000; Charniak and Johnson, 2005; Huang, 2008). Typically, reranking methods first generate a list of top-k candidates (or a forest) from a baseline system, then rerank the candidates with arbitrary features that are intractable within the baseline system. In the reranking framework, the baseline system is usually modeled with a generative model, and a discriminative model is used for reranking. Sangati et al. (2009) reversed the usual order of the two models for dependency parsing by employing a generative model to rescore the k-best candidates provided by a discriminative model. They use a variant of Eisner's generative model C (Eisner, 1996b; Eisner, 1996a) for reranking and extend it to capture higher-order information than Eisner's second-order generative model. Their reranking model showed large improvements in dependency parsing accuracy. They reported that the discriminative model is very effective at filtering out bad candidates, while the generative model is able to further refine the selection among the few best candidates.

In this paper, we propose a *forest generative reranking* algorithm, opposed to Sangati et al. (2009)'s approach which reranks only k-best candidates. Forests usually encode better candidates more compactly than k-best lists (Huang, 2008). Moreover, our reranking uses not only a generative model obtained from training data, but also a sentence specific generative model learned from a forest. In the reranking stage, we use linearly combined model of these models. We call this *variational reranking* model. The model proposed in this paper is factored in the third-order structure, therefore, its non-locality makes it difficult to perform the reranking with an usual 1-best *Viterbi* search. To solve this problem, we also propose a new search algorithm, which is inspired by the third-order dynamic programming parsing algorithm (Koo and Collins, 2010). This algorithm enables us an exact 1-best reranking without any approximation. We summarize our contributions in this paper as follows.

- To extend k-best to forest generative reranking.
- We introduce *variational reranking* which is a combination approach of generative reranking and variational decoding (Li et al., 2009).
- To obtain 1-best tree in the reranking stage, we

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1479–1488,
Edinburgh, Scotland, UK, July 27–31, 2011.

propose an exact 1-best search algorithm with the third-order model.

In experiments on English Penn Treebank data, we show that our proposed methods bring significant improvement to dependency parsing. Moreover, our variational reranking framework achieves consistent improvement, compared to conventional approaches, such as simple k-best and forest-based generative reranking algorithms.

2 Dependency Parsing

Given an input sentence $x \in \mathcal{X}$, the task of statistical dependency parsing is to predict output dependencies $\hat{y}$ for x. The task is usually modeled within a discriminative framework, defined by the following equation:

$$\begin{aligned} \hat{y} &= \operatorname*{argmax}_{y \in \mathcal{Y}} s(x, y) \\ &= \operatorname*{argmax}_{y \in \mathcal{Y}} \boldsymbol{\lambda}^{\top} \cdot \mathbf{F}(y, x) \end{aligned} \quad (1)$$

where $\mathcal{Y}$ is the output space, $\boldsymbol{\lambda}$ is a parameter vector, and $\mathbf{F}()$ is a set of feature functions.

We denote a set of candidates as $G(x)$. By using $G(x)$, the conditional probability $p(y|x)$ is typically derived as follows:

$$p(y|x) = \frac{e^{\gamma \cdot s(x,y)}}{Z(x)} = \frac{e^{\gamma \cdot s(x,y)}}{\sum_{y \in G(x)} e^{\gamma \cdot s(x,y)}} \quad (2)$$

where $s(x, y)$ is the score function shown in Eq.1 and γ is a scaling factor to adjust the sharpness of the distribution and $Z(x)$ is a normarization factor.

2.1 Hypergraph Representation

We propose to encode many hypotheses in a compact representation called dependency forest. While there may be exponentially many dependency trees, the forest represents them in polynomial space. A dependency forest (or tree) can be defined as a hypergraph data strucure $\mathbf{HG}$ (Tu et al., 2010).

Figure 1 shows an example of a hypergraph for a dependency tree. A shaded hyperedge e is defined as the following form:

$$e : \langle (\mathrm{I}_{1,2}, \mathrm{girl}_{3,5}, \mathrm{with}_{5,8}), \mathrm{saw}_{1,8} \rangle.$$

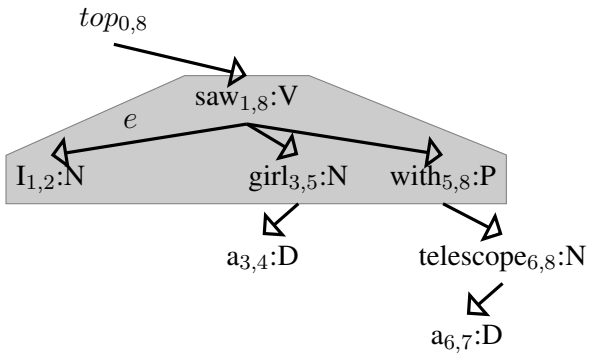

Figure 1: An example of dependency tree for a sentence "I saw a girl with a telescope".

