Probabilistic disambiguation models for wide-coverage HPSG parsing

Yusuke Miyao
Department of Computer Science
University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan
yusuke@is.s.u-tokyo.ac.jp

Jun'ichi Tsujii
Department of Computer Science
University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan
CREST, JST
tsujii@is.s.u-tokyo.ac.jp

Abstract

This paper reports the development of log-linear models for the disambiguation in wide-coverage HPSG parsing. The estimation of log-linear models requires high computational cost, especially with wide-coverage grammars. Using techniques to reduce the estimation cost, we trained the models using 20 sections of Penn Treebank. A series of experiments empirically evaluated the estimation techniques, and also examined the performance of the disambiguation models on the parsing of real-world sentences.

1 Introduction

Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) has been studied extensively from both linguistic and computational points of view. However, despite research on HPSG processing efficiency (Oepen et al., 2002a), the application of HPSG parsing is still limited to specific domains and short sentences (Oepen et al., 2002b; Toutanova and Manning, 2002). Scaling up HPSG parsing to assess real-world texts is an emerging research field with both theoretical and practical applications.

Recently, a wide-coverage grammar and a large treebank have become available for English HPSG (Miyao et al., 2004). A large treebank can be used as training and test data for statistical models. Therefore, we now have the basis for the development and the evaluation of statistical disambiguation models for wide-coverage HPSG parsing.

The aim of this paper is to report the development of log-linear models for the disambiguation in wide-coverage HPSG parsing, and their empirical evaluation through the parsing of the Wall Street Journal of Penn Treebank II (Marcus et al., 1994). This is challenging because the estimation of log-linear models is computationally expensive, and we require solutions to make the model estimation tractable. We apply two techniques for reducing the training cost. One is the estimation on a packed representation of HPSG parse trees (Section 3). The other is the filtering of parse candidates according to a preliminary probability distribution (Section 4).

To our knowledge, this work provides the first results of extensive experiments of parsing Penn Treebank with a probabilistic HPSG. The results from the Wall Street Journal are significant because the complexity of the sentences is different from that of short sentences. Experiments of the parsing of real-world sentences can properly evaluate the effectiveness and possibility of parsing models for HPSG.

2 Disambiguation models for HPSG

Discriminative log-linear models are now becoming a de facto standard for probabilistic disambiguation models for deep parsing (Johnson et al., 1999; Riezler et al., 2002; Geman and Johnson, 2002; Miyao and Tsujii, 2002; Clark and Curran, 2004b; Kaplan et al., 2004). Previous studies on probabilistic models for HPSG (Toutanova and Manning, 2002; Baldridge and Osborne, 2003; Malouf and van Noord, 2004) also adopted log-linear models. HPSG exploits feature structures to represent linguistic constraints. Such constraints are known

Proceedings of the 43rd Annual Meeting of the ACL, pages 83–90,
Ann Arbor, June 2005.

to introduce inconsistencies in probabilistic models estimated using simple relative frequency (Abney, 1997). Log-linear models are required for credible probabilistic models and are also beneficial for incorporating various overlapping features.

This study follows previous studies on the probabilistic models for HPSG. The probability, $p(t|s)$, of