Feature Forest Models for Probabilistic HPSG Parsing

Yusuke Miyao*
University of Tokyo

Jun'ichi Tsujii**
University of Tokyo
University of Manchester

Probabilistic modeling of lexicalized grammars is difficult because these grammars exploit complicated data structures, such as typed feature structures. This prevents us from applying common methods of probabilistic modeling in which a complete structure is divided into substructures under the assumption of statistical independence among sub-structures. For example, part-of-speech tagging of a sentence is decomposed into tagging of each word, and CFG parsing is split into applications of CFG rules. These methods have relied on the structure of the target problem, namely lattices or trees, and cannot be applied to graph structures including typed feature structures.

This article proposes the feature forest model as a solution to the problem of probabilistic modeling of complex data structures including typed feature structures. The feature forest model provides a method for probabilistic modeling without the independence assumption when probabilistic events are represented with feature forests. Feature forests are generic data structures that represent ambiguous trees in a packed forest structure. Feature forest models are maximum entropy models defined over feature forests. A dynamic programming algorithm is proposed for maximum entropy estimation without unpacking feature forests. Thus probabilistic modeling of any data structures is possible when they are represented by feature forests.

This article also describes methods for representing HPSG syntactic structures and predicate–argument structures with feature forests. Hence, we describe a complete strategy for developing probabilistic models for HPSG parsing. The effectiveness of the proposed methods is empirically evaluated through parsing experiments on the Penn Treebank, and the promise of applicability to parsing of real-world sentences is discussed.

1. Introduction

Following the successful development of wide-coverage lexicalized grammars (Riezler et al. 2000; Hockenmaier and Steedman 2002; Burke et al. 2004; Miyao, Ninomiya, and

* Department of Computer Science, University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033 Japan. E-mail: yusuke@is.s.u-tokyo.ac.jp.
** Department of Computer Science, University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033 Japan. E-mail: tsujii@is.s.u-tokyo.ac.jp.

Submission received: 11 June 2006; revised submission received: 2 March 2007; accepted for publication: 5 May 2007.

Tsujii 2005), statistical modeling of these grammars is attracting considerable attention. This is because natural language processing applications usually require disambiguated or ranked parse results, and statistical modeling of syntactic/semantic preference is one of the most promising methods for disambiguation.

The focus of this article is the problem of probabilistic modeling of wide-coverage HPSG parsing. Although previous studies have proposed maximum entropy models (Berger, Della Pietra, and Della Pietra 1996) of HPSG-style parse trees (Oepen, Toutanova, et al. 2002b; Toutanova and Manning 2002; Baldridge and Osborne 2003; Malouf and van Noord 2004), the straightforward application of maximum entropy models to wide-coverage HPSG parsing is infeasible because estimation of maximum entropy models is computationally expensive, especially when targeting wide-coverage parsing. In general, complete structures, such as transition sequences in Markov models and parse trees, have an exponential number of ambiguities. This causes an exponential explosion when estimating the parameters of maximum entropy models. We therefore require solutions to make model estimation tractable.

This article first proposes feature forest models, which are a general solution to the problem of maximum entropy modeling of tree structures (Miyao and Tsujii 2002). Our algorithm avoids exponential explosion by representing probabilistic events with feature forests, which are packed representations of tree structures. When complete structures are represented with feature forests of a tractable size, the parameters of maximum entropy models are efficiently estimated without unpacking the feature forests. This is due to dynamic programming similar to the algorithm for computing inside/outside probabilities in PCFG parsing.

The latter half of this article (Section 4) is on the application of feature forest models to disambiguation in wide-coverage HPSG parsing. We describe methods for representing HPSG parse trees and predicate–argument structures using feature forests (Miyao, Ninomiya, and Tsujii 2003; Miyao and Tsujii 2003, 2005). Together with the parameter estimation algorithm for feature forest models, these methods constitute a complete procedure for the probabilistic modeling of wide-coverage HPSG parsing.

The methods we propose here were applied to an English HPSG parser, Enju (Tsujii Laboratory 2004). We report on an extensive evaluation of the parser through parsing experiments on the *Wall Street Journal* portion of the Penn Treebank (Marcus et al. 1994).

The content of this article is an extended version of our earlier work reported in Miyao and Tsujii (2002, 2003, 2005) and Miyao, Ninomiya, and Tsujii (2003). The major contribution of this article is a strict mathematical definition of the feature forest model and the parameter estimation algorithm, which are substantially refined and extended from Miyao and Tsujii (2002). Another contribution is that this article thoroughly discusses the relationships between the feature forest model and its application to HPSG parsing. We also provide an extensive empirical evaluation of the resulting HPSG parsing approach using real-world text.

Section 2 discusses a problem of conventional probabilistic models for lexicalized grammars. Section 3 proposes feature forest models for solving this problem. Section 4 describes the application of feature forest models to probabilistic HPSG parsing. Section 5 presents an empirical evaluation of probabilistic HPSG parsing, and Section 6 introduces research related to our proposals. Section 7 concludes.

2. Problem

Maximum entropy models (Berger, Della Pietra, and Della Pietra 1996) are now becoming the de facto standard approach for disambiguation models for lexicalized or

feature structure grammars (Johnson et al. 1999; Riezler et al. 2000, 2002; Geman and Johnson 2002; Clark and Curran 2003, 2004b; Kaplan et al. 2004; Carroll and Oepen 2005). Previous studies on probabilistic models for HPSG (Oepen, Toutanova et al. 2002; Toutanova and Manning 2002; Baldridge and Osborne 2003; Malouf and van Noord 2004) have also adopted log-linear models. This is because these grammar formalisms exploit feature structures to represent linguistic constraints. Such constraints are known to introduce inconsistencies in probabilistic models estimated using simple relative frequency, as discussed in Abney (1997). The maximum entropy model is a reasonable choice for credible probabilistic models. It also allows various overlapping features to be incorporated, and we can expect higher accuracy in disambiguation.

A maximum entropy model gives a probabilistic distribution that maximizes the likelihood of training data under given feature functions. Given training data $E = \{\langle x, y \rangle\}$, a maximum entropy model gives conditional probability $p(y|x)$ as follows.

Definition 1 (Maximum entropy model)
A maximum entropy model is defined as the solution of the following optimization problem.

$$p_M(y|x) = \operatorname*{argmax}_p \left\{ - \sum_{\langle x,y \rangle \in E} \tilde{p}(x,y) \log p(y|x) \right\}$$

where:

$$p(y|x) = \frac{1}{Z(x)} \exp\left(\sum_i \lambda_i f_i(x,y) \right)$$

$$Z(x) = \sum_{y \in Y(x)} \exp\left(\sum_i \lambda_i f_i(x,y) \right)$$

In this definition, $\tilde{p}(x,y)$ is the relative frequency of $\langle x, y \rangle$ in the training data. f_i is a **feature function**, which represents a characteristic of probabilistic events by mapping an event into a real value. λ_i is the model parameter of a corresponding feature function f_i, and is determined so as to maximize the likelihood of the training data (i.e., the optimization in this definition). $Y(x)$ is a set of y for given x; for example, in parsing, x is a given sentence and $Y(x)$ is a parse forest for x. An advantage of maximum entropy models is that feature functions can represent any characteristics of events. That is, independence assumptions are unnecessary for the design of feature functions. Hence, this method provides a principled solution for the estimation of consistent probabilistic distributions over feature structure grammars.

The remaining issue is how to estimate parameters. Several numerical algorithms, such as Generalized Iterative Scaling (GIS) (Darroch and Ratcliff 1972), Improved Iterative Scaling (IIS) (Della Pietra, Della Pietra, and Lafferty 1997), and the Limited-memory Broyden-Fletcher-Goldfarb-Shanno method (L-BFGS) (Nocedal and Wright 1999), have been proposed for parameter estimation. Although the algorithm proposed in the present article is applicable to all of the above algorithms, we used L-BFGS for experiments.

However, a computational problem arises in these parameter estimation algorithms. The size of $Y(x)$ (i.e., the number of parse trees for a sentence) is generally

very large. This is because local ambiguities in parse trees potentially cause exponential growth in the number of structures assigned to sub-sequences of words, resulting in billions of structures for whole sentences. For example, when we apply rewriting rule $\mathsf{S} \rightarrow \mathsf{NP\ VP}$, and the left NP and the right VP, respectively, have n and m ambiguous subtrees, the result of the rule application generates $n \times m$ trees.

This is problematic because the complexity of parameter estimation is proportional to the size of $Y(x)$. The cost of the parameter estimation algorithms is bound by the computation of **model expectation**, μ_i, given as (Malouf 2002):

$$\begin{aligned}\mu_i &= \sum_{x \in \mathcal{X}} \tilde{p}(x) \sum_{y \in Y(x)} f_i(x,y) p(y|x) \\ &= \sum_{x \in \mathcal{X}} \tilde{p}(x) \sum_{y \in Y(x)} f_i(x,y) \frac{1}{Z(x)} \exp\left(\sum_j \lambda_j f_j(x,y)\right) \end{aligned} \quad (1)$$

As shown in this definition, the computation of model expectation requires the summation over $Y(x)$ for every x in the training data. The complexity of the overall estimation algorithm is $O(|\tilde{Y}||\tilde{F}||E|)$, where $|\tilde{Y}|$ and $|\tilde{F}|$ are the average numbers of y and activated features for an event, respectively, and $|E|$ is the number of events. When $Y(x)$ grows exponentially, the parameter estimation becomes intractable.

In PCFGs, the problem of computing probabilities of parse trees is avoided by using a dynamic programming algorithm for computing inside/outside probabilities (Baker 1979). With the algorithm, the computation becomes tractable. We can expect that the same approach would be effective for maximum entropy models as well.

This notion yields a novel algorithm for parameter estimation for maximum entropy models, as described in the next section.

3. Feature Forest Model

Our solution to the problem is a dynamic programming algorithm for computing **inside/outside α-products**. Inside/outside α-products roughly correspond to inside/outside probabilities in PCFGs. In maximum entropy models, a probability is defined as a normalized product of $\alpha_j^{f_j} (= \exp(\lambda_j f_j))$. Hence, similar to the algorithm of computing inside/outside probabilities, we can compute $\exp\left(\sum_j \lambda_j f_j\right)$, which we define as the **α-product**, for each node in a tree structure. If we can compute α-products at a tractable cost, the model expectation μ_i is also computed at a tractable cost.

We first define the notion of a **feature forest**, a packed representation of a set of an exponential number of tree structures. Feature forests correspond to packed charts in CFG parsing. Because feature forests are generalized representations of forest structures, the notion is not only applicable to syntactic parsing but also to sequence tagging, such as POS tagging and named entity recognition (which will be discussed in Section 6). We then define inside/outside α-products that represent the α-products of partial structures of a feature forest. Inside α-products correspond to inside probabilities in PCFG, and represent the summation of α-products of the daughter sub-trees. Outside α-products correspond to outside probabilities in PCFG, and represent the summation of α-products in the upper part of the feature forest. Both can be computed incrementally by a dynamic programming algorithm similar to the algorithm for computing

inside/outside probabilities in PCFG. Given inside/outside α-products of all nodes in a feature forest, the model expectation μ_i is easily computed by multiplying them for each node.

3.1 Feature Forest

To describe the algorithm, we first define the notion of a feature forest, the generalized representation of features in a packed forest structure. Feature forests are used for enumerating possible structures of events, that is, they correspond to $Y(x)$ in Equation 1.

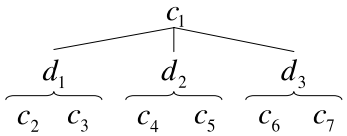
Definition 2 (Feature forest)
A feature forest Φ is a tuple $\langle C, D, r, \gamma, \delta \rangle$, where:

- C is a set of conjunctive nodes,
- D is a set of disjunctive nodes,
- r is the root node: $r \in C$,
- $\gamma : D \mapsto 2^C$ is a conjunctive daughter function,
- $\delta : C \mapsto 2^D$ is a disjunctive daughter function.

We denote a feature forest for x as $\Phi(x)$. For example, $\Phi(x)$ can represent the set of all possible tag sequences of a given sentence x, or the set of all parse trees of x. A feature forest is an acyclic graph, and unpacked structures extracted from a feature forest are trees. We also assume that terminal nodes of feature forests are conjunctive nodes. That is, disjunctive nodes must have daughters (i.e., $\gamma(d) \neq \emptyset$ for all $d \in D$).

A feature forest represents a set of trees of conjunctive nodes in a packed structure. Conjunctive nodes correspond to entities such as states in Markov chains and nodes in CFG trees. Feature functions are assigned to conjunctive nodes and express their characteristics. Disjunctive nodes are for enumerating alternative choices. Conjunctive/disjunctive daughter functions represent immediate relations of conjunctive and disjunctive nodes. By selecting a conjunctive node as a child of each disjunctive node, we can extract a tree consisting of conjunctive nodes from a feature forest.

Figure 1 shows an example of a feature forest. Each disjunctive node enumerates alternative nodes, which are conjunctive nodes. Each conjunctive node has disjunctive



$C = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7\}$
$D = \{d_1, d_2, d_3\}$
$r = c_1$
$\gamma(d_1) = \{c_2, c_3\},\ \gamma(d_2) = \{c_4, c_5\},\ \gamma(d_3) = \{c_6, c_7\}$
$\delta(c_1) = \{d_1, d_2, d_3\},\ \delta(c_2) = \{\},\ \delta(c_3) = \{\},\ \delta(c_4) = \{\},$
$\delta(c_5) = \{\},\ \delta(c_6) = \{\},\ \delta(c_7) = \{\}$

Figure 1
A feature forest.

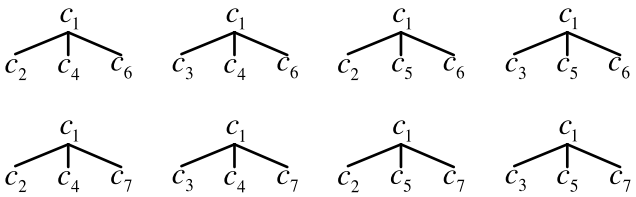


Figure 2
Unpacked trees.

nodes as its daughters. The feature forest in Figure 1 represents a set of $2 \times 2 \times 2 = 8$ unpacked trees shown in Figure 2. For example, by selecting the left-most conjunctive node at each disjunctive node, we extract an unpacked tree (c_1, c_2, c_4, c_6). An unpacked tree is represented as a set of conjunctive nodes. Generally, a feature forest represents an exponential number of trees with a polynomial number of nodes. Thus, complete structures, such as tag sequences and parse trees with ambiguities, can be represented in a tractable form.

Feature functions are defined over conjunctive nodes.[1]

Definition 3 (Feature function for feature forests)
A feature function for a feature forest is:

$$f_i : C \mapsto \mathbf{R}$$

Hence, together with feature functions, a feature forest represents a set of trees of features.

Feature forests may be regarded as a packed chart in CFG parsing. Although feature forests have the same structure as PCFG parse forests, nodes in feature forests do not necessarily correspond to nodes in PCFG parse forests. In fact, in Sections 4.2 and 4.3, we will demonstrate that syntactic structures and predicate–argument structures in HPSG can be represented with tractable-size feature forests. The actual interpretation of a node in a feature forest may thus be ignored in the following discussion. Our algorithm is applicable whenever feature forests are of a tractable size. The descriptive power of feature forests will be discussed again in Section 6.