The node saw$_{1,8}$ is a head node of e. The nodes, I$_{1,2}$, girl$_{3,5}$ and with$_{5,8}$, are tail nodes of e. The hyperedge e is an incoming edge for saw$_{1,8}$ and outgoing edge for each of I$_{1,2}$, girl$_{3,5}$ and with$_{5,8}$[1].

More formally, $\mathbf{HG}(x)$ of a forest is a pair $\langle V, E \rangle$, where V is a set of nodes and E is a set of hyperedges. Given a length m sentence $x = (w_1 \dots w_m)$, each node $v \in V$ is in the form of $w_{i,j}$ $(= (w_i \dots w_{j+1}))$ which denotes that a word w dominates the substring from positions i to j. In our implementation, each word is paired with POS-tag $tag(w)$. We denote the root node of dependency tree y as top. Each hyperedge $e \in E$ is a pair $\langle tails(e), head(e) \rangle$, where $head(e) \in V$ is the head and $tails(e) \in V^+$ are its dependants. For notational brevity of algorithmic description, we do not distinguish left and right tails in the definition, but, our implementation implicitly distinguishes left tails $tails_L(e)$ and right tails $tails_R(e)$. We define the set of incoming edges of a node v as $IE(v)$ and the set of outgoing edges of a node v as $OE(v)$.

3 Forest Reranking

3.1 Generative Model for Reranking

Given a node v in a dependency tree y, the left and right children are generated as two separate Markov sequences, each conditioned on ancestral and sibling information (context). Like a variation of Eisner's generative model C (Eisner, 1996b; Eisner, 1996a),

[1] In Figure 1, according to custom of dependency tree description, the direction of hyperedge is written as from head to tail nodes. However, in this paper, "incoming" and "outgoing" have the same meanings as those in (Huang, 2006).

Table 1: An event list of tri-sibling model whose event space is $v|h, sib, tsib, dir$, extracted from hyperedge e in Figure 1. EOC is an end symbol of sequence.

event space
I | saw NONE NONE L
EOC | saw I NONE L
girl | saw NONE NONE R
with | saw girl NONE R
EOC | saw with girl R

the probability of our model q is defined as follows:

$$q(v) = \prod_{l=1}^{|tails_L(e)|} q(v_l|\mathrm{C}(v_l)) \cdot q(v_l) \times \prod_{r=1}^{|tails_R(e)|} q(v_r|\mathrm{C}(v_r)) \cdot q(v_r) \quad (3)$$

where $|tails_L(e)|$ and $|tails_R(e)|$ are the number of left and right children of v, v_l and v_r are the left and right child of position l and r in each side. $\mathrm{C}(v)$ is a context event space of v. We explain the context event space later in more detail. The probability of the entire dependency tree y is recursively computed by $q(y(top))$ where $y(top)$ denotes a top node of y.

The probability $q(v|\mathrm{C}(v))$ is dependent on a context space $\mathrm{C}(v)$ for a node v. We define two kinds of context spaces. First, we define a tri-sibling model whose context space consists of the head node, sibling node, tri-sibling node and direction of a node v:

$$q_1(v|\mathrm{C}(v)) = q_1(v|h, sib, tsib, dir) \quad (4)$$

where h, sib and $tsib$ are head, sibling and tri-sibling node of v, and dir is a direction of v from h. Table 1 shows an example of an event list of the tri-sibling model, which is extracted from hyperedge e in Figure 1. EOC indicates the end of the left or right child sequence. This is factored in a tri-sibling structure shown in the left side of Figure 2.

Eq.4 is further decomposed into a product of the form consisting of three terms:

$$\begin{aligned} & q_1(v|h, sib, tsib, dir) \quad (5) \\ & = q_1(dist(v,h), wrd(v), tag(v)|h, sib, tsib, dir) \\ & = q_1(tag(v)|h, sib, tsib, dir) \\ & \times q_1(wrd(v)|tag(v), h, sib, tsib, dir) \\ & \times q_1(dist(v,h)|wrd(v), tag(v), h, sib, tsib, dir) \end{aligned}$$

where $tag(v)$ and $wrd(v)$ are the POS-tag and word of v and $dist(v,h)$ is the distance between positions of v and h. The values of $dist(v,h)$ are partitioned into 4 categories: $1, 2, 3-6, 7-\infty$.