As mentioned, a feature forest is a packed representation of trees of features. We first define model expectations, μ_i, on a set of unpacked trees, and then show that they can be computed without unpacking feature forests. We denote an unpacked tree as a set, $\mathbf{c} \subseteq C$, of conjunctive nodes. Our concern is only the set of features associated with each conjunctive node, and the shape of the tree structure is irrelevant to the computation of probabilities of unpacked trees. Hence, we do not distinguish an unpacked tree from a set of conjunctive nodes.

The collection of unpacked trees represented by a feature forest is defined as a multi-set of unpacked trees because we allow multiple occurrences of equivalent unpacked

1 Feature functions may also be conditioned on x. In this case, feature functions can be written as $f_i(c, x)$. For simplicity, we omit x in the following discussion.

trees in a feature forest.[2] Given multisets of unpacked trees, A, B, we define the union and the product as follows.

$$A \oplus B \equiv A \cup B$$
$$A \otimes B \equiv \{\mathbf{a} \cup \mathbf{b} | \mathbf{a} \in A, \mathbf{b} \in B\}$$

Intuitively, the first operation is a collection of trees, and the second lists all combinations of trees in A and B. It is trivial that they satisfy commutative, associative, and distributive laws.

$$A \oplus B = B \oplus A$$
$$A \otimes B = B \otimes A$$
$$A \oplus (B \oplus C) = (A \oplus B) \oplus C$$
$$A \otimes (B \otimes C) = (A \otimes B) \otimes C$$
$$A \otimes (B \oplus C) = (A \otimes B) \oplus (A \otimes C)$$

We denote a set of unpacked trees rooted at node $n \in C \cup D$ as $\Omega(n)$. $\Omega(n)$ is defined recursively. For a terminal node $c \in C$, obviously $\Omega(c) = \{\{c\}\}$. For an internal conjunctive node $c \in C$, an unpacked tree is a combination of trees, each of which is selected from a disjunctive daughter. Hence, a set of all unpacked trees is represented as a product of trees from disjunctive daughters.

$$\Omega(c) = \{\{c\}\} \otimes \bigotimes_{d \in \delta(c)} \Omega(d)$$

A disjunctive node $d \in D$ represents alternatives of packed trees, and obviously a set of its unpacked trees is represented as a union of the daughter trees, that is, $\Omega(d) = \bigoplus_{c \in \gamma(d)} \Omega(c)$.

To summarize, a set of unpacked trees is defined formally as follows.

Definition 4 (Unpacked tree)
Given a feature forest $\Phi = \langle C, D, r, \gamma, \delta \rangle$, a set $\Omega(n)$ of unpacked trees rooted at node $n \in C \cup D$ is defined recursively as follows.

- If $n \in C$ is a terminal, that is, $\delta(n) = \emptyset$,

$$\Omega(n) \equiv \{\{n\}\}$$

- If $n \in C$,

$$\Omega(n) \equiv \{\{n\}\} \otimes \bigotimes_{d \in \delta(n)} \Omega(d)$$

2 In fact, no feature forests include equivalent unpacked trees if no disjunctive nodes have identical daughter nodes. Thus we may define a set of unpacked trees as an ordinary set, although the details are omitted here for simplicity.

- If $n \in D$,

$$\Omega(n) \equiv \bigoplus_{c \in \gamma(n)} \Omega(c)$$

Feature forests are directed acyclic graphs and, as such, this definition does not include a loop. Hence, $\Omega(n)$ is properly defined.

A set of all unpacked trees is then represented by $\Omega(r)$; henceforth, we denote $\Omega(r)$ as $\Omega(\Phi)$, or just Ω when it is not confusing in context. Figure 3 shows $\Omega(\Phi)$ of the feature forest in Figure 1. Following Definition 4, the first element of each set is the root node, c_1, and the rest are elements of the product of $\{c_2, c_3\}$, $\{c_4, c_5\}$, and $\{c_6, c_7\}$. Each set in Figure 3 corresponds to a tree in Figure 2.

Given this formalization, the feature function for an unpacked tree is defined as follows.

Definition 5 (Feature function for unpacked tree)
The feature function f_i for an unpacked tree, $\mathbf{c} \in \Omega(\Phi)$ is defined as:

$$f_i(\mathbf{c}) = \sum_{c \in \mathbf{c}} f_i(c)$$

Because $\mathbf{c} \in \Omega(\Phi)$ corresponds to y of the conventional maximum entropy model, this function substitutes for $f_i(x, y)$ in the conventional model. Once a feature function for an unpacked tree is given, a model expectation is defined as in the traditional model.

Definition 6 (Model expectation of feature forests)
The model expectation μ_i for a set of feature forests $\{\Phi(x)\}$ is defined as:

$$\begin{aligned}
\mu_i &= \sum_{x \in \mathcal{X}} \tilde{p}(x) \sum_{\mathbf{c} \in \Omega(\Phi(x))} f_i(\mathbf{c}) p(\mathbf{c}|x) \\
&= \sum_{x \in \mathcal{X}} \tilde{p}(x) \sum_{\mathbf{c} \in \Omega(\Phi(x))} f_i(\mathbf{c}) \frac{1}{Z(x)} \exp\left(\sum_j \lambda_j f_j(\mathbf{c})\right) \\
\text{where} \qquad Z(x) &= \sum_{\mathbf{c} \in \Omega(\Phi(x))} \exp\left(\sum_j \lambda_j f_j(\mathbf{c})\right)
\end{aligned}$$

It is evident that the naive computation of model expectations requires exponential time complexity because the number of unpacked trees (i.e., $|\Omega(\Phi)|$) is exponentially related to the number of nodes in the feature forest Φ. We therefore need an algorithm for computing model expectations without unpacking a feature forest.

$$\begin{aligned}
\Omega(\Phi) = \{\ &\{c_1, c_2, c_4, c_6\}, \{c_1, c_3, c_4, c_6\}, \{c_1, c_2, c_5, c_6\}, \{c_1, c_3, c_5, c_6\}, \\
&\{c_1, c_2, c_4, c_7\}, \{c_1, c_3, c_4, c_7\}, \{c_1, c_2, c_5, c_7\}, \{c_1, c_3, c_5, c_7\}\ \}
\end{aligned}$$

Figure 3
Unpacked trees represented as sets of conjunctive nodes.

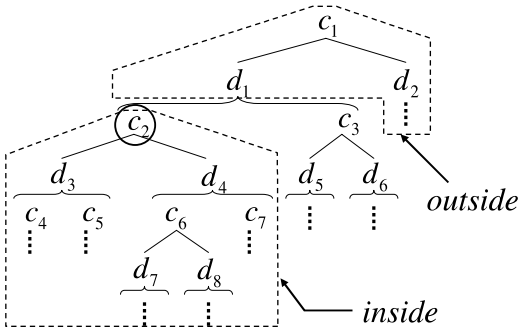


Figure 4
Inside/outside at node c_2 in a feature forest.

3.2 Dynamic Programming

To efficiently compute model expectations, we incorporate an approach similar to the dynamic programming algorithm for computing inside/outside probabilities in PCFGs. We first define the notion of inside/outside of a feature forest. Figure 4 illustrates this concept, which is similar to the analogous concept in PCFGs.[3] Inside denotes a set of partial trees (sets of conjunctive nodes) derived from node c_2. Outside denotes a set of partial trees that derive node c_2. That is, outside trees are partial trees of complements of inside trees.

We denote a set of inside trees at node n as $\iota(n)$, and that of outside trees as $o(n)$.

Definition 7 (Inside trees)
We define a set $\iota(n)$ of inside trees rooted at node $n \in C \cup D$ as a set of unpacked trees rooted at n.

$$\iota(n) \equiv \Omega(n)$$

Definition 8 (Outside trees)
We define a set $o(n)$ of outside trees rooted at node $n \in C \cup D$ as follows.

$$o(r) \equiv \{\emptyset\}$$

$$o(c) \equiv \bigoplus_{d \in \gamma^{-1}(c)} o(d)$$

$$o(d) \equiv \bigoplus_{c \in \delta^{-1}(d)} \left\{ \{\{c\}\} \otimes o(c) \otimes \bigotimes_{d' \in \delta(c), d' \neq d} \iota(d') \right\}$$

3 A node may have multiple outside trees in general as in the case of CFGs, although Figure 4 shows only one outside tree of c_2 for simplicity.

In the definition, γ^{-1} and δ^{-1} denote mothers of conjunctive and disjunctive nodes, respectively. Formally,

$$\gamma^{-1}(c) \equiv \{d | c \in \gamma(d)\}$$
$$\delta^{-1}(d) \equiv \{c | d \in \delta(c)\}$$

Next, inside/outside α-products are defined for conjunctive and disjunctive nodes. The inside (or outside) α-products are the summation of $\exp\left(\sum_j \lambda_j f_j(\boldsymbol{c})\right)$ of all inside (or outside) trees c.

Definition 9 (Inside/outside α-product)
An inside α-product at conjunctive node $c \in C$ is

$$\varphi_c = \sum_{\mathbf{c} \in \iota(c)} \exp\left(\sum_j \lambda_j f_j(\mathbf{c})\right)$$

An outside α-product is

$$\psi_c = \sum_{\mathbf{c} \in o(c)} \exp\left(\sum_j \lambda_j f_j(\mathbf{c})\right)$$

Similarly, inside/outside α-products at disjunctive node $d \in D$ are defined as follows:

$$\varphi_d = \sum_{\mathbf{c} \in \iota(d)} \exp\left(\sum_j \lambda_j f_j(\mathbf{c})\right)$$

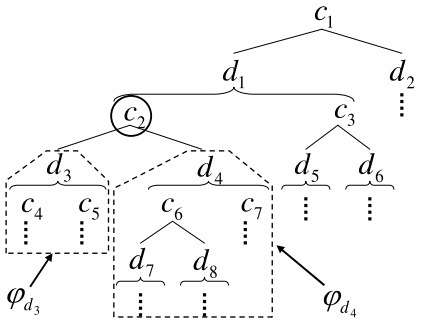
$$\psi_d = \sum_{\mathbf{c} \in o(d)} \exp\left(\sum_j \lambda_j f_j(\mathbf{c})\right)$$

We can derive that the model expectations of a feature forest are computed as the product of the inside and outside α-products.

Theorem 1 (Model expectation of feature forests)
The model expectation μ_i of a feature forest $\Phi(x) = \langle C_x, D_x, r_x, \gamma_x, \delta_x \rangle$ is computed as the product of inside and outside α-products as follows:

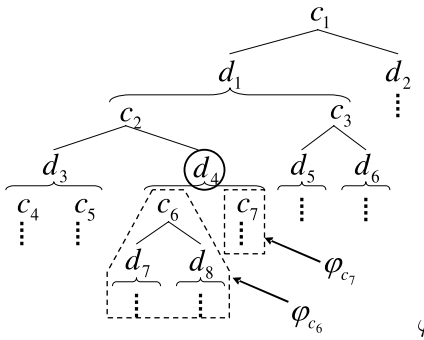
$$\mu_i = \sum_{x \in \mathcal{X}} \tilde{p}(x) \frac{1}{Z(x)} \sum_{c \in C_x} f_i(c) \varphi_c \psi_c$$

$$\text{where} \qquad Z(x) = \varphi_{r_x}$$



$$\varphi_{c_2} = \varphi_{d_3}\varphi_{d_4} \exp\left(\sum_j \lambda_j f_j(c_2)\right)$$

Figure 5
Incremental computation of inside α-products at conjunctive node c_2.



$$\varphi_{d_4} = \varphi_{c_6} + \varphi_{c_7}$$

Figure 6
Incremental computation of inside α-products at disjunctive node d_4.

This equation shows a method for efficiently computing model expectations by traversing conjunctive nodes without unpacking the forest, if the inside/outside α-products are given. The remaining issue is how to efficiently compute inside/outside α-products.

Fortunately, inside/outside α-products can be incrementally computed by dynamic programming without unpacking feature forests. Figure 5 shows the process of computing the inside α-product at a conjunctive node from the inside α-products of its daughter nodes. Because the inside of a conjunctive node is a set of the combinations of all of its descendants, the α-product is computed by multiplying the α-products of the daughter trees. The following equation is derived.

$$\varphi_c = \left(\prod_{d \in \delta(c)} \varphi_d\right) \exp\left(\sum_j \lambda_j f_j(c)\right)$$

The inside of a disjunctive node is the collection of the inside trees of its daughter nodes. Hence, the inside α-product at disjunctive node $d \in D$ is computed as follows (Figure 6).

$$\varphi_d = \sum_{c \in \gamma(d)} \varphi_c$$

Theorem 2 (Inside α-product)
The inside α-product φ_c at a conjunctive node c is computed by the following equation if φ_d is given for all daughter disjunctive nodes $d \in \delta(c)$.

$$\varphi_c = \left(\prod_{d \in \delta(c)} \varphi_d \right) \exp \left(\sum_j \lambda_j f_j(c) \right)$$

The inside α-product φ_d at a disjunctive node d is computed by the following equation if φ_c is given for all daughter conjunctive nodes $c \in \gamma(d)$.

$$\varphi_d = \sum_{c \in \gamma(d)} \varphi_c$$

The outside of a disjunctive node is equivalent to the outside of its daughter nodes. Hence, the outside α-product of a disjunctive node is propagated to its daughter conjunctive nodes (Figure 7).

$$\psi_c = \sum_{\{d | c \in \gamma(d)\}} \psi_d$$

The computation of the outside α-product of a disjunctive node is somewhat complicated. As shown in Figure 8, the outside trees of a disjunctive node are all combinations of

- the outside trees of the mother nodes, and
- the inside trees of the sister nodes.

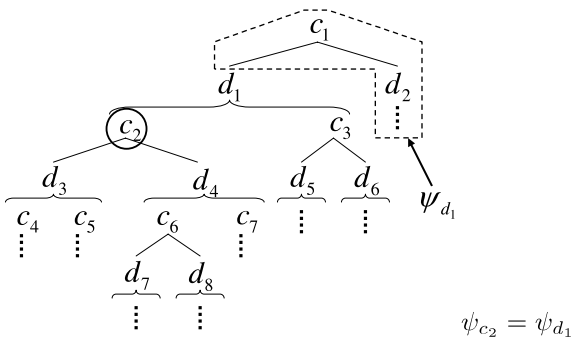
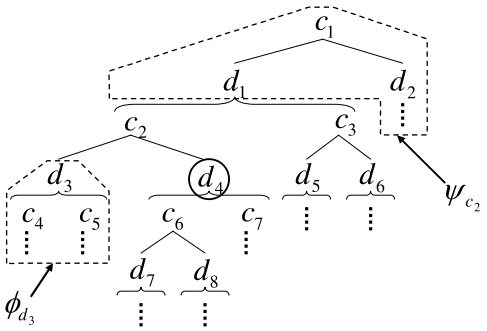


Figure 7
Incremental computation of outside α-products at conjunctive node c_2.



$$\psi_{d_4} = \psi_{c_2} \varphi_{d_3} \exp\left(\sum_j \lambda_j f_j(c_2)\right)$$

Figure 8
Incremental computation of outside α-products at disjunctive node d_4.

From this, we find:

$$\psi_d = \sum_{\{c|d\in\delta(c)\}} \left\{ \psi_c \exp\left(\sum_j \lambda_j f_j(c)\right) \prod_{\substack{d'\in\delta(c)\\ d'\neq d}} \varphi_{d'} \right\}$$

We finally find the following theorem for the computation of outside α-products.

Theorem 3 (Outside α-product)
The outside α-product ψ_c at conjunctive node c is computed by the following equation if ψ_d is given for all mother disjunctive nodes, that is, all d such that $c \in \gamma(d)$.

$$\psi_c = \sum_{\{d|c\in\gamma(d)\}} \psi_d$$

The outside α-product ψ_d at disjunctive node d is computed by the following equation if ψ_c is given for all mother conjunctive nodes, that is, all c such that $d \in \delta(c)$, and $\varphi_{d'}$ for all sibling disjunctive nodes d'.

$$\psi_d = \sum_{\{c|d\in\delta(c)\}} \left\{ \psi_c \exp\left(\sum_j \lambda_j f_j(c)\right) \prod_{\substack{d'\in\delta(c)\\ d'\neq d}} \varphi_{d'} \right\}$$

Figure 9 shows the overall algorithm for estimating the parameters, given a set of feature forests. The key point of the algorithm is to compute inside α-products φ and outside α-products ψ for each node in C, and not for all unpacked trees. The functions `inside_product` and `outside_product` compute φ and ψ efficiently by dynamic programming.

Note that the order in which nodes are traversed is important for incremental computation, although it is not shown in Figure 9. The computation for the daughter nodes and mother nodes must be completed before computing the inside and outside

Input: training data $E = \{\langle x_i, y_i \rangle\}$, feature forests $\{\Phi(x_i)\}$
feature functions $\boldsymbol{f} = \{f_i\}$, initial parameters $\boldsymbol{\lambda} = \{\lambda_i\}$
Output: optimal parameters $\boldsymbol{\lambda}$

foreach $\langle x, y \rangle \in E$
  $\{\varphi\} \leftarrow$ inside_product($\Phi(x)$)
  $\{\psi\} \leftarrow$ outside_product($\Phi(x)$)
  foreach $c \in C$
    foreach $f_i \in \boldsymbol{f}$ such that $f_i(c) \neq 0$
      $\mu_i \leftarrow \mu_i + f_i \varphi_c \psi_c$
    end_foreach
  end_foreach
  foreach $f_i \in \boldsymbol{f}$
    $\mu_i \leftarrow \frac{\mu_i}{\varphi_r}$
  end_foreach
end_foreach

function inside_product($\Phi(x) = \langle C, D, r, \gamma, \delta \rangle$)
  foreach $c \in C$, $d \in D$
    $$\varphi_c \leftarrow \prod_{d' \in \delta(c)} \varphi_{d'} \exp\left(\sum_j \lambda_j f_j(c)\right)$$
    $$\varphi_d \leftarrow \sum_{c' \in \gamma(d)} \varphi_{c'}$$
  end_foreach
  return $\{\varphi\}$

function outside_product($\Phi(x) = \langle C, D, r, \gamma, \delta \rangle$)
  foreach $c \in C$, $d \in D$
    $$\psi_c \leftarrow \sum_{\{d' | c \in \gamma(d')\}} \psi_{d'}$$
    $$\psi_d \leftarrow \sum_{\{c' | d \in \delta(c')\}} \left\{ \psi_{c'} \exp\left(\sum_j \lambda_j f_j(c')\right) \prod_{\substack{d' \in \delta(c') \\ d' \neq d}} \varphi_{d'} \right\}$$
  end_foreach
  return $\{\psi\}$

Figure 9
Algorithm for computing model expectations of feature forests.

α-products, respectively. This constraint is easily solved using any topological sort algorithm. A topological sort is applied once at the beginning. The result of the sorting does not affect the cost and the result of estimation. In our implementation, we assume that conjunctive/disjunctive nodes are already ordered from the root node in input data.

The complexity of this algorithm is $O((|\tilde{C}| + |\tilde{D}|)|\tilde{F}||E|)$, where $|\tilde{C}|$ and $|\tilde{D}|$ are the average numbers of conjunctive and disjunctive nodes, respectively. This is tractable when $|\tilde{C}|$ and $|\tilde{D}|$ are of a reasonable size. As noted in this section, the number of

nodes in a feature forest is usually polynomial even when that of the unpacked trees is exponential. Thus we can efficiently compute model expectations with polynomial computational complexity.

4. Probabilistic HPSG Parsing

Following previous studies on probabilistic models for HPSG (Oepen, Toutanova, et al. 2002; Toutanova and Manning 2002; Baldridge and Osborne 2003; Malouf and van Noord 2004), we apply a maximum entropy model to HPSG parse disambiguation. The probability, $p(t|\mathbf{w})$, of producing parse result t of a given sentence $\mathbf{w}$ is defined as

$$p(t|\mathbf{w}) = \frac{1}{Z_{\mathbf{w}}} p_0(t|\mathbf{w}) \exp\left(\sum_i \lambda_i f_i(t, \mathbf{w})\right)$$

where

$$Z_{\mathbf{w}} = \sum_{t' \in T(\mathbf{w})} p_0(t'|\mathbf{w}) \exp\left(\sum_i \lambda_i f_i(t', \mathbf{w})\right)$$

where $p_0(t|\mathbf{w})$ is a reference distribution (usually assumed to be a uniform distribution) and $T(\mathbf{w})$ is a set of parse candidates assigned to $\mathbf{w}$. The feature function $f_i(t, \mathbf{w})$ represents the characteristics of t and $\mathbf{w}$, and the corresponding model parameter λ_i is its weight. Model parameters that maximize the log-likelihood of the training data are computed using a numerical optimization method (Malouf 2002).

Estimation of the model requires a set of pairs $\langle t_{\mathbf{w}}, T(\mathbf{w})\rangle$, where $t_{\mathbf{w}}$ is the correct parse for a sentence $\mathbf{w}$. Whereas $t_{\mathbf{w}}$ is provided by a treebank, $T(\mathbf{w})$ has to be computed by parsing each $\mathbf{w}$ in the treebank. Previous studies assumed $T(\mathbf{w})$ could be enumerated; however, this assumption is impractical because the size of $T(\mathbf{w})$ is exponentially related to the length of $\mathbf{w}$.

Our solution here is to apply the feature forest model of Section 3 to the probabilistic modeling of HPSG parsing. Section 4.1 briefly introduces HPSG. Section 4.2 and 4.3 describe how to represent HPSG parse trees and predicate–argument structures by feature forests. Together with the parameter estimation algorithm in Section 3, these methods constitute a complete method for probabilistic disambiguation. We also address a method for accelerating the construction of feature forests for all treebank sentences in Section 4.4. The design of feature functions will be given in Section 4.5.

4.1 HPSG

HPSG (Pollard and Sag 1994; Sag, Wasow, and Bender 2003) is a syntactic theory that follows the lexicalist framework. In HPSG, linguistic entities, such as words and phrases, are denoted by **signs**, which are represented by typed feature structures (Carpenter 1992). Signs are a formal representation of combinations of phonological forms and syntactic/semantic structures, and express which phonological form signifies which syntactic/semantic structure. Figure 10 shows the lexical sign for *loves*. The geometry of signs follows Pollard and Sag: HEAD represents the part-of-speech of the head word, MOD denotes modifiee constraints, and SPR, SUBJ, and COMPS describe constraints of a specifier, a syntactic subject, and complements, respectively. CONT denotes the

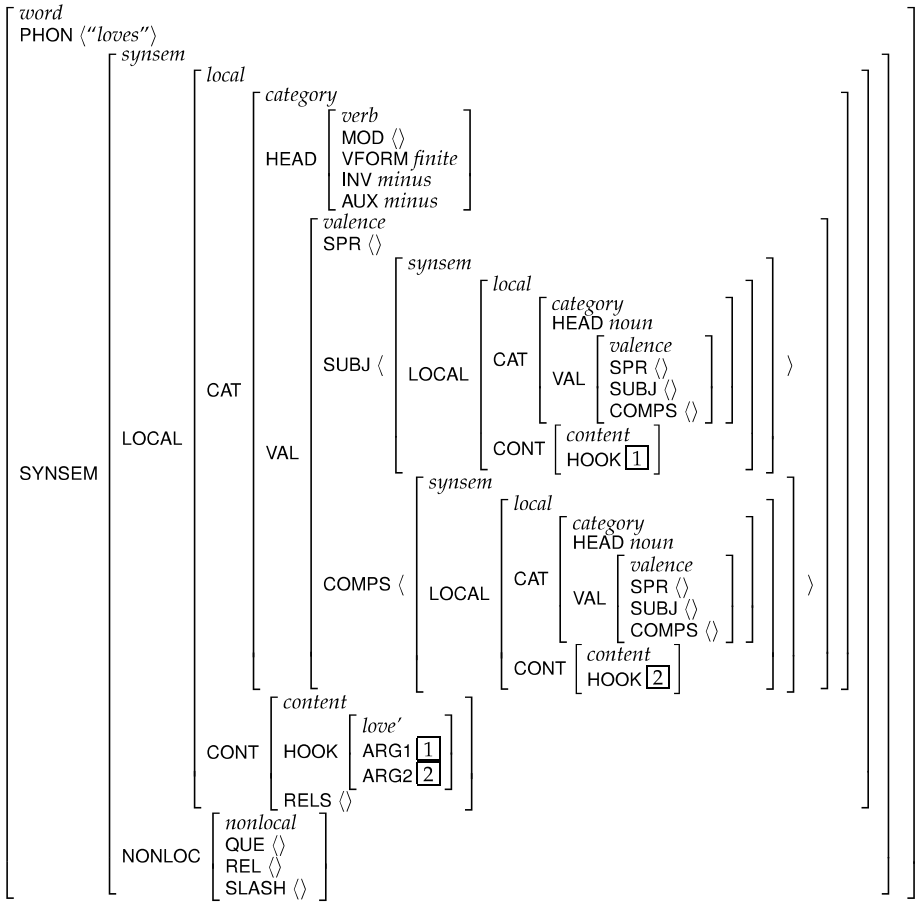


Figure 10
Lexical entry for the transitive verb *loves*.

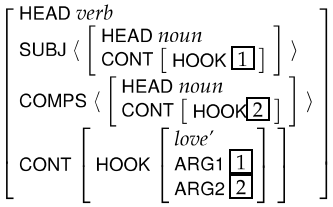


Figure 11
Simplified representation of the lexical entry in Figure 10.

predicate–argument structure of a phrase/sentence. The notation of CONT in this article is borrowed from that of Minimal Recursion Semantics (Copestake et al. 2006): HOOK represents a structure accessed by other phrases, and RELS describes the remaining structure of the semantics. In what follows, we represent signs in a reduced form as shown in Figure 11, because of the large size of typical HPSG signs, which often include information not immediately relevant to the point being discussed. We will only show attributes that are relevant to an explanation, expecting that readers can fill in the values of suppressed attributes.

In our actual implementation of the HPSG grammar, lexical/phrasal signs contain additional attributes that are not defined in the standard HPSG theory but are used by a disambiguation model. Examples include the surface form of lexical heads, and the type of lexical entry assigned to lexical heads, which are respectively used for computing the features WORD and LE introduced in Section 4.5. By incorporating additional attributes into signs, we can straightforwardly compute feature functions for each sign. This allows for a simple mapping between a parsing chart and a feature forest as described subsequently. However, this might increase the size of parse forests and therefore decrease parsing efficiency, because differences between additional attributes interfere with equivalence relations for ambiguity packing.

4.2 Packed Representation of HPSG Parse Trees

We represent an HPSG parse tree with a set of tuples $\langle m, l, r \rangle$, where m, l, and r are the signs of the mother, left daughter, and right daughter, respectively.[4] In chart parsing, partial parse candidates are stored in a *chart*, in which phrasal signs are identified and packed into equivalence classes if they are judged to be equivalent and dominate the same word sequences. A set of parse trees is then represented as a set of relations among equivalence classes.[5]

Figure 12 shows a chart for parsing *he saw a girl with a telescope*, where the modifiee of *with* is ambiguous (*saw* or *girl*). Each feature structure expresses an equivalence class, and the arrows represent immediate-dominance relations. The phrase, *saw a girl with a telescope*, has two trees (A in the figure). Because the signs of the top-most nodes are equivalent, they are packed into an equivalence class. The ambiguity is represented as the two pairs of arrows leaving the node A.

A set of HPSG parse trees is represented in a chart as a tuple $\langle E, E_r, \alpha \rangle$, where E is a set of equivalence classes, $E_r \subseteq E$ is a set of root nodes, and $\alpha : E \rightarrow 2^{E \times E}$ is a function to represent immediate-dominance relations.

Our representation of a chart can be interpreted as an instance of a feature forest. We map the tuple $\langle e_m, e_l, e_r \rangle$, which corresponds to $\langle m, l, r \rangle$, into a conjunctive node. Figure 13 shows (a part of) the HPSG parse trees in Figure 12 represented as a feature forest. Square boxes (c_i) are conjunctive nodes, and d_i disjunctive nodes. A solid arrow represents a disjunctive daughter function, and a dotted line expresses a conjunctive daughter function.

Formally, a chart $\langle E, E_r, \alpha \rangle$ is mapped into a feature forest $\langle C, D, R, \gamma, \delta \rangle$ as follows.[6]

- $C = \{\langle e_m, e_l, e_r \rangle | e_m \in E \wedge (e_l, e_r) \in \alpha(e_m)\} \cup \{w | w \in \mathbf{w}\}$
- $D = E$
- $R = \{\langle e_m, e_l, e_r \rangle | e_m \in E_r \wedge \langle e_m, e_l, e_r \rangle \in C\}$

4 For simplicity, only binary trees are considered. Extension to unary and n-ary ($n > 2$) trees is trivial.

5 We assume that CONT and DTRS (a feature used to represent daughter signs) are *restricted* (Shieber 1985), and we will discuss a method for encoding CONT in a feature forest in Section 4.3. We also assume that parse trees are packed according to equivalence relations rather than subsumption relations (Oepen and Carroll 2000). We cannot simply map parse forests packed under subsumption into feature forests, because they over-generate possible unpacked trees.

6 For ease of explanation, the definition of the root node is different from the original definition given in Section 3. In this section, we define R as a set of conjunctive nodes rather than a single node r. The definition here is translated into the original definition by introducing a dummy root node r' that has no features and only one disjunctive daughter whose daughters are R.

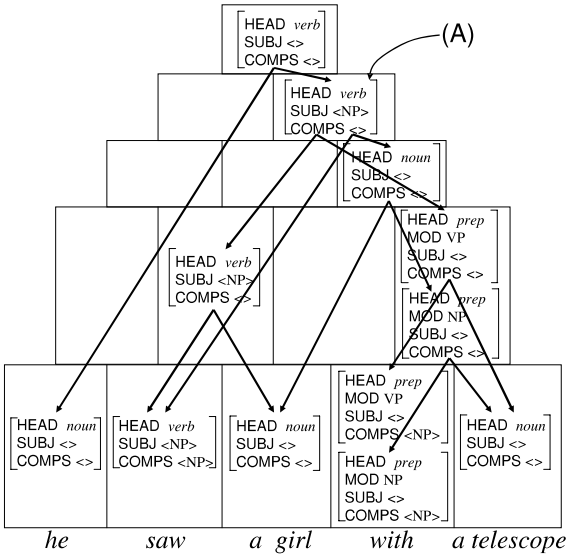


Figure 12
Chart for parsing *he saw a girl with a telescope*.