Second, following Sangati et al. (2009), we define a grandsibling model whose context space consists of the head node, sibling node, grandparent node and direction of a node v.

$$q_2(v|\mathrm{C}(v)) = q_2(v|h, sib, g, dir) \quad (6)$$

where g is a grandparent node of v. Analogous to Eq.5, Eq.6 is decomposed into three terms:

$$\begin{aligned} & q_2(v|h, sib, g, dir) \quad (7) \\ & = q_2(dist(v,h), wrd(v), tag(v)|h, sib, g, dir) \\ & = q_2(tag(v)|h, sib, g, dir) \\ & \times q_2(wrd(v)|tag(v), h, sib, g, dir) \\ & \times q_2(dist(v,h)|wrd(v), tag(v), h, sib, g, dir) \end{aligned}$$

where notations are the same as those in Eq.5 with the exception of tri-sibling $tsib$ and grandparent g. This model is factored in a grandsibling structure shown in the right side of Figure 2.

The direct estimation of tri-sibling and grandsibling models from a corpus suffers from serious data sparseness issues. To overcome this, Eisner (1996a) proposed a back-off strategy which reduces the conditioning of a model. We show the reductions list for each term of two models in Table 2. The usage of reductions list is identical to Eisner (1996a) and readers may refer to it for further details.

The final prediction is performed using a log-linear interpolated model. It interpolates the baseline discriminative model and two (tri-sibling and grandsibling) generative models.

$$\hat{y} = \operatorname*{argmax}_{y \in G(x)} \sum_{n=1}^{2} \log q_n(top(y))^{\theta_n} + \log p(y|x)^{\theta_{base}} \quad (8)$$

where $\boldsymbol{\theta}$ are parameters to adjust the weight of each term in prediction. These parameters are tuned using *MERT* algorithm (Och, 2003) on development data using a criterion of accuracy maximization. The reason why we chose *MERT* is that it effectively tunes dense parameters with a line search algorithm.

Table 2: Reduction lists for tri-sibling and grandsibling models: $wt()$, $w()$ and $t()$ mean word and POS-tag, word, POS-tag for a node. d indicates the direction. The first reduction on the list keeps all or most of the original condition; later reductions throw away more and more of this information.

tri-sibling			grandsibling		
1-st term	2-nd term	3-rd term	1-st term	2-nd term	3-rd term
$wt(h)$,$wt(sib)$,$wt(tsib)$,d	$wt(h)$,$t(sib)$,d	$wt(v)$,$t(h)$,$t(sib)$,d	$wt(h)$,$wt(sib)$,$wt(g)$,d	$wt(h)$,$t(sib)$,d	$wt(v)$,$t(h)$,$t(sib)$,d
$wt(h)$,$wt(sib)$,$t(tsib)$,d	$t(h)$,$t(sib)$,d	$t(v)$,$t(h)$,$t(sib)$,d	$wt(h)$,$wt(sib)$,$t(g)$,d	$t(h)$,$t(sib)$,d	$t(v)$,$t(h)$,$t(sib)$,d
$t(h)$,$wt(sib)$,$t(tsib)$,d $wt(h)$,$t(sib)$,$t(tsib)$,d	—	—	$t(h)$,$wt(sib)$,$t(g)$,d $wt(h)$,$t(sib)$,$t(g)$,d	—	—
$t(h)$,$t(sib)$,$t(tsib)$,d	—	—	$t(h)$,$t(sib)$,$t(g)$,d	—	—

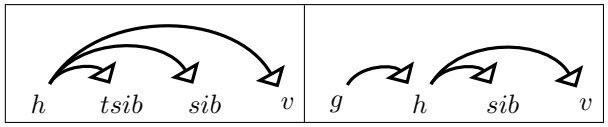

Figure 2: The left side denotes tri-sibling structure and the right side denotes grandsibling structure.

Table 3: A summarization of the model factorization and order

first-order	McDonald et al. (2005)
second-order (sibling)	Eisner (1996a) McDonald et al. (2005)
third-order (tri-sibling)	tri-sibling model Model 2 (Koo and Collins, 2010)
third-order (grandsibling)	grandsibling model (Sangati et al., 2009) Model 1 (Koo and Collins, 2010)

3.2 Exact Search Algorithm

Our baseline discriminative model uses first- and second-order features provided in (McDonald et al., 2005; McDonald and Pereira, 2006). Therefore, both our tri-sibling model and baseline discriminative model integrate local features that are factored in one hyperedge. On the other hand, the grandsibling model has non-local features because the grandparent is not factored in one hyperedge. We summarize the order of each model in Table 3. Our reranking models are generative versions of Koo and Collins (2010)'s third-order factorization model.

Non-locality of weight function makes it difficult to perform the search of Eq.8 with an usual exact *Viterbi* 1-best algorithm. One solution to resolve the intractability is an approximate k-best *Viterbi* search. For a constituent parser, Huang (2008) applied *cube pruning* techniques to forest reranking with non-local features. Cube pruning is originally proposed for the decoding of *statistical machine translation* (SMT) with an integrated n-gram language model (Chiang, 2007). It is an approximate k-best *Viterbi* search algorithm using beam search and lazy computation (Huang and Chiang, 2005).