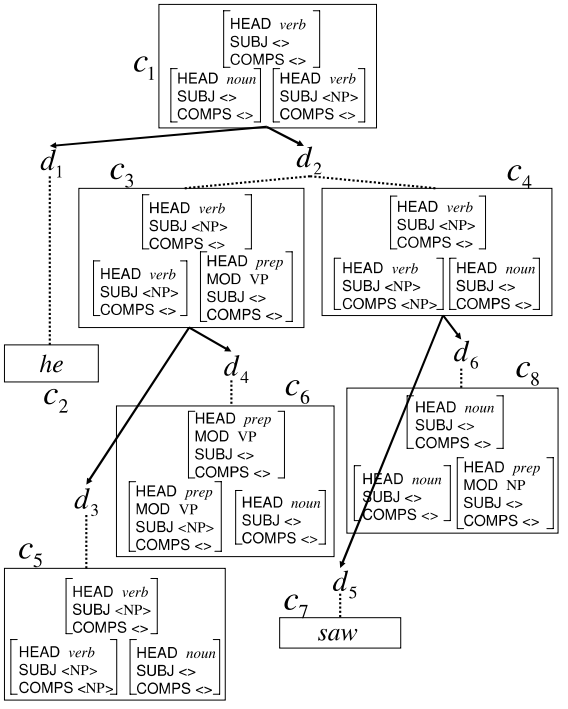


Figure 13
Feature forest representation of HPSG parse trees in Figure 12.

- $\gamma(e_m) = \begin{cases} \{\langle e_m, e_l, e_r\rangle | (e_l, e_r) \in \alpha(e_m)\} & \text{if } \alpha(e_m) \neq \emptyset \\ \{w | e_m \text{ is a lexical entry for } w\} & \text{otherwise} \end{cases}$
- $\delta(c) = \begin{cases} \{e_l, e_r\} & \text{if } c = \langle e_m, e_l, e_r\rangle \\ \emptyset & \text{if } c \in \mathbf{w} \end{cases}$

One may claim that restricting the domain of feature functions to $\langle e_m, e_l, e_r\rangle$ limits the flexibility of feature design. Although this is true to some extent, it does not necessarily mean the impossibility of incorporating features on nonlocal dependencies into the model. This is because a feature forest model does not assume probabilistic independence of conjunctive nodes. This means that we can unpack a part of the forest without changing the model. Actually, we successfully developed a probabilistic model including features on nonlocal predicate–argument dependencies, as described subsequently.

4.3 Packed Representation of Predicate–Argument Structures

With the method previously described, we can represent an HPSG parsing chart with a feature forest. However, equivalence classes in a chart might increase exponentially because predicate–argument structures in HPSG signs represent the semantic relations of all words that the phrase dominates. For example, Figure 14 shows phrasal signs with predicate–argument structures for *saw a girl with a telescope*. In the chart in Figure 12, these signs are packed into an equivalence class. However, Figure 14 shows that the values of CONT, that is, predicate–argument structures, have different values, and the signs as they are cannot be equivalent. As seen in this example, predicate–argument structures prevent us from packing signs into equivalence classes.

In this section, we apply the feature forest model to predicate–argument structures, which may include reentrant structures and non-local dependencies. It is theoretically difficult to apply the feature forest model to predicate–argument structures; a feature forest cannot represent graph structures that include reentrant structures in a straightforward manner. However, if predicate–argument structures are constructed as in the manner described subsequently, they can be represented by feature forests of a tractable size.

Feature forests can represent predicate–argument structures if we assume some locality and monotonicity in the composition of predicate–argument structures.

Locality: In each step of composition of a predicate–argument structure, only a limited depth of the daughters' predicate–argument structures are referred to. That is, local structures in the deep descendent phrases may be ignored to construct larger phrases. This assumption means that predicate–argument structures can be packed into conjunctive nodes by ignoring local structures.

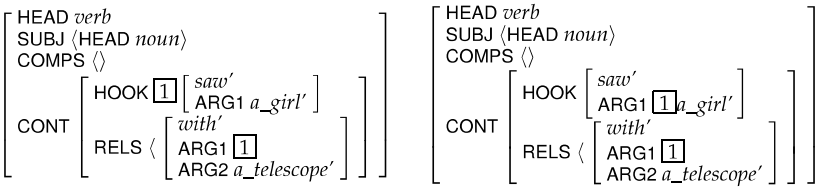


Figure 14
Signs with predicate–argument structures.

Monotonicity: All relations in the daughters' predicate–argument structures are percolated to the mother. That is, none of the predicate–argument relations in the daughter phrases disappear in the mother. Thus predicate–argument structures of descendent phrases can be located at lower nodes in a feature forest.

Predicate–argument structures usually satisfy the above conditions, even when they include non-local dependencies. For example, Figure 15 shows HPSG lexical entries for the *wh*-extraction of the object of *love* (left) and for the control construction of *try* (right). The first condition is satisfied because both lexical entries refer to CONT|HOOK of argument signs in SUBJ, COMPS, and SLASH. None of the lexical entries directly access ARG*X* of the arguments. The second condition is also satisfied because the values of CONT|HOOK of all of the argument signs are percolated to ARG*X* of the mother. In addition, the elements in CONT|RELS are percolated to the mother by the Semantic Principle. Compositional semantics usually satisfies the above conditions, including MRS (Copestake et al. 1995, 2006). The composition of MRS refers to HOOK, and no internal structures of daughters. The Semantic Principle of MRS also assures that all semantic relations in RELS are percolated to the mother. When these conditions are satisfied, semantics may include any constraints, such as selectional restrictions, although the grammar we used in the experiments does not include semantic restrictions to constrain parse forests.

Under these conditions, local structures of predicate–argument structures are encoded into a conjunctive node when the values of all of its arguments have been instantiated. We introduce the notion of inactives to denote such local structures.

Definition 10 (Inactives)
An **inactive** is a subset of predicate–argument structures in which all arguments have been instantiated.

Because inactive parts will not change during the rest of the parsing process, they can be placed in a conjunctive node. By placing newly generated inactives into corresponding conjunctive nodes, a set of predicate–argument structures can be represented in a feature forest by packing local ambiguities, and non-local dependencies are preserved.

Figure 16 illustrates a process of parsing the sentence *She ignored the fact that I wanted to dispute*, where *dispute* has an ambiguity (*dispute1*, intransitive, and *dispute2*, transitive)

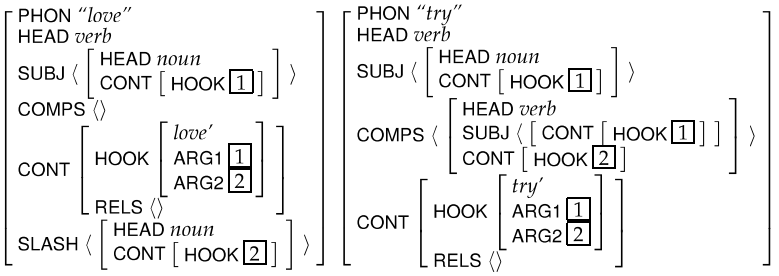


Figure 15
Lexical entries including non-local relations.

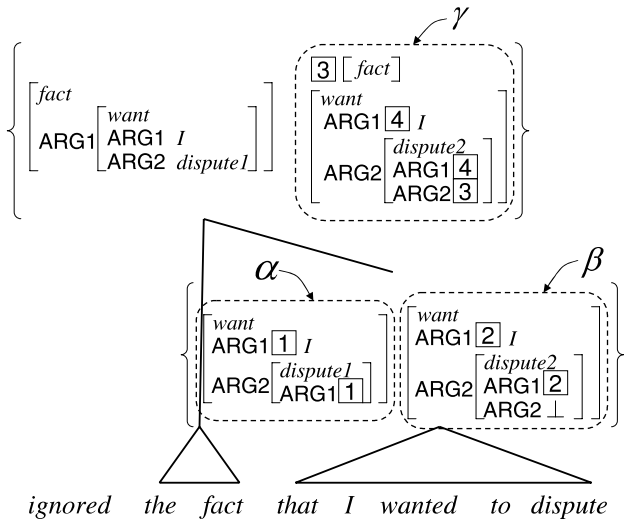


Figure 16
Process of composing predicate–argument structures.

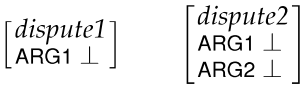


Figure 17
Predicate–argument structures of *dispute*.

and *fact* may optionally take a complementizer phrase.[7] The predicate–argument structures for *dispute1* and *dispute2* are shown in Figure 17. Curly braces express the ambiguities of partially constructed predicate–argument structures. The resulting feature forest is shown in Figure 18. The boxes denote conjunctive nodes and d_x represent disjunctive nodes.

The clause *I wanted to dispute* has two possible predicate–argument structures: one corresponding to *dispute1* (α in Figure 16) and the other corresponding to *dispute2* (β in Figure 16). The nodes of the predicate–argument structure α are all instantiated, that is, it contains only inactives. The corresponding conjunctive node (α' in Figure 18) has two inactives, for *want* and *dispute1*. The other structure β has an unfilled object in the argument (ARG2[8]) of *dispute2*, which will be filled by the non-local dependency. Hence, the corresponding conjunctive node β' has only one inactive corresponding to *want*, and the remaining part that corresponds to *dispute2* is passed on for further processing. When we process the phrase *the fact that I wanted to dispute*, the object of *dispute2* is filled by *fact* (γ in Figure 16), and the predicate–argument structure of *dispute2* is then placed into a conjunctive node (γ' in Figure 18).

7 In Figure 16, feature structures of different nodes of parse trees are assigned distinct variables, even when they are from the same lexical entries. This is because feature structures are copied during chart parsing. Although these variables are from the same lexical entry, it is copied to several chart items, and hence there are no structure sharings among them.

8 ⊥ (bottom) represents an uninstantiated value (Carpenter 1992).

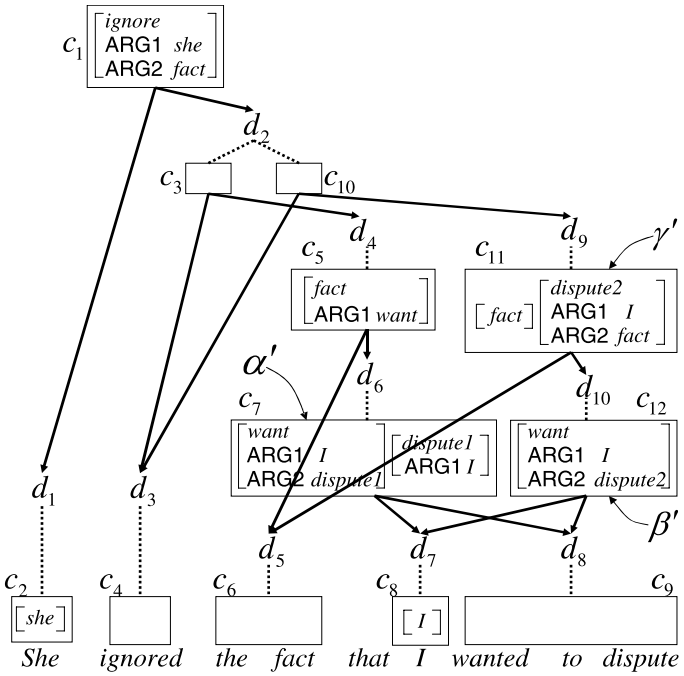


Figure 18
A feature forest representation of predicate–argument structures.

One of the beneficial characteristics of this packed representation is that the representation is isomorphic to the parsing process, that is, a chart. Hence, we can assign features of HPSG parse trees to a conjunctive node, together with features of predicate–argument structures. In Section 5, we will investigate the contribution of features on parse trees and predicate–argument structures to the disambiguation of HPSG parsing.

4.4 Filtering by Preliminary Distribution

The method just described is the essence of our solution for the tractable estimation of maximum entropy models on exponentially many HPSG parse trees. However, the problem of computational cost remains. Construction of feature forests requires parsing of all of the sentences in a treebank. Despite the development of methods to improve HPSG parsing efficiency (Oepen, Flickinger, et al. 2002), exhaustive parsing of all sentences is still expensive.

We assume that computation of parse trees with low probabilities can be omitted in the estimation stage because $T(\mathbf{w})$ can be approximated by parse trees with high probabilities. To achieve this, we first prepared *a preliminary probabilistic model* whose estimation did not require the parsing of a treebank. The preliminary model was used to reduce the search space for parsing a training treebank.

The preliminary model in this study is a unigram model, $\bar{p}(t|\mathbf{w}) = \prod_{w \in \mathbf{w}} p(l|w)$, where $w \in \mathbf{w}$ is a word in the sentence $\mathbf{w}$, and l is a lexical entry assigned to w. This model is estimated by counting the relative frequencies of lexical entries used for w in the training data. Hence, the estimation does not require parsing of a treebank. Actually, we use a maximum entropy model to compute this probability as described in Section 5.

The preliminary model is used for filtering lexical entries when we parse a treebank. Given this model, we restrict the number of lexical entries used to parse a treebank. With a threshold n for the number of lexical entries and a threshold ϵ for the probability, lexical entries are assigned to a word in descending order of probability, until the number of assigned entries exceeds n, or the accumulated probability exceeds ϵ. If this procedure does not assign a lexical entry necessary to produce a correct parse (i.e., an oracle lexical entry), it is added to the list of lexical entries. It should be noted that oracle lexical entries are given by the HPSG treebank. This assures that the filtering method does not exclude correct parse trees from parse forests.

Figure 19 shows an example of filtering the lexical entries assigned to *saw*. With $\epsilon = 0.95$, four lexical entries are assigned. Although the lexicon includes other lexical entries, such as a verbal entry taking a sentential complement (p = 0.01 in the figure), they are filtered out. Although this method reduces the time required for parsing a treebank, this approximation causes bias in the training data and results in lower accuracy. The trade-off between parsing cost and accuracy will be examined experimentally in Section 5.4.

We have several ways to integrate $\bar{p}$ with the estimated model $p(t|T(\mathbf{w}))$. In the experiments, we will empirically compare the following methods in terms of accuracy and estimation time.

Filtering only: The unigram probability $\bar{p}$ is used only for filtering in training.

Product: The probability is defined as the product of $\bar{p}$ and the estimated model p.

Reference distribution: $\bar{p}$ is used as a reference distribution of p.

Feature function: $\log \bar{p}$ is used as a feature function of p. This method has been shown to be a generalization of the reference distribution method (Johnson and Riezler 2000).

4.5 Features

Feature functions in maximum entropy models are designed to capture the characteristics of $\langle e_m, e_l, e_r \rangle$. In this article, we investigate combinations of the atomic features listed

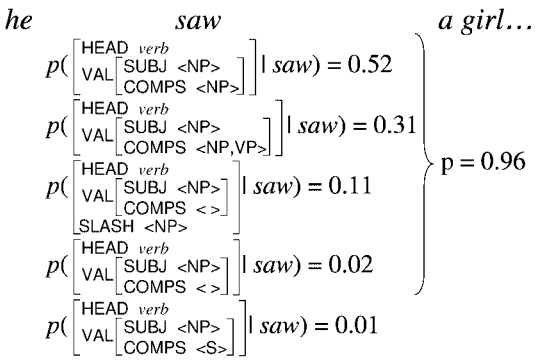


Figure 19
Filtering of lexical entries for *saw*.

Table 1
Templates for atomic features.