In the case of a dependency parser, Koo and Collins (2010) proposed dynamic-programming-based third-order parsing algorithm, which enumerates all grandparents with an additional loop. Our hypergraph based search algorithm for Eq.8 share the same spirit to their third-order parsing algorithm since the grandsibling model is similar to their model 1 in that it is factored in grandsibling structure. Algorithm 1 shows the search algorithm. This is almost the same bottom-up 1-best *Viterbi* algorithm except an additional loop in line 4. Line 4 references outgoing edge e' of node h from a set of outgoing edges $OE(h)$. $tails(e)$ contains a node v, the sibling node sib and tri-sibling node $tsib$ of v, moreover, the head of e' ($head(e')$) is the grandparent for v and sib. Thus, in line 5, we can capture tri-sibling and grandsibling information and compute the current inside estimate of Eq.8.

In our actual implementation, each score of components in Eq.8 is represented as a cost. This is written as a shortest path search algorithm with a tropical (real) semiring framework (Mohri, 2002; Huang, 2006). Therefore, $\oplus$ denotes the $\min$ operater and $\otimes$ denotes the $+$ operater. The function f is defined as follows:

$$f(d(v_1, e), \ldots, d(v_{|e|}, e))) = \bigotimes_{i=1}^{|e|} d(v_i, e) \qquad (9)$$

where $d(v_i, e)$ denotes the current estimate of the best cost for a pair of node v_i and a hyperedge e. $\bigotimes$ sums the best cost of a pair of a sub span node and hyperedge e. Each c_{tsib} and c_{gsib} in line 5 and 7 indicates the cost of tri-sibling and grandsibling

Algorithm 1 Exact DP-Search Algorithm($\mathbf{HG}(x)$)

1: **for** $h \in V$ in bottom-up topological order **do**
2: **for** $e \in IE(h)$ **do**
3: // $tails(e)$ is $\{v_1, \ldots, v_{|e|}\}$.
4: **for** $e' \in OE(h)$ **do**
5: $d(h, e') = \oplus f(d(v_1, e), \ldots, d(v_{|e|}, e)) \otimes w_e \otimes c_{tsib}(h, tails(e)) \otimes c_{gsib}(head(e'), h, tails(e))$
6: **if** $h == top$ **then**
7: $d(h) = \oplus f(d(v_1, e), \ldots, d(v_{|e|}, e)) \otimes w_e \otimes c_{tsib}(h, tails(e))$

model. w_e indicates the cost of hyperedge e computed from a baseline discriminative model. Lines 6-7 denote the calculation of the best cost for a top node. We do not compute the cost of the grandsibling model when h is top node because top node has no outgoing edges.

Our baseline k-best second-order parser is implemented using Huang and Chiang (2005)'s algorithm 2 whose time complexity is $O(m^3 + mk \log k)$. Koo and Collins (2010)'s third-order parser has $O(m^4)$ time complexity and is theoretically slower than our baseline k-best parser for a long sentence. Our search algorithm is based on the third-order parsing algorithm, but, the search space is previously shrank by a baseline parser's k-best approximation and a forest pruning algorithm presented in the next section. Therefore, the time efficiency of our reranking is unimpaired.

3.3 Forest Pruning

Charniak and Johnson (2005) and Huang (2008) proposed forest pruning algorithms to reduce the size of a forest. Huang (2008)'s pruning algorithm uses a 1-best *Viterbi inside/outside* algorithm to compute an inside probability $\beta(v)$ and an outside probability $\alpha(v)$, while Charniak and Johnson (2005) use the usual *inside/outside* algorithm.

In our experiments, we use Charniak and Johnson (2005)'s forest pruning criterion because the variational model needs traditional *inside/outside* probabilities for its ML estimation. We prune away all hyperedges that have $score < \rho$ for a threshold ρ.

$$score = \frac{\alpha\beta(e)}{\beta(top)}. \tag{10}$$

Following Huang (2008), we also prune away nodes with all incoming and outgoing hyperedges pruned.

4 Variational Reranking Model

In place of a *maximum a posteriori* (MAP) decision based on Eq.2, the *minimum Bayes risk* (MBR) decision rule (Titov and Henderson, 2006) is commonly used and defined as following equation:

$$\hat{y} = \operatorname*{argmin}_{y \in G(x)} \sum_{y' \in G(x)} loss(y, y') p(y'|x) \tag{11}$$

where $loss(y, y')$ represents a loss function[2]. As an alternative to the MBR decision rule, Li et al. (2009) proposed a variational decision rule that rescores candidates with an approximate distribution $q^* \in \mathcal{Q}$.

$$\hat{y} = \operatorname*{argmax}_{y \in G(x)} q^*(y) \tag{12}$$

where q^* minimizes the KL divergence $\mathrm{KL}(p||q)$

$$\begin{aligned} q^* &= \operatorname*{argmin}_{q \in \mathcal{Q}} \mathrm{KL}(p||q) \\ &= \operatorname*{argmax}_{q \in \mathcal{Q}} \sum_{y \in G(x)} p \log q \end{aligned} \tag{13}$$

where each p and q represents $p(y|x)$ and $q(y)$. For SMT systems, q^* is modeled by n-gram language model over output strings. While the decoding based on q^* is an approximation of intractable MAP decoding[3], it works as a rescoring function for candidates generated from a baseline model. Here, we propose to apply the variational decision rule to dependency parsing. For dependency parsing, we can choose to model q^* as the tri-sibling and grandsibling generative models in section 3.

[2] In case of dependency parsing, Titov and Henderson (2006) proposed that a loss function is simply defined using a dependency attachment score.

[3] In SMT, a marginalization of all derivations which yield a paticular translation needs to be carried out for each translation. This makes the MAP decoding NP-hard in SMT. This variational approximate framework can be applied to other tasks collapsing *spurious ambiguity*, such as latent-variable parsing (Matsuzaki et al., 2005).

Algorithm 2 DP-ML Estimation($\mathbf{HG}(x)$)

```
1: run inside and outside algorithm on HG(x)
2: for v ∈ V do
3:   for e ∈ IE(v) do
4:     c_tsib = p_e · α(v)/β(top)
5:     for u ∈ tails(e) do
6:       c_tsib = c_tsib · β(u)
7:       for e' ∈ IE(u) do
8:         c_gsib = p_e · p_e' · α(v)/β(top)
9:         for u' ∈ tails(e) \ u do
10:          c_gsib = c_gsib · β(u')
11:        for u'' ∈ tails(e') do
12:          c_gsib = c_gsib · β(u'')
13:        for u'' ∈ tails(e') do
14:          c̄2(u''|C(u''))+ = c_gsib
15:          c̄2(C(u''))+ = c_gsib
16:    for u ∈ tails(e) do
17:      c̄1(u|C(u))+ = c_tsib
18:      c̄1(C(u))+ = c_tsib
19: MLE estimate q1*, q2* using formula Eq.14
```

4.1 ML Estimation from a Forest

$q^*(v|\mathrm{C}(v))$ is estimated from a forest using a *maximum likelihood estimation* (MLE). The count of events is no longer an integer count, but an expected count under p, which is formulated as follows:

$$q^*(v|\mathrm{C}(v)) = \frac{\overline{c}(v|\mathrm{C}(v))}{\overline{c}(\mathrm{C}(v))} = \frac{\sum_y p(y|x) c_{v|\mathrm{C}(v)}(y)}{\sum_y p(y|x) c_{\mathrm{C}(v)}(y)} \quad (14)$$

where $c_{\mathrm{e}}(y)$ is the number of event e in y. The estimation of Eq.14 can be efficiently performed on a hypergraph data structure $\mathbf{HG}(x)$ of a forest.

Algorithm 2 shows the estimation algorithm. First, it runs the *inside/outside* algorithm on $\mathbf{HG}(x)$. We denote inside weight for a node v as $\beta(v)$ and outside weight as $\alpha(v)$. For each hyperedge e, we denote c_{tsib} as the posterior weight for computing expected count $\overline{c}_1$ of events in the tri-sibling model q_1^*. Lines 16-18 compute $\overline{c}_1$ for all events occuring in a hyperedge e.

The expected count $\overline{c}_2$ needed for the estimation of grandsibling model q_2^* is extracted in lines 7-15. $\overline{c}_2$ for a grandsibling model must be extracted over two hyperedges e and e' because it needs grandparent information. Lines 8-12 show the algorithm to compute the posterior weight c_{gsib} of e and e', which is similar to that to compute the posterior weight of rules of *tree substitution grammars* used in tree-based MT systems (Mi and Huang, 2008). Lines 13-15 compute expected counts $\overline{c}_2$ of events occuring over two hyperedges e and e'. Finally, line 19 estimates q_1^* and q_2^* using the form in Eq.14.