RULE	name of the applied schema
DIST	distance between the head words of the daughters
COMMA	whether a comma exists between daughters and/or inside of daughter phrases
SPAN	number of words dominated by the phrase
SYM	symbol of the phrasal category (e.g., NP, VP)
WORD	surface form of the head word
POS	part-of-speech of the head word
LE	lexical entry assigned to the head word
ARG	argument label of a predicate

in Table 1. The following combinations are used for representing the characteristics of binary/unary schema applications.

$$f_{\text{binary}} = \left\langle \begin{array}{l} \text{RULE,DIST,COMMA,} \\ \text{SPAN}_l, \text{SYM}_l, \text{WORD}_l, \text{POS}_l, \text{LE}_l, \\ \text{SPAN}_r, \text{SYM}_r, \text{WORD}_r, \text{POS}_r, \text{LE}_r \end{array} \right\rangle$$

$$f_{\text{unary}} = \langle \text{RULE,SYM,WORD,POS,LE} \rangle$$

where subscripts l and r denote left and right daughters.

In addition, the following is used for expressing the condition of the root node of the parse tree.

$$f_{\text{root}} = \langle \text{SYM,WORD,POS,LE} \rangle$$

Feature functions to capture predicate–argument dependencies are represented as follows:

$$f_{\text{pa}} = \left\langle \text{ARG}, \text{DIST}, \text{WORD}_p, \text{POS}_p, \text{LE}_p, \text{WORD}_a, \text{POS}_a, \text{LE}_a \right\rangle$$

where subscripts p and a represent predicate and argument, respectively.

Figure 20 shows examples: f_{root} is for the root node, in which the phrase symbol is S and the surface form, part-of-speech, and lexical entry of the lexical head are *saw*, VBD, and a transitive verb, respectively. f_{binary} is for the binary rule application to *saw a girl* and *with a telescope*, in which the applied schema is the Head-Modifier Schema, the left daughter is VP headed by *saw*, and the right daughter is PP headed by *with*, whose part-of-speech is IN and whose lexical entry is a VP-modifying preposition.

Figure 21 shows example features for predicate–argument structures. The figure shows features assigned to the conjunctive node denoted as α' in Figure 18. Because inactive structures in the node have three predicate–argument relations, three features are activated. The first one is for the relation of *want* and *I*, where the label of the relation is ARG1, the distance between the head words is 1, the surface string and the POS of

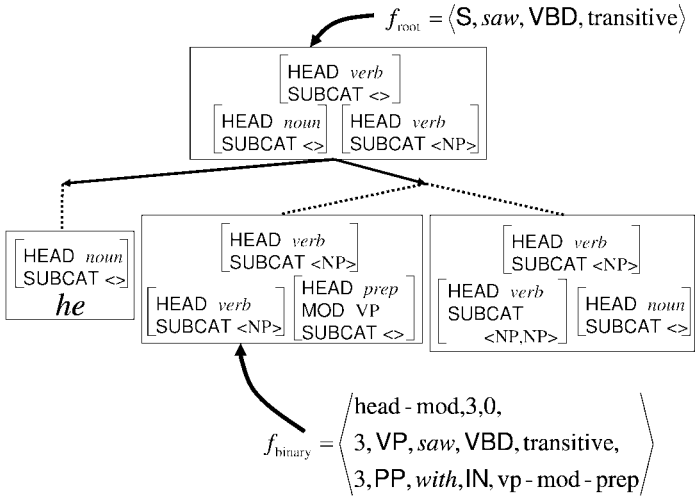


Figure 20
Example features for binary schema application and root condition.

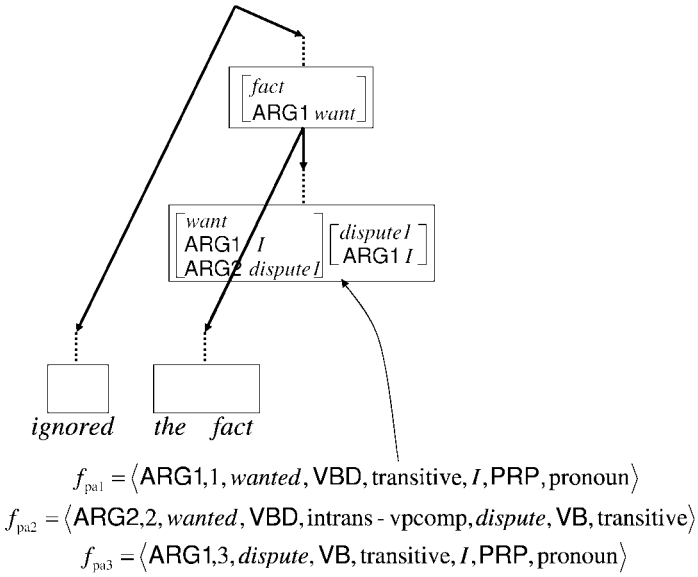


Figure 21
Example features for predicate–argument structures.

the predicate are *want* and VBD, and those of the argument are *I* and PRP. The second and the third features are for the other two relations. We may include features on more than two relations, such as the dependencies among *want*, *I*, and *dispute*, although such features are not incorporated currently.

In our implementation, some of the atomic features are abstracted (i.e., ignored) for smoothing. Tables 2, 3, and 4 show the full set of templates of combined features used in the experiments. Each row represents the template for a feature function. A check indicates the atomic feature is incorporated, and a hyphen indicates the feature is ignored.

Table 2
Feature templates for binary schema (left) and unary schema (right).

RULE	DIST	COMMA	SPAN	SYM	WORD	POS	LE
√	√	√	–	–	√	√	√
√	√	√	–	–	√	√	–
√	√	√	–	–	√	–	√
√	√	√	–	√	√	–	–
√	–	√	√	–	√	√	√
√	–	√	√	–	√	√	–
√	–	√	√	–	√	–	√
√	–	√	√	√	√	–	–
√	√	√	–	–	–	√	√
√	√	√	–	–	–	√	–
√	√	√	–	–	–	–	√
√	√	√	–	√	–	–	–
√	–	√	√	–	–	√	√
√	–	√	√	–	–	√	–
√	–	√	√	–	–	–	√
√	–	√	√	√	–	–	–

RULE	SYM	WORD	POS	LE
√	–	√	√	√
√	–	√	√	–
√	–	√	–	√
√	√	√	–	–
√	–	–	√	√
√	–	–	√	–
√	–	–	–	√
√	√	–	–	–

Table 3
Feature templates for root condition.

SYM	WORD	POS	LE
–	√	√	√
–	√	√	–
–	√	–	√
√	√	–	–
–	–	√	√
–	–	√	–
–	–	–	√
√	–	–	–

Table 4
Feature templates for predicate–argument dependencies.

ARG	DIST	WORD	POS	LE
√	√	√	√	√
√	√	√	–	√
√	√	–	√	√
√	–	√	√	√
√	–	√	–	√
√	–	–	√	√
√	√	√	√	–
√	√	√	–	–
√	√	–	√	–
√	–	√	√	–
√	–	√	–	–
√	–	–	√	–

5. Experiments

This section presents experimental results on the parsing accuracy attained by the feature forest models. In all of the following experiments, we use the HPSG grammar developed by the method of Miyao, Ninomiya, and Tsujii (2005). Section 5.1 describes how this grammar was developed. Section 5.2 explains other aspects of the experimental settings. In Sections 5.3 to 5.7, we report results of the experiments on HPSG parsing.

5.1 The HPSG Grammar

In the following experiments, we use Enju 2.1 (Tsujii Laboratory 2004), which is a wide-coverage HPSG grammar extracted from the Penn Treebank by the method of Miyao, Ninomiya, and Tsujii (2005). In this method, we convert the Penn Treebank into an HPSG treebank, and collect HPSG lexical entries from terminal nodes of the HPSG treebank. Figure 22 illustrates the process of treebank conversion and lexicon collection. We first convert and fertilize parse trees of the Penn Treebank. This step identifies syntactic constructions that require special treatment in HPSG, such as raising/control and long-distance dependencies. These constructions are then annotated with typed feature structures so that they conform to the HPSG analysis. Next, we apply HPSG schemas and principles, and obtain fully specified HPSG parse trees. This step solves feature structure constraints given in the previous step, and fills unspecified constraints. Failures of schema/principle applications indicate that the annotated constraints do not

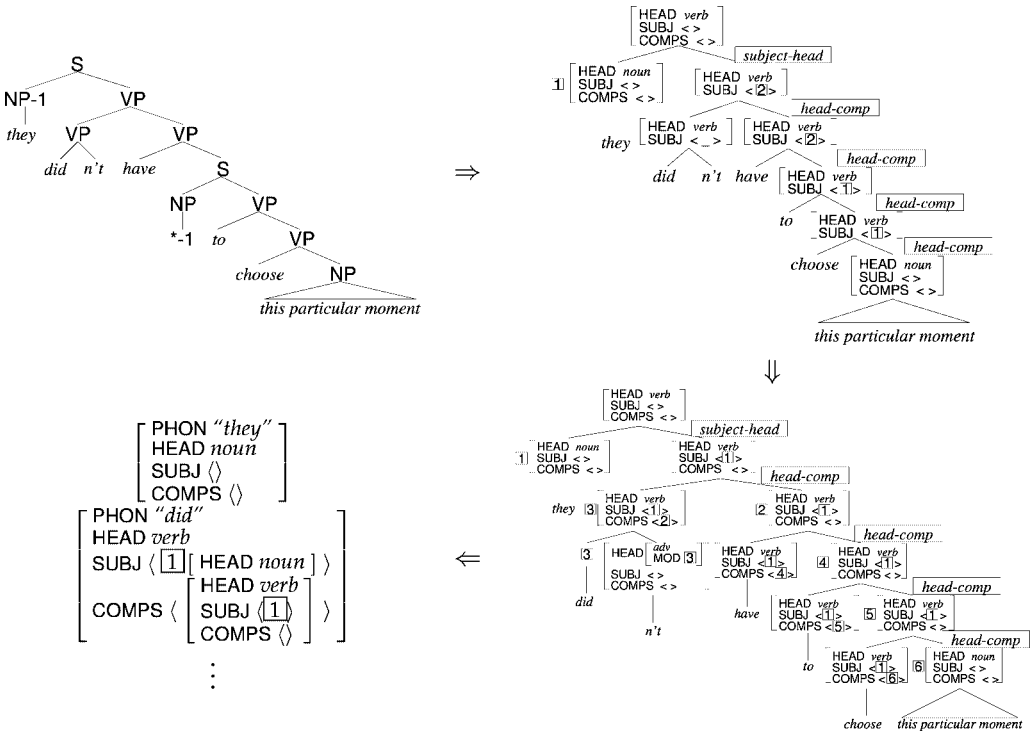


Figure 22
Extracting HPSG lexical entries from the Penn Treebank.

conform to the HPSG analysis, and require revisions. Finally, we obtain lexical entries from the HPSG parse trees. The terminal nodes of HPSG parse trees are collected, and they are generalized by removing word-specific or context-specific constraints.

An advantage of this method is that a wide-coverage HPSG lexicon is obtained because lexical entries are extracted from real-world sentences. Obtained lexical entries are guaranteed to construct well-formed HPSG parse trees because HPSG schemas and principles are successfully applied during the development of the HPSG treebank. Another notable feature is that we can additionally obtain an HPSG treebank, which can be used as training data for disambiguation models. In the following experiments, this HPSG treebank is used for the training of maximum entropy models.

The lexicon used in the following experiments was extracted from Sections 02–21 of the *Wall Street Journal* portion of the Penn Treebank. This lexicon can assign correct lexical entries to 99.09% of words in the HPSG treebank converted from Penn Treebank Section 23. This number expresses "lexical coverage" in the strong sense defined by Hockenmaier and Steedman (2002). In this notion of "coverage," this lexicon has 84.1% sentential coverage, where this means that the lexicon can assign correct lexical entries to all of the words in a sentence. Although the parser might produce parse results for uncovered sentences, these parse results cannot be completely correct.

5.2 Experimental Settings

The data for the training of the disambiguation models was the HPSG treebank derived from Sections 02–21 of the *Wall Street Journal* portion of the Penn Treebank, that is, the same set used for lexicon extraction. For training of the disambiguation models, we eliminated sentences of 40 words or more and sentences for which the parser could not produce the correct parses. The resulting training set consists of 33,604 sentences (when $n = 10$ and $\epsilon = 0.95$; see Section 5.4 for details). The treebanks derived from Sections 22 and 23 were used as the development and final test sets, respectively. Following previous studies on parsing with PCFG-based models (Collins 1997; Charniak 2000), accuracy is measured for sentences of less than 40 words and for those with less than 100 words. Table 5 shows the specifications of the test data.

The measure for evaluating parsing accuracy is precision/recall of predicate–argument dependencies output by the parser. A predicate–argument dependency is defined as a tuple $\langle w_h, w_n, \pi, \rho \rangle$, where w_h is the head word of the predicate, w_n is the head word of the argument, π is the type of the predicate (e.g., adjective, intransitive verb), and ρ is an argument label (MODARG, ARG1, ..., ARG4). For example, *He tried running* has three dependencies as follows:

- $\langle$*tried*, *he*, transitive verb, ARG1$\rangle$

Table 5
Specification of test data for the evaluation of parsing accuracy.

	No. of Sentences	Avg. Length
Test set (Section 23, $<$ 40 words)	2,144	20.52
Test set (Section 23, $<$ 100 words)	2,299	22.23
Development set (Section 22, $<$ 40 words)	1,525	20.69
Development set (Section 22, $<$ 100 words)	1,641	22.43

- $\langle$*tried*, *running*, transitive verb, ARG2$\rangle$
- $\langle$*running*, *he*, intransitive verb, ARG1$\rangle$

Labeled precision/recall (LP/LR) is the ratio of tuples correctly identified by the parser, and unlabeled precision/recall (UP/UR) is the ratio of w_h and w_n correctly identified regardless of π and ρ. F-score is the harmonic mean of LP and LR. Sentence accuracy is the exact match accuracy of complete predicate–argument relations in a sentence. These measures correspond to those used in other studies measuring the accuracy of predicate–argument dependencies in CCG parsing (Clark, Hockenmaier, and Steedman 2002; Hockenmaier 2003; Clark and Curran 2004b) and LFG parsing (Burke et al. 2004), although exact figures cannot be compared directly because the definitions of dependencies are different. All predicate–argument dependencies in a sentence are the target of evaluation except quotation marks and periods. The accuracy is measured by parsing test sentences with gold-standard part-of-speech tags from the Penn Treebank unless otherwise noted.

The Gaussian prior was used for smoothing (Chen and Rosenfeld 1999a), and its hyper-parameter was tuned for each model to maximize F-score for the development set. The algorithm for parameter estimation was the limited-memory BFGS method (Nocedal 1980; Nocedal and Wright 1999). The parser was implemented in C++ with the LiLFeS library (Makino et al. 2002), and various speed-up techniques for HPSG parsing were used such as quick check and iterative beam search (Tsuruoka, Miyao, and Tsujii 2004; Ninomiya et al. 2005). Other efficient parsing techniques, including global thresholding, hybrid parsing with a chunk parser, and large constituent inhibition, were not used. The results obtained using these techniques are given in Ninomiya et al. A limit on the number of constituents was set for time-out; the parser stopped parsing when the number of constituents created during parsing exceeded 50,000. In such a case, the parser output nothing, and the recall was computed as zero.