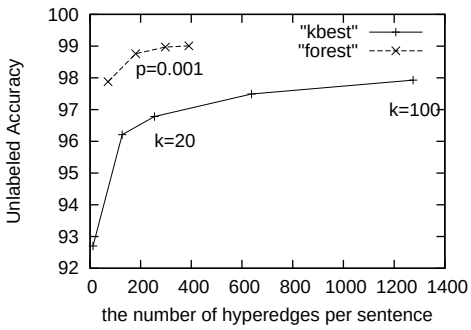

Figure 3: The relationship between tha data size (the number of hyperedges) and oracle scores on development data: Forests encode candidates with high accuracy scores more compactly than k-best lists.

Li et al. (2009) assumes n-gram locality of the forest to efficiently train the model, namely, the baseline n-gram model has larger n than that of variational n-gram model. In our case, grandsibling locality is not embedded in the forest generated from the baseline parser. Therefore, we need to reference incoming hyperedges of tail nodes in line 7.

y^* of Eq.12 may be locally appropriate but globally inadequate because q^* only approximates p. Therefore, we log-linearly combine q^* with a global generative model estimated from the training data and the baseline discriminative model.

$$\hat{y} = \operatorname*{argmax}_{y \in G(x)} \sum_{n=1}^{2} \log q_n(top(y))^{\theta_n} + \sum_{n=1}^{2} \log q_n^*(top(y))^{\theta_n^*} + \log p(y|x)^{\theta_{base}} \quad (15)$$

Algorithm1 is also applicable to the decoding of Eq.15. Note that this framework is a combination of variational decoding and generative reranking. We call this framework *variational reranking*.

Table 4: The statistics of forests and 20-best lists on development data: this shows the average number of hyperedges and nodes per sentence and oracle scores.

	forest	20-best
pruning threshold	$\rho = 10^{-3}$	—
ave. num of hyperedges	180.67	255.04
ave. num of nodes	135.74	491.42
oracle scores	98.76	96.78

5 Experiments

Experiments are performed on English Penn Treebank data. We split WSJ part of the Treebank into sections 02-21 for training, sections 22 for development, sections 23 for testing. We use Yamada and Matsumoto (2003)'s head rules to convert phrase structure to dependency structure. We obtain k-best lists and forests generated from the baseline discriminative model which has the same feature set as provided in (McDonald et al., 2005), using the second-order Eisner algorithms. We use MIRA for training as it is one of the learning algorithms that achieves the best performance in dependency parsing. We set the scaling factor $\gamma = 1.0$.

We also train a generative reranking model from the training data. To reduce the data sparseness problem, we use the back-off strategy proposed in (Eisner, 1996a). Parameters $\boldsymbol{\theta}$ are trained using MERT (Och, 2003) and for each sentence in the development data, 300-best dependency trees are extracted from its forest. Our variational reranking does not need much time to train the model because the training is performed over not the training data (39832 sentences) but the development data (1700 sentences)[4]. After MERT was performed until the convergence, the variational reranking finally achieved a 94.5 accuracy score on development data.

5.1 k-best Lists vs. Forests

Figure 3 shows the relationship between the size of data structure (the number of hyperedges) and accuracy scores on development data. Obviously, forests can encode a large number of potential candidates more compactly than k-best lists. This means that for reranking, there is more possibility of selecting good candidates in forests than k-best lists.

Table 4 shows the statistics of forests and 20-best lists on development data. This setting, threshold $\rho = 10^{-3}$ for pruning, is also used for testing. Forests, which have an average of 180.67 hyperedges per sentence, achieve oracle score of 98.76, which is about 1.0% higher than the 96.78 oracle score of 20-best lists with 255.04 hyperedges per sentence. Though the size of forests is smaller than that of k-best lists, the oracle scores of forests are much higher than those of k-best lists.

5.2 The Performance of Reranking

First, we compare the performance of variational decoding with that of MBR decoding. The results are shown in Table 5. Variational decoding outperforms MBR decodings. However, compared with baseline, the gains of variational and MBR decoding are small. Second, we also compare the performance of variational reranking with k-best and forest generative reranking algorithms. Table 6 shows that our variational reranking framework achieves the highest accuracy scores.

Being different from the decoding framework, reranking achieves significant improvements. This result is intuitively reasonable because the reranking model obtained from training data has the ability to select a globally consistent candidate, while the variational approximate model obtained from a forest only supports selecting a locally consistent candidate. On the other hand, the fact that variational reranking achieves the best results clearly indicates that the combination of sentence specific generative model and that obtained from training data is successful in selecting both locally and globally appropriate candidate from a forest.

Table 7 shows the parsing time (on 2.66GHz Quad-Core Xeon) of the baseline k-best, generative reranking and variational reranking parsers (java implemented). The variational reranking parser contains the following procedures.

1. k-best forest creation (baseline)
2. Estimation of variational model
3. Forest pruning
4. Search with the third-order model

Our reranking parser incurred little overhead to the

[4]To generate forests, sentences are parsed only once before the training. MERT is performed over the forests. We can also apply a more efficient hypergraph MERT algorithm (Kumar et al., 2009) to the training than a simple MERT algorithm.