Features occurring more than twice were included in the probabilistic models. A method of filtering lexical entries was applied to the parsing of training data (Section 4.4). Unless otherwise noted, parameters for filtering were $n = 10$ and $\epsilon = 0.95$, and a reference distribution method was applied. The unigram model, $p_0(t|s)$, for filtering is a maximum entropy model with two feature templates, $\langle$WORD, POS, LE$\rangle$ and $\langle$POS, LE$\rangle$. The model includes 24,847 features.

5.3 Efficacy of Feature Forest Models

Tables 6 and 7 show parsing accuracy for the test set. In the tables, "Syntactic features" denotes a model with syntactic features, that is, f_{binary}, f_{unary}, and f_{root} introduced

Table 6
Accuracy of predicate–argument relations (test set, <40 words).

	LP	LR	UP	UR	F-score	Sentence acc.
Baseline	78.10	77.39	82.83	82.08	77.74	18.3
Syntactic features	86.92	86.28	90.53	89.87	86.60	36.3
Semantic features	84.29	83.74	88.32	87.75	84.01	30.9
All	86.54	86.02	90.32	89.78	86.28	36.0

Table 7
Accuracy of predicate–argument relations (test set, <100 words).

	LP	LR	UP	UR	F-score	Sentence acc.
Baseline	77.58	76.84	82.22	81.43	77.21	17.1
Syntactic features	86.47	85.83	90.06	89.40	86.15	34.1
Semantic features	83.81	83.26	87.75	87.16	83.53	28.9
All	86.13	85.59	89.85	89.29	85.86	33.8

in Section 4.5. "Semantic features" represents a model with features on predicate–argument structures, that is, f_{pa} given in Table 4. "All" is a model with both syntactic and semantic features. The "Baseline" row shows the results for the reference model, $p_0(t|s)$, used for lexical entry filtering in the estimation of the other models. This model is considered as a simple application of a traditional PCFG-style model; that is, $p(r) = 1$ for any rule r in the construction rules of the HPSG grammar.

The results demonstrate that feature forest models have significantly higher accuracy than a baseline model. Comparing "Syntactic features" with "Semantic features," we see that the former model attained significantly higher accuracy than the latter. This indicates that syntactic features are more important for overall accuracy. We will examine the contributions of each atomic feature of the syntactic features in Section 5.5.

Features on predicate–argument relations were generally considered as important for the accurate disambiguation of syntactic structures. For example, PP-attachment ambiguity cannot be resolved with only syntactic preferences. However, the results show that a model with only semantic features performs significantly worse than one with syntactic features. Even when combined with syntactic features, semantic features do not improve accuracy. Obviously, semantic preferences are necessary for accurate parsing, but the features used in this work were not sufficient to capture semantic preferences. A possible reason is that, as reported in Gildea (2001), bilexical dependencies may be too sparse to capture semantic preferences.

For reference, our results are competitive with the best corresponding results reported in CCG parsing (LP/LR = 86.6/86.3) (Clark and Curran 2004b), although our results cannot be compared directly with other grammar formalisms because each formalism represents predicate–argument dependencies differently. In contrast with the results of CCG and PCFG (Collins 1997, 1999, 2003; Charniak 2000), the recall is clearly lower than precision. This may have resulted from the HPSG grammar having stricter feature constraints and the parser not being able to produce parse results for around 1% of the sentences. To improve recall, we need techniques to deal with these 1% of sentences.

Table 8 gives the computation/space costs of model estimation. "Estimation time" indicates user times required for running the parameter estimation algorithm. "No. of feature occurrences" denotes the total number of occurrences of features in the training data, and "Data size" gives the sizes of the compressed files of training data. We can conclude that feature forest models are estimated at a tractable computational cost and a reasonable data size, even when a model includes semantic features including non-local dependencies. The results reveal that feature forest models essentially solve the problem of the estimation of probabilistic models of sentence structures.

Table 8
Computation/space costs of model estimation.

	No. of features	Estimation time (sec.)	No. of feature occurrences	Data size (MB)
Baseline	24,847	499	6,948,364	21
Syntactic features	599,104	511	127,497,615	727
Semantic features	334,821	278	176,534,753	375
All	933,925	716	304,032,368	1,093

Table 9
Estimation method vs. accuracy and estimation time.

	LP	LR	F-score	Estimation time (sec.)
Filtering only	51.70	49.89	50.78	449
Product	86.50	85.94	86.22	1,568
Reference distribution	86.92	86.28	86.60	511
Feature function	84.81	84.09	84.45	945

5.4 Comparison of Filtering Methods

Table 9 compares the estimation methods introduced in Section 4.4. In all of the following experiments, we show the accuracy for the test set (<40 words) only. Table 9 reveals that our method achieves significantly lower accuracy when it is used only for filtering in the training phrase. One reason is that the feature forest model prefers lexical entries that are filtered out in the training phase, because they are always oracle lexical entries in the training. This means that we must incorporate the preference of filtering into the final parse selection. As shown in Table 9, the models combined with a preliminary model achieved sufficient accuracy. The reference distribution method achieved higher accuracy and lower cost. The feature function method achieved lower accuracy in our experiments. A possible reason for this is that a hyper-parameter of the prior was set to the same value for all the features including the feature of the log-probability given by the preliminary distribution.

Tables 10 and 11 show the results of changing the filtering threshold. We can determine the correlation between the estimation/parsing cost and accuracy. In our experiment, $n \geq 10$ and $\epsilon \geq 0.90$ seem necessary to preserve the F-score over 86.0.

5.5 Contribution of Features

Table 12 shows the accuracy with different feature sets. Accuracy was measured for 15 models with some atomic features removed from the final model. The last row denotes the accuracy attained by the unigram model (i.e., the reference distribution). The numbers in bold type represent a significant difference from the final model according to stratified shuffling tests with the Bonferroni correction (Cohen 1995) with p-value $< .05$ for 32 pairwise comparisons. The results indicate that DIST, COMMA, SPAN, WORD, and

Table 10
Filtering threshold vs. accuracy.

n, ϵ	LP	LR	F-score	Sentence acc.
5, 0.80	85.09	84.30	84.69	32.4
5, 0.90	85.44	84.61	85.02	32.5
5, 0.95	85.52	84.66	85.09	32.7
5, 0.98	85.50	84.63	85.06	32.6
10, 0.80	85.60	84.65	85.12	32.5
10, 0.90	86.49	85.92	86.20	34.7
10, 0.95	86.92	86.28	86.60	36.3
10, 0.98	87.18	86.66	86.92	37.7
15, 0.80	85.59	84.63	85.11	32.4
15, 0.90	86.48	85.80	86.14	35.7
15, 0.95	87.21	86.68	86.94	37.0
15, 0.98	87.69	87.16	87.42	39.2

Table 11
Filtering threshold vs. estimation cost.

n, ϵ	Estimation time (sec.)	Parsing time (sec.)	Data size (MB)
5, 0.80	108	5,103	341
5, 0.90	150	6,242	407
5, 0.95	190	7,724	469
5, 0.98	259	9,604	549
10, 0.80	130	6,003	370
10, 0.90	268	8,855	511
10, 0.95	511	15,393	727
10, 0.98	1,395	36,009	1,230
15, 0.80	123	6,298	372
15, 0.90	259	9,543	526
15, 0.95	735	20,508	854
15, 0.98	3,777	86,844	2,031

POS features contributed to the final accuracy, although the differences were slight. In contrast, RULE, SYM, and LE features did not affect accuracy. However, when each was removed together with another feature, the accuracy decreased drastically. This implies that such features carry overlapping information.

5.6 Factors for Parsing Accuracy

Table 13 shows parsing accuracy for covered and uncovered sentences. As defined in Section 5.1, "covered" indicates that the HPSG lexicon has all correct lexical entries for a sentence. In other words, for covered sentences, exactly correct parse trees are obtained if the disambiguation model worked perfectly. The result reveals clear differences in accuracy between covered and uncovered sentences. The F-score for covered sentences is around 2.5 points higher than the overall F-score, whereas the F-score is more than 10 points lower for uncovered sentences. This result indicates improvement of lexicon quality is an important factor for higher accuracy.

Table 12
Accuracy with different feature sets.

Features	LP	LR	F-score	Sentence acc.	No. of features
All	86.92	86.28	86.60	36.3	599,104
−RULE	86.83	86.19	86.51	36.3	596,446
−DIST	**86.52**	85.96	86.24	35.7	579,666
−COMMA	**86.31**	**85.81**	86.06	34.4	584,040
−SPAN	**86.32**	**85.75**	86.03	35.5	559,490
−SYM	86.74	86.16	86.45	35.4	406,545
−WORD	**86.39**	**85.77**	86.08	35.3	91,004
−POS	**86.18**	**85.61**	85.89	34.1	406,545
−LE	86.91	86.32	86.61	36.8	387,938
−DIST,SPAN	**85.39**	**84.82**	85.10	33.1	270,467
−DIST,SPAN,COMMA	**83.75**	**83.25**	83.50	28.9	261,968
−RULE,DIST,SPAN,COMMA	**83.44**	**82.93**	83.18	27.6	259,372
−WORD,LE	**86.40**	85.81	86.10	34.7	25,429
−WORD,POS	**85.44**	**84.87**	85.15	32.7	40,102
−WORD,POS,LE	**84.68**	**84.12**	84.40	31.1	8,899
−SYM,WORD,POS,LE	**82.77**	**82.14**	82.45	24.9	1,914
None	**78.10**	**77.39**	77.74	18.3	0

Table 13
Accuracy for covered/uncovered sentences.

	LP	LR	F-score	Sentence acc.	No. of sentences
covered sentences	89.36	88.96	89.16	42.2	1,825
uncovered sentences	75.57	74.04	74.80	2.5	319

Figure 23 shows the learning curve. A feature set was fixed, and the parameter of the Gaussian prior was optimized for each model. High accuracy is attained even with a small training set, and the accuracy seems to be saturated. This indicates that we cannot further improve the accuracy simply by increasing the size of the training data set. The exploration of new types of features is necessary for higher accuracy. It should also be noted that the upper bound of the accuracy is not 100%, because the grammar cannot produce completely correct parse results for uncovered sentences.

Figure 24 shows the accuracy for each sentence length. It is apparent from this figure that the accuracy is significantly higher for sentences with less than 10 words. This implies that experiments with only short sentences overestimate the performance of parsers. Sentences with at least 10 words are necessary to properly evaluate the performance of parsing real-world texts. The accuracies for the sentences with more than 10 words are not very different, although data points for sentences with more than 50 words are not reliable.

Table 14 shows the accuracies for predicate–argument relations when parts-of-speech tags are assigned automatically by a maximum-entropy-based parts-of-speech tagger (Tsuruoka and Tsujii 2005). The results indicate a drop of about three points in labeled precision/recall (a two-point drop in unlabeled precision/recall). A reason why we observed larger accuracy drops in labeled precision/recall is that

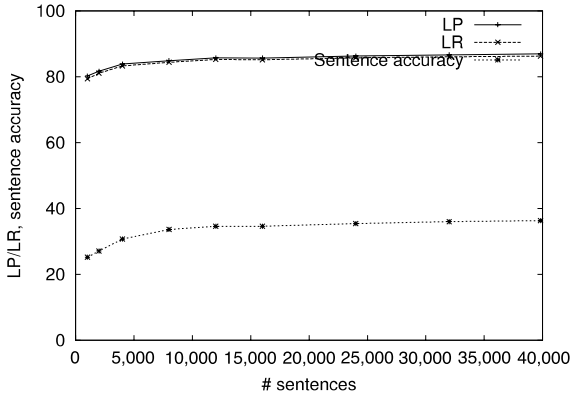


Figure 23
Corpus size vs. accuracy.

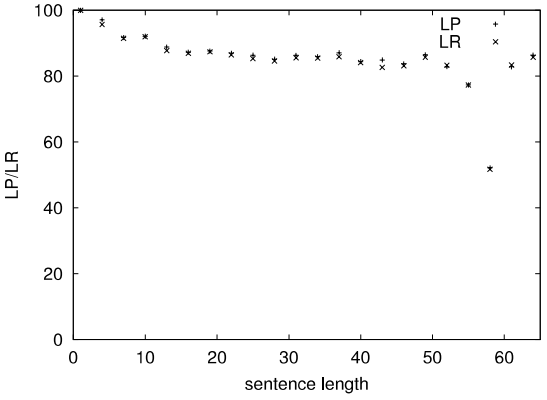


Figure 24
Sentence length vs. accuracy.

predicate–argument relations are fragile with respect to parts-of-speech errors because predicate types (e.g., adjective, intransitive verb) are determined depending on the parts-of-speech of predicate words. Although our current parsing strategy assumes that parts-of-speech are given beforehand, for higher accuracy in real application contexts, we will need a method for determining parts-of-speech and parse trees jointly.

Table 14
Accuracy with automatic parts-of-speech tags (test set).

	LP	LR	UP	UR	F-score	Sentence acc.
<40 words	83.88	82.84	88.83	87.73	83.36	30.1
<100 words	83.45	82.40	88.37	87.26	82.92	28.2

5.7 Analysis of Disambiguation Errors

Table 15 shows a manual classification of the causes of disambiguation errors in 100 sentences randomly chosen from Section 00. In our evaluation, one error source may cause multiple dependency errors. For example, if an incorrect lexical entry is assigned to a verb, all of the argument dependencies of the verb are counted as errors. The numbers in the table include such double-counting. Figure 25 shows examples of disambiguation errors. The figure shows output from the parser.

Major causes are classified into three types: attachment ambiguity, argument/modifier distinction, and lexical ambiguity. As attachment ambiguities are well-known error sources, PP-attachment is the largest source of errors in our evaluation. Our disambiguation model cannot accurately resolve PP-attachment ambiguities because it does not include dependencies among a modifiee and the argument of the preposition. Because previous studies revealed that such dependencies are effective features for PP-attachment resolution, we should incorporate them into our model. Some of the attachment ambiguities, including adjective and adverb, should also be resolved with an extension of features. However, we cannot identify any effective features for the disambiguation of attachment of verbal phrases, including relative clauses, verb phrases, subordinate clauses, and *to*-infinitives. For example, Figure 25 shows an example error of the attachment of a relative clause. The correct answer is that the

Table 15
Classification of disambiguation errors.

Error cause		No. of errors
Attachment ambiguity	prepositional phrase	32
	relative clause	14
	adjective	7
	adverb	6
	verb phrase	5
	subordinate clause	3
	to-infinitive	3
	others	6
Argument/modifier distinction	*to*-infinitive	19
	noun phrase	7
	verb phrase	7
	subordinate clause	7
	others	9
Lexical ambiguity	preposition/modifier	13
	verb subcategorization frame	13
	participle/adjective	12
	others	6
Test set errors	errors of treebank conversion	18
	errors of Penn Treebank	4
Comma		32
Noun phrase identification		15
Coordination/insertion		15
Zero-pronoun resolution		9
Others		4

Attachment of a subordinate clause It's made only in years when the grapes ripen perfectly (the last was 1979) and comes from a single acre of [NP grapes [S' that yielded a mere 75 cases in 1987]].
Argument/modifier distinction of a to-infinitive More than a few CEOs say the red-carpet treatment tempts them [S-modifier to return to a heartland city for future meetings].
Prepositional phrase or modifier Mitsui Mining & Smelting Co. posted a 62 % rise in pretax profit to 5.276 billion yen ($ 36.9 million) in its fiscal first half ended Sept. 30 [VP compared with 3.253 billion yen a year earlier].
Wrong assignment of a subcategorization frame [NP-subject "Nasty innuendoes,"] [VP says [NP-object John Siegal, Mr. Dinkins's issues director, "designed to prosecute a case of political corruption that simply doesn't exist."]]

Figure 25
Examples of disambiguation errors.

subject of *yielded* is *acre*, but this cannot be determined only by the relation among *yield*, *grapes*, and *acre*. The resolution of these errors requires a novel type of feature function.

Errors of argument/modifier distinction are prominent in deep syntactic analysis, because arguments and modifiers are not explicitly distinguished in the evaluation of CFG parsers. Figure 25 shows an example of the argument/modifier distinction of a *to*-infinitive clause. In this case, the *to*-infinitive clause is a complement of *tempts*. The subcategorization frame of *tempts* seems responsible for this problem. However, the disambiguation model wrongly assigned a lexical entry for a transitive verb because of the sparseness of the training data (*tempts* occurred only once in the training data). The resolution of this sort of ambiguity requires the refinement of a probabilistic model of lexical entries. Errors of verb phrases and subordinate clauses are similar to this example. Errors of argument/modifier distinction of noun phrases are mainly caused by temporal nouns and cardinal numbers. The resolution of these errors seems to require the identification of temporal expressions and usage of cardinal numbers.

Errors of lexical ambiguities were mainly caused by idioms. For example, in Figure 25, *compared with* is a compound preposition, but the parser recognized it as a verb phrase. This indicates that the grammar or the disambiguation model requires the special treatment of idioms. Errors of verb subcategorization frames were mainly caused by difficult constructions such as insertions. Figure 25 shows that the parser could not identify the inserted clause (*says John Siegel...*) and a lexical entry for a declarative transitive verb was chosen.

Attachment errors of commas are also significant. It should be noted that commas were ignored in the evaluation of CFG parsers. We did not eliminate punctuation from the evaluation because punctuation sometimes contributes to semantics, as in coordination and insertion. In this error analysis, errors of commas representing coordination/insertion are classified into "coordination/insertion," and "comma" indicates errors that do not contribute to the computation of semantics.

Errors of noun phrase identification mean that a noun phrase was split into two phrases. These errors were mainly caused by the indirect effects of other errors.

Errors of identifying coordination/insertion structures sometimes resulted in catastrophic analyses. While accurate analysis of such constructions is indispensable, it is also known to be difficult because disambiguation of coordination/insertion requires the computation of preferences over global structures, such as the similarity of syntactic/semantic structure of coordinates. Incorporating features for representing the similarity of global structures is difficult for feature forest models.

Zero-pronoun resolution is also a difficult problem. However, we found that most were indirectly caused by errors of argument/modifier distinction in *to*-infinitive clauses.

A significant portion of the errors discussed above cannot be resolved by the features we investigated in this study, and the design of other features will be necessary for improving parsing accuracy.

6. Discussion

6.1 Probabilistic Modeling of Complete Structures

The model described in this article was first published in Miyao and Tsujii (2002), and has been applied to probabilistic models for parsing with lexicalized grammars. Applications to CCG parsing (Clark and Curran 2003, 2004b) and LFG parsing (Kaplan et al. 2004; Riezler and Vasserman 2004) demonstrated that feature forest models attained higher accuracy than other models. These researchers applied feature forests to representations of the packed parse results of LFG and the dependency/derivation structures of CCG. Their work demonstrated the applicability and effectiveness of feature forest models in parsing with wide-coverage lexicalized grammars. Feature forest models were also shown to be effective for wide-coverage sentence realization (Nakanishi, Miyao, and Tsujii 2005). This work demonstrated that feature forest models are generic enough to be applied to natural language processing tasks other than parsing.

The work of Geman and Johnson (2002) independently developed a dynamic programming algorithm for maximum entropy models. The solution was similar to our approach, although their method was designed to traverse LFG parse results represented with disjunctive feature structures as proposed by Maxwell and Kaplan (1995). The difference between the two approaches is that feature forests use a simpler generic data structure to represent packed forest structures. Therefore, without assuming what feature forests represent, our algorithm can be applied to various tasks, including theirs.

Another approach to the probabilistic modeling of complete structures is a method of approximation. The work on whole sentence maximum entropy models (Rosenfeld 1997; Chen and Rosenfeld 1999b) proposed an approximation algorithm to estimate parameters of maximum entropy models on whole sentence structures. However, the algorithm suffered from slow convergence, and the model was basically a sequence model. It could not produce a solution for complex structures as our model can.

We should also mention Conditional Random Fields (CRFs) (Lafferty, McCallum, and Pereira 2001) for solving a similar problem in the context of maximum entropy Markov models. Their solution was an algorithm similar to the computation of forward/backward probabilities of hidden Markov models (HMMs). Their algorithm is a special case of our algorithm in which each conjunctive node has only one daughter. This is obvious because feature forests can represent Markov chains. In an analogy, CRFs correspond to HMMs, whereas feature forest models correspond to PCFGs.

Extensions of CRFs, such as semi-Markov CRFs (Sarawagi and Cohen 2004), are also regarded as instances of feature forest models. This fact implies that our algorithm is applicable to not only parsing but also to other tasks. CRFs are now widely used for sequence-based tasks, such as parts-of-speech tagging and named entity recognition, and have been shown to achieve the best performance in various tasks (McCallum and Li 2003; McCallum, Rohanimanesh, and Sutton 2003; Pinto et al. 2003; Sha and Pereira 2003; Peng and McCallum 2004; Roark et al. 2004; Settles 2004; Sutton, Rohanimanesh, and McCallum 2004). These results suggest that the method proposed in the present article will achieve high accuracy when applied to various statistical models with tree structures. Dynamic CRFs (McCallum, Rohanimanesh, and Sutton 2003; Sutton, Rohanimanesh, and McCallum 2004) provide us with an interesting inspiration for extending feature forest models. The purpose of dynamic CRFs is to incorporate feature functions that are not represented locally, and the solution is to apply a variational method, which is an algorithm of numerical computation, to obtain approximate solutions. A similar method may be developed to overcome a bottleneck of feature forest models, that is, the fact that feature functions are localized to conjunctive nodes.

The structure of feature forests is common in natural language processing and computational linguistics. As is easily seen, lattices, Markov chains, and CFG parse trees are represented by feature forests. Furthermore, because conjunctive nodes do not necessarily represent CFG nodes or rules and terminals of feature forests need not be words, feature forests can express any forest structures in which ambiguities are packed in local structures. Examples include the derivation trees of LTAG and CCG. Chiang (2003) proved that feature forests could be considered as the derivation forests of *linear context-free rewriting systems* (LCFRSs) (Vijay-Shanker, Weir, and Joshi 1987; Weir 1988). LCFRSs define a wide variety of grammars, including LTAG and CCG, while preserving polynomial-time complexity of parsing. This demonstrates that feature forest models are applicable to probabilistic models far beyond PCFGs. Feature forests are also isomorphic to *support graphs* (or *explanation graphs*) used in the graphical EM algorithm (Kameya and Sato 2000). In their framework, a program in a logic programming language, PRISM (Sato and Kameya 1997), is converted into support graphs, and parameters of probabilistic models are automatically learned by an EM algorithm. Support graphs have been proved to represent various statistical structural models, including HMMs, PCFGs, Bayesian networks, and many other graphical structures (Sato and Kameya 2001; Sato 2005). Taken together, these results imply the high applicability of feature forest models to various real tasks.

Because feature forests have a structure isomorphic to parse forests of PCFG, it might seem that they can represent only immediate dominance relations of CFG rules as in PCFG, resulting in only a slight, trivial extension of PCFG. As described herein, however, feature forests can represent structures beyond CFG parse trees. Furthermore, because feature forests are a generalized representation of ambiguous structures, each node in a feature forest need not correspond to a node in a PCFG parse forest. That is, a node in a feature forest may represent any linguistic entity, including a fragment of a syntactic structure, a semantic relation, or other sentence-level information.

The idea of feature forest models could be applied to non-probabilistic machine learning methods. Taskar et al. (2004) proposed a dynamic programming algorithm for the learning of large-margin classifiers including support vector machines (Vapnik 1995), and presented its application to disambiguation in CFG parsing. Their algorithm resembles feature forest models; an optimization function is computed by a dynamic programing algorithm without unpacking packed forest structures. From the discussion in this article, it is evident that if the main part of an update formula is represented

with (the exponential of) linear combinations, a method similar to feature forest models should be applicable.

6.2 Probabilistic Parsing with Lexicalized Grammars

Before the advent of feature forest models, studies on probabilistic models of HPSG adopted conventional maximum entropy models to select the most probable parse from parse candidates given by HPSG grammars (Oepen, Toutanova, et al. 2002; Toutanova and Manning 2002; Baldridge and Osborne 2003). The difference between these studies and our work is that we used feature forests to avoid the exponential increase in the number of structures that results from unpacked parse results. These studies ignored the problem of exponential explosion; in fact, training sets in these studies were very small and consisted only of short sentences. A possible approach to avoid this problem is to develop a fully restrictive grammar that never causes an exponential explosion, although the development of such a grammar requires considerable effort and it cannot be acquired from treebanks using existing approaches. We think that exponential explosion is inevitable, particularly with the large-scale wide-coverage grammars required to analyze real-world texts. In such cases, these methods of model estimation are intractable.

Another approach to estimating log-linear models for HPSG was to extract a small *informative sample* from the original set $T(\mathbf{w})$ (Osborne 2000). The method was successfully applied to Dutch HPSG parsing (Malouf and van Noord 2004). A possible problem with this method is in the approximation of exponentially many parse trees by a polynomial-size sample. However, their method has an advantage in that any features on parse results can be incorporated into a model, whereas our method forces feature functions to be defined locally on conjunctive nodes. We will discuss the trade-off between the approximation solution and the locality of feature functions in Section 6.3.

Non-probabilistic statistical classifiers have also been applied to disambiguation in HPSG parsing: voted perceptrons (Baldridge and Osborne 2003) and support vector machines (Toutanova, Markova, and Manning 2004). However, the problem of exponential explosion is also inevitable using their methods. As described in Section 6.1, an approach similar to ours may be applied, following the study of Taskar et al. (2004).

A series of studies on parsing with LFG (Johnson et al. 1999; Riezler et al. 2000, 2002) also proposed a maximum entropy model for probabilistic modeling of LFG parsing. However, similarly to the previous studies on HPSG parsing, these groups had no solution to the problem of exponential explosion of unpacked parse results. As discussed in Section 6.1, Geman and Johnson (2002) proposed an algorithm for maximum entropy estimation for packed representations of LFG parses.

Recent studies on CCG have proposed probabilistic models of dependency structures or predicate–argument dependencies, which are essentially the same as the predicate–argument structures described in the present article. Clark, Hockenmaier, and Steedman (2002) attempted the modeling of dependency structures, but the model was inconsistent because of the violation of the independence assumption. Hockenmaier (2003) proposed a consistent generative model of predicate–argument structures. The probability of a non-local dependency was conditioned on multiple words to preserve the consistency of the probability model; that is, probability $p(I|want, dispute)$ in Section 4.3 was directly estimated. The problem was that such probabilities could not be estimated directly from the data due to data sparseness, and a heuristic method had to be employed. Probabilities were therefore estimated as the average of individual probabilities conditioned on a single word. Another problem is that the model is no longer consistent when unification constraints such as those in HPSG are introduced.

Our solution is free of these problems, and is applicable to various grammars, not only HPSG and CCG.

Most of the state-of-the-art studies on parsing with lexicalized grammars have adopted feature forest models (Clark and Curran 2003, 2004b; Kaplan et al. 2004; Riezler and Vasserman 2004). Their methods of translating parse results into feature forests are basically the same as our method described in Section 4, and details differ because different grammar theories represent syntactic structures differently. They reported higher accuracy in parsing the Penn Treebank than the previous methods introduced herein, and these results attest the effectiveness of feature forest models in practical deep parsing. A remaining problem is that no studies could provide empirical comparisons across grammar theories. The above studies and our research evaluated parsing accuracy on their own test sets. The construction of theory-independent standard test sets requires enormous effort because we must establish theory-independent criteria such as agreed definitions of phrases and headedness. Although this issue is beyond the scope of the present article, it is a fundamental obstacle to the transparency of these studies on parsing.

Clark and Curran (2004a) described a method for reducing the cost of parsing a training treebank without sacrificing accuracy in the context of CCG parsing. They first assigned each word a small number of *supertags*, corresponding to lexical entries in our case, and parsed *supertagged sentences*. Because they did not use the probabilities of supertags in a parsing stage, their method corresponds to our "filtering only" method. The difference from our approach is that they also applied the supertagger in a parsing stage. We suppose that this was crucial for high accuracy in their approach, although empirical investigation is necessary.

6.3 Trade-Off between Dynamic Programming and Feature Locality

The proposed algorithm is an essential solution to the problem of estimating probabilistic models on exponentially many complete structures. However, the applicability of this algorithm relies on the constraint that features are defined locally in conjunctive nodes. As discussed in Section 6.1, this does not necessarily mean that features in our model can represent only the immediate-dominance relations of CFG rules, because conjunctive nodes may encode any fragments of complete structures. In fact, we demonstrated in Section 4.3 that certain assumptions allowed us to encode non-local predicate–argument dependencies in tractable-size feature forests. In addition, although in the experiments we used only features on bilexical dependencies, the method described in Section 4.3 allows us to define any features on a predicate and all of its arguments, such as a ternary relation among a subject, a verb, and a complement (e.g., the relation among *I*, *want*, and *dispute1* in Figure 21), and a generalized relation among semantic classes of a predicate and its arguments. This is because a predicate and all of its arguments are included in a conjunctive node, and feature functions can represent any relations expressed within a conjunctive node.

When we define more global features, such as co-occurrences of structures at distant places in a sentence, conjunctive nodes must be expanded so that they include all structures that are necessary to define these features. However, this obviously increases the number of conjunctive nodes, and consequently, the cost of parameter estimation increases. In an extreme case, for example, if we define features on any co-occurrences of partial parse trees, the full unpacking of parse forests would be necessary, and parameter estimation would be intractable. This indicates that there is a trade-off between the locality of features and the cost of estimation. That is, larger context features might

contribute to higher accuracy, while they inflate the size of feature forests and increase the cost of parameter estimation.

Sampling techniques (Rosenfeld 1997; Chen and Rosenfeld 1999b; Osborne 2000; Malouf and van Noord 2004) allow us to define any features on complete structures without any constraints. However, they force us to employ approximation methods for tractable computation. The effectiveness of those techniques therefore relies on convergence speed and approximation errors, which may vary depending on the characteristics of target problems and features.

It is an open research question whether dynamic programming or sampling can deliver a better balance of estimation efficiency and accuracy. The answer will differ in different problems. When most effective features can be represented locally in tractable-size feature forests, dynamic programming methods including ours are suitable. However, when global context features are indispensable for high accuracy, sampling methods might be better. We should also investigate compromise solutions such as dynamic CRFs (McCallum, Rohanimanesh, and Sutton 2003; Sutton, Rohanimanesh, and McCallum 2004) and reranking techniques (Collins 2000; Charniak and Johnson 2005). There is no analytical way of predicting the best solution, and it must be investigated experimentally for each target task.