Table 5: The comparison of the decoding frameworks: MBR decoding seeks a candidate which has the highest accuracy scores over a forest (Kumar et al., 2009). Variational decoding is performed based on Eq.8.

Decoding \ Eval	Unlabeled
baseline	91.9
MBR (8-best forest)	91.99
Variational (8-best forest)	**92.17**

Table 6: The comparison of the reranking frameworks: Generative means k-best or forest reranking algorithm based on a generative model estimated from a corpus. Variational reranking is performed based on Eq.15.

Reranking \ Eval	Unlabeled
Generative (8-best)	92.66
Generative (8-best forest)	92.72
Variational (8-best forest)	**92.87**

Table 7: The parsing time (CPU second per sentence) and accuracy score of the baseline k-best, generative reranking and variational reranking parsers

k	baseline	generative	variational
2	0.09 (91.9)	+0.03 (92.67)	+0.05 (92.76)
4	0.1 (91.9)	+0.05 (92.68)	+0.09 (92.81)
8	0.13 (91.9)	+0.06 (92.72)	+0.11 (92.87)
16	0.18 (91.9)	+0.07 (92.75)	+0.12 (92.89)
32	0.29 (91.9)	+0.07 (92.73)	+0.13 (92.89)
64	0.54 (91.9)	+0.08 (92.72)	+0.15 (92.87)

Table 8: The comparison of tri-sibling and grandsibling models: the performance of the grandsibling model outperforms that of the tri-sibling model.

Model \ Eval	Unlabeled
tri-sibling	92.63
grandsibling	92.74

Table 10: Comparison of our best result (using 16-best forests) with other best-performing Systems on the whole section 23

Parser	English
McDonald et al. (2005)	90.9
McDonald and Pereira (2006)	91.5
Koo et al. (2008) standard	92.02
Huang and Sagae (2010)	92.1
Koo and Collins (2010) model1	93.04
Koo and Collins (2010) model2	92.93
this work	92.89
Koo et al. (2008) semi-sup	93.16
Suzuki et al. (2009)	93.79

baseline parser in terms of runtime. This means that our reranking parser can parse sentences at reasonable times.

5.3 The Effects of Third-order Factors and Error Analysis

From results in section 5.2, our variational reranking model achieves higher accuracy scores than the others. To analyze the factors that improve accuracy scores, we further investigate whether variational reranking is performed better with the tri-sibling or grandsibling model. Table 8 indicates that grandsibling model achieves a larger gain than that of tri-sibling model. Table 9 shows the examples whose accuracy scores improved by the grandsibling model. For example, the dependency relationship from *Verb* to *Noun phrase* was corrected by our proposed model.

On the other hand, many errors remain still in our results. In our experiments, 48% of sentences which contain errors have *Prepositional* word errors. In fact, well-known *PP-Attachment* is a problem to be solved for natural language parsers. Other remaining errors are caused by symbols such as .,:“”(). 45% sentences contain such a dependency mistake. Adding features to solve these problems may potentially improve our parser more.

5.4 Comparison with Other Systems

Table 10 shows the comparison of the performance of variational reranking (16-best forests) with that of other systems. Our method outperforms supervised parsers with second-order features, and achieves comparable results compared to a parser with third-order features (Koo and Collins, 2010). We can not directly compare our method with semi-supervised parsers such as Koo et al. (2008)'s semi-sup and Suzuki et al. (2009), because ours does not use additional unlabeled data for training. The model trained from unlabeled data can be easily incorporated into our reranking framework. We plan to investigate semi-supervised learning in future work.

Table 9: Examples of outputs for input sentence No.148 and No.283 in section 23 from baseline and variational reranking parsers. The underlined portions show the effect of the grandsibling model.

sent (No.148)	A quick turnaround is crucial to Quantum because its cash requirements remain heavy .													
correct	3	3	4	0	4	5	6	4	11	11	12	8	12	4
baseline	3	3	4	0	4	5	6	4	11	11	8	8	12	4
proposed	3	3	4	0	4	5	6	4	11	11	12	8	12	4
sent (No.283)	Many called it simply a contrast in styles .													
correct	2	0	2	6	6	2	6	7	2					
baseline	2	0	2	2	6	2	6	7	2					
proposed	2	0	2	6	6	2	6	7	2					

6 Related Work

Collins (2000) and Charniak and Johnson (2005) proposed a reranking algorithm for constituent parsers. Huang (2008) extended it to a forest reranking algorithm with non-local features. Our framework is for a dependency parser and the decoding in the reranking stage is done with an exact 1-best dynamic programming algorithm. Sangati et al. (2009) proposed a k-best generative reranking algorithm for dependency parsing. In this paper, we use a similar generative model, but combined with a variational model learned on the fly. Moreover, our framework is applicable to forests, not k-best lists.

Koo and Collins (2010) presented third-order dependency parsing algorithm. Their model 1 is defined by an enclosing grandsibling for each sibling or grandchild part used in Carreras (2007). Our grandsibling model is similar to the model 1, but ours is defined by a generative model. The decoding in the reranking stage is also similar to the parsing algorithm of their model 1. In order to capture grandsibling factors, our decoding calculates inside probablities for not the current head node but each pair of the node and its outgoing edges.

Titov and Henderson (2006) reported that the MBR approach could be applied to a projective dependency parser. In the field of SMT, for an approximation of MAP decoding, Li et al. (2009) proposed variational decoding and Kumar et al. (2009) presented hypergraph MBR decoding. Our variational model is inspired by the study of Li et al. (2009) and we apply it to a dependency parser in order to select better candidates with third-order information. We also propose an efficient algorithm to estimate the non-local third-order model structure.

7 Conclusions

In this paper, we propose a novel forest reranking algorithm for dependency parsing. Our reranking algorithm is a combination approach of generative reranking and variational decoding. The search algorithm in the reranking stage can be performed using dynamic programming algorithm. Our variational reranking is aimed at selecting a candidate from a forest, which is correct both in local and global. Our experimental results show more significant improvements than conventional approaches, such as k-best and forest generative reranking.

In the future, we plan to investigate more appropriate generative models for reranking. *PP-Attachment* is one of the most difficult problems for a natural language parser. We plan to examine to model such a complex structure (granduncle) (Goldberg and Elhadad, 2010) or higher-order structure than third-order for reranking which is computationally expensive for a baseline parser. As we mentioned in Section 5.4, we also plan to incorporate semi-supervised learning into our framework, which may potentially improve our reranking performance.

Acknowledgments

We would like to thank Graham Neubig and Masashi Shimbo for their helpful comments and to the anonymous reviewers for their effort of reviewing our paper and giving valuable comments. This work was supported in part by Grant-in-Aid for Japan Society for the Promotion of Science (JSPS) Research Fellowship for Young Scientists.

References

X. Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proc. the CoNLL-EMNLP*, pages 957–961.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. the 43rd ACL*, pages 173–180.

D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33:201–228.

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. the ICML*.

J. M. Eisner. 1996a. An empirical comparison of probability models for dependency grammar. In *Technical Report*, pages 1–18.

J. M. Eisner. 1996b. Three new probabilistic models for dependency parsing: An exploration. In *Proc. the 16th COLING*, pages 340–345.

Y. Goldberg and M. Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Proc. the HLT-NAACL*, pages 742–750.

L. Huang and D. Chiang. 2005. Better k-best parsing. In *Proc. the IWPT*, pages 53–64.

L. Huang and K. Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proc. the ACL*, pages 1077–1086.

L. Huang. 2006. Dynamic programming algorithms in semiring and hypergraph frameworks. *Qualification Exam Report*, pages 1–19. http://www.cis.upenn.edu/ lhuang3/wpe2/.

L. Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. the 46th ACL*, pages 586–594.

T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *Proc. the 48th ACL*, pages 1–11.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *Proc. the ACL*, pages 595–603.

S. Kumar, W. Macherey, C. Dyer, and F. Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proc. the 47th ACL*, pages 163–171.

Z. Li, J. Eisner, and S. Khudanpur. 2009. Variational decoding for statistical machine translation. In *Proc. the 47th ACL*, pages 593–601.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic cfg with latent annotations. In *Proc. the ACL*, pages 75–82.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. EACL*, pages 81–88.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. the 43rd ACL*, pages 91–98.

H. Mi and L. Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP*, pages 206–214.

M. Mohri. 2002. Semiring framework and algorithms for shortest-distance problems. *Automata, Languages and Combinatorics*, 7:321–350.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. the 41st ACL*, pages 160–167.

F. Sangati, W. Zuidema, and R. Bod. 2009. A generative re-ranking model for dependency parsing. In *Proc. the 11th IWPT*, pages 238–241.

J. Suzuki, H. Isozaki, X. Carreras, and M. Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proc. the EMNLP*, pages 551–560.

I. Titov and J. Henderson. 2006. Bayes risk minimization in natural language parsing. In *Technical Report*, pages 1–9.

Z. Tu, Y. Liu, Y. Hwang, Q. Liu, and S. Lin. 2010. Dependency forest for statistical machine translation. In *Proc. the 23rd COLING*, pages 1092–1100.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. the IWPT*, pages 195–206.