7. Conclusion

A dynamic programming algorithm was presented for maximum entropy modeling and shown to provide a solution to the parameter estimation of probabilistic models of complete structures without the independence assumption. We first defined the notion of a feature forest, which is a packed representation of an exponential number of trees of features. When training data is represented with feature forests, model parameters are estimated at a tractable cost without unpacking the forests. The method provides a more flexible modeling scheme than previous methods of application of maximum entropy models to natural language processing. Furthermore, it is applicable to complex data structures where an event is difficult to decompose into independent sub-events.

We also demonstrated that feature forest models are applicable to probabilistic modeling of linguistic structures such as the syntactic structures of HPSG and predicate–argument structures including non-local dependencies. The presented approach can be regarded as a general solution to the probabilistic modeling of syntactic analysis with lexicalized grammars. Table 16 summarizes the best performance of the HPSG parser described in this article. The parser demonstrated impressively high coverage and accuracy for real-world texts. We therefore conclude that the HPSG parser for English is moving toward a practical level of use in real-world applications. Recently, the applicability of the HPSG parser to practical applications, such as information extraction and retrieval, has also been demonstrated (Miyao et al. 2006; Yakushiji et al. 2006; Chun 2007).

Table 16
Final results.

Parsing accuracy for Section 23 (<40 words)	
# parsed sentences	2,137/2,144 (99.7%)
Precision/recall	87.69%/87.16%
Sentential accuracy	39.2%

From our extensive investigation of HPSG parsing, we observed that exploration of new types of features is indispensable to further improvement of parsing accuracy. A possible research direction is to encode larger contexts of parse trees, which has been shown to improve accuracy (Toutanova and Manning 2002; Toutanova, Markova, and Manning 2004). Future work includes not only the investigation of these features but also the abstraction of predicate–argument dependencies using semantic classes. Experimental results also suggest that an improvement in grammar coverage is crucial for higher accuracy. This indicates that an improvement in the quality of the grammar is a key factor for the improvement of parsing accuracy.

The feature forest model provides new insight into the relationship between a linguistic structure and a unit of probability. Traditionally, a unit of probability was implicitly assumed to correspond to a meaningful linguistic structure; a tagging of a word or an application of a rewriting rule. One reason for the assumption is to enable dynamic programming algorithms, such as the Viterbi algorithm. The probability of a complete structure must be decomposed into atomic structures in which ambiguities are limited to a tractable size. Another reason is to estimate plausible probabilities. Because a probability is defined over atomic structures, they should also be meaningful so as to be assigned a probability. In feature forest models, however, conjunctive nodes are responsible for the former, whereas feature functions are responsible for the latter. Although feature functions must be defined locally in conjunctive nodes, they are not necessarily equivalent. Conjunctive nodes may represent any fragments of a complete structure, which are not necessarily linguistically meaningful. They should be designed to pack ambiguities and enable us to define useful features. Meanwhile, feature functions indicate an atomic unit of probability, and are designed to capture statistical regularity of the target problem. We expect the separation of a unit of probability from linguistic structures to open up a new framework for flexible probabilistic modeling.

Acknowledgments

The authors wish to thank the anonymous reviewers of *Computational Linguistics* for their helpful comments and discussions. We would also like to thank Takashi Ninomiya and Kenji Sagae for their precious support.

References

Abney, Steven P. 1997. Stochastic attribute-value grammars. *Computational Linguistics*, 23(4):597–618.

Baker, James K. 1979. Trainable grammars for speech recognition. In Jared J. Wolf and Dennis H. Klatt, editors, *Speech Communication Papers Presented at the 97th Meeting of the Acoustical Society of America.* MIT Press, Cambridge, MA, pages 547–550.

Baldridge, Jason and Miles Osborne. 2003. Active learning for HPSG parse selection. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 17–24, Edmonton, Canada.

Berger, Adam L., Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Burke, Michael, Aoife Cahill, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2004. Treebank-based acquisition of wide-coverage, probabilistic LFG resources: Project overview, results and evaluation. In *Proceedings of the IJCNLP-04 Workshop "Beyond Shallow Analyses"*, Hainan Island. Available at `www-tsujii.is.s.u-tokyo.ac.jp/bsa`.

Carpenter, Bob. 1992. *The Logic of Typed Feature Structures*. Cambridge University Press, Cambridge, England.

Carroll, John and Stephan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP-05)*, pages 165–176, Jeju Island.

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *Proceedings of the First Conference on North American Chapter of the Association for Computational*

Linguistics (NAACL 2000), pages 132–139, Seattle, WA.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 173–180, Ann Arbor, MI.

Chen, Stanley and Ronald Rosenfeld. 1999a. A Gaussian prior for smoothing maximum entropy models. Technical Report CMUCS-99-108, Carnegie Mellon University.

Chen, Stanley F. and Ronald Rosenfeld. 1999b. Efficient sampling and feature selection in whole sentence maximum entropy language models. In *Proceedings of the 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 549–552, Phoenix, AZ.

Chiang, David. 2003. Mildly context sensitive grammars for estimating maximum entropy parsing models. In *Proceedings of the 8th Conference on Formal Grammar*, pages 19–31, Vienna.

Chun, Hong-Woo. 2007. *Mining Literature for Disease-Gene Relations*. Ph.D. thesis, University of Tokyo.

Clark, Stephen and James R. Curran. 2003. Log-linear models for wide-coverage CCG parsing. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP 2003)*, pages 97–104, Sapporo.

Clark, Stephen and James R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 282–288, Geneva.

Clark, Stephen and James R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, pages 104–111, Barcelona.

Clark, Stephen, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures with a wide-coverage CCG parser. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 327–334, Philadephia.

Cohen, Paul R. 1995. *Empirical Methods for Artificial Intelligence*. The MIT Press, Cambridge, MA.

Collins, Michael. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL'97)*, pages 16–23, Madrid.

Collins, Michael. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Collins, Michael. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 175–182, Palo Alto, CA.

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

Copestake, Ann, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag. 1995. Translation using minimal recursion semantics. In *Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation (TMI95)*, pages 15–32, Leuven.

Copestake, Ann, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 2006. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(4):281–332.

Darroch, J. N. and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480.

Della Pietra, Stephen, Vincent Della Pietra, and John Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

Geman, Stuart and Mark Johnson. 2002. Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 279–286, Philadelphia, PA.

Gildea, Daniel. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*, pages 167–202, Pittsburgh, PA.

Hockenmaier, Julia. 2003. Parsing with generative models of predicate-argument structure. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 359–366, Sapporo.

Hockenmaier, Julia and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002)*, pages 1974–1981, Las Palmas.

Johnson, Mark, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, pages 535–541, College Park, Maryland.

Johnson, Mark and Stefan Riezler. 2000. Exploiting auxiliary distributions in stochastic unification-based grammars. In *Proceedings of the First Conference on North American Chapter of the Association for Computational Linguistics*, pages 154–161, Seattle, WA.

Kameya, Yoshitaka and Taisuke Sato. 2000. Efficient EM learning with tabulation for parameterized logic programs. In *Proceedings of the 1st International Conference on Computational Logic (CL2000)*, volume 1861 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 269–294, Imperial College, London.

Kaplan, Ronald M., Stefan Riezler, Tracy H. King, John T. Maxwell, III, Alexander Vasserman, and Richard Crouch. 2004. Speed and accuracy in shallow and deep stochastic parsing. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2004)*, pages 97–104, Boston, MA.

Lafferty, John, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning 2001*, pages 282–289, Williams College, Williamstown, MA.

Makino, Takaki, Yusuke Miyao, Kentaro Torisawa, and Jun'ichi Tsujii. 2002. Native-code compilation of feature structures. In Stephen Oepen, Dan Flickinger, Jun'ichi Tsujii, and Hans Uszkoreit, editors, *Collaborative Language Engineering: A Case Study in Efficient Grammar-based Parsing*. CSLI Publications, Palo Alto, CA, pages 49–80.

Malouf, Robert. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002)*, pages 1–7, Taipei.

Malouf, Robert and Gertjan van Noord. 2004. Wide coverage parsing with stochastic attribute value grammars. In *Proceedings of the IJCNLP-04 Workshop "Beyond Shallow Analyses"*, Hainan Island. Available at `www.tsujii.is.s.u-tokyo.ac.jp/bsa`.

Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, pages 114–119, Plainsboro, NJ.

Maxwell John T., III and Ronald M. Kaplan. 1995. A method for disjunctive constraint satisfaction. In Mary Dalrymple, Ronald M. Kaplan, John T. Maxwell, III, and Annie Zaenen, editors, *Formal Issues in Lexical-Functional Grammar*, number 47 in CSLI Lecture Notes Series. CSLI Publications, Palo Alto, CA, chapter 14, pages 381–481.

McCallum, Andrew and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the 7th Conference on Natural Language Learning (CoNLL)*, pages 188–191, Edmonton.

McCallum, Andrew, Khashayar Rohanimanesh, and Charles Sutton. 2003. Dynamic conditional random fields for jointly labeling multiple sequences. In *Proceedings of the Workshop on Syntax, Semantics, Statistics at the 16th Annual Conference on Neural Information Processing Systems*, Vancouver. Available at `www.cs.umasse.du/~mccallum/papers/derf-nips03.pdf`.

Miyao, Yusuke, Takashi Ninomiya, and Jun'ichi Tsujii. 2003. Probabilistic modeling of argument structures including non-local dependencies. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2003)*, pages 285–291, Borovets.

Miyao, Yusuke, Takashi Ninomiya, and Jun'ichi Tsujii. 2005. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the Penn Treebank. In *Natural Language Processing - IJCNLP 2004*, pages 684–693, Hainan Island.

Miyao, Yusuke, Tomoko Ohta, Katsuya Masuda, Yoshimasa Tsuruoka, Kazuhiro Yoshida, Takashi Ninomiya, and Jun'ichi Tsujii. 2006. Semantic retrieval for the accurate identification of relational concepts in massive textbases. In *Proceedings of the Joint Conference of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the*

Association for Computational Linguistics (COLING-ACL 2006), pages 1017–1024, Sydney.

Miyao, Yusuke and Jun'ichi Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proceedings of the Human Language Technology Conference (HLT-2002)*, pages 292–297, San Diego, CA.

Miyao, Yusuke and Jun'ichi Tsujii. 2003. A model of syntactic disambiguation based on lexicalized grammars. In *Proceedings of the Seventh Conference on Computational Natural Language Learning (CoNLL-2003)*, pages 1–8, Edmonton.

Miyao, Yusuke and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 83–90, Ann Arbor, MI.

Nakanishi, Hiroko, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic models for disambiguation of an HPSG-based chart generator. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT 2005)*, pages 93–102, Vancouver.

Ninomiya, Takashi, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Efficacy of beam thresholding, unification filtering and hybrid parsing in probabilistic HPSG parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies*, pages 103–114, Vancouver.

Nocedal, Jorge. 1980. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35:773–782.

Nocedal, Jorge and Stephen J. Wright. 1999. *Numerical Optimization*. Springer, New York.

Oepen, Stephan and John Carroll. 2000. Ambiguity packing in constraint-based parsing: practical results. In *Proceedings of the First Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2000)*, pages 162–169, Seattle, WA.

Oepen, Stephan, Dan Flickinger, Jun'ichi Tsujii, and Hans Uszkoreit, editors. 2002. *Collaborative Language Engineering: A Case Study in Efficient Grammar-Based Processing*. CSLI Publications, Palo Alto, CA.

Oepen, Stephan, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. 2002. The LinGO Redwoods treebank motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, volume 2, pages 1–5, Taipei.

Osborne, Miles. 2000. Estimation of stochastic attribute-value grammar using an informative sample. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, volume 1, pages 586–592, Saarbrücken.

Peng, Fuchun and Andrew McCallum. 2004. Accurate information extraction from research papers using conditional random fields. In *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT/NAACL-04)*, pages 329–336, Boston, MA.

Pinto, David, Andrew McCallum, Xen Lee, and W. Bruce Croft. 2003. Table extraction using conditional random fields. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2003)*, pages 235–242, Toronto.

Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, IL.

Riezler, Stefan, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 271–278, Philadephia, PA.

Riezler, Stefan, Detlef Prescher, Jonas Kuhn, and Mark Johnson. 2000. Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL 2000)*, pages 480–487, Hong Kong.

Riezler, Stefan and Alexander Vasserman. 2004. Incremental feature selection and l_1 regularization for relaxed maximum-entropy modeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 174–181, Barcelona.

Roark, Brian, Murat Saraclar, Michael Collins, and Mark Johnson. 2004. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, pages 47–54, Barcelona.

Rosenfeld, Ronald. 1997. A whole sentence maximum entropy language model. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 230–237, Santa Barbara, CA.

Sag, Ivan A., Thomas Wasow, and Emily M. Bender. 2003. *Syntactic Theory: A Formal Introduction*. Number 152 in CSLI Lecture Notes. CSLI Publications, Standford, CA.

Sarawagi, Sunita and William W. Cohen. 2004. Semi-Markov conditional random fields for information extraction. In *Proceedings of the 18th Annual Conference on Neural Information Processing Systems*, pages 1185–1192, Vancouver.

Sato, Taisuke. 2005. A generic approach to em learning for symbolic-statistical models. In *Proceedings of the 4th Learning Language in Logic Workshop (LLL05)*, pages 21–28, Bonn.

Sato, Taisuke and Yoshitaka Kameya. 1997. PRISM: a language for symbolic-statistical modeling. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI '97)*, pages 1330–1335, Nagoya.

Sato, Taisuke and Yoshitaka Kameya. 2001. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, 15:391–454.

Settles, Burr. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA)*, pages 104–107, Geneva.

Sha, Fei and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, pages 213–220, Edmonton.

Shieber, Stuart M. 1985. Using restriction to extend parsing algorithms for complex-feature-based formalisms. In *Proceedings of the 23rd Annual Meeting on Association for Computational Linguistics*, pages 145–152, Chicago, IL.

Sutton, Charles, Khashayar Rohanimanesh, and Andrew McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proceedings of the 21st International Conference on Machine Learning (ICML 2004)*, pages 783–790, Alberta.

Taskar, Ben, Dan Klein, Michael Collins, Daphne Koller, and Chris Manning. 2004. Max-margin parsing. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 1–8, Barcelona.

Toutanova, Kristina and Christopher Manning. 2002. Feature selection for a rich HPSG grammar using decision trees. In *Proceedings of the Sixth Conference on Natural Language Lerning (CoNLL-2002)*, pages 77–83, Taipei.

Toutanova, Kristina, Penka Markova, and Christopher Manning. 2004. The leaf projection path view of parse trees: Exploring string kernels for HPSG parse selection. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 166–173, Barcelona.

Tsujii Laboratory. 2004. Enju—A practical HPSG parser. Available at `http://www.tsujii.is.s.u-tokyo.ac.jp/enju/`.

Tsuruoka, Yoshimasa, Yusuke Miyao, and Jun'ichi Tsujii. 2004. Towards efficient probabilistic HPSG parsing: Integrating semantic and syntatic preference to guide the parsing. In *Proceedings of the IJCNLP-04 Workshop "Beyond Shallow Analyses"*, Hainan Island. Available at `www.tsujii.is.s.u-tokyo.ac.jp/bsa`.

Tsuruoka, Yoshimasa and Jun'ichi Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, pages 467–474, Vancouver.

Vapnik, Vladimir N. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

Vijay-Shanker, K., David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Palo Alto, CA.

Weir, David J. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania.

Yakushiji, Akane, Yusuke Miyao, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2006. Automatic construction of predicate-argument structure patterns for biomedical information extraction. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 284–292, Sydney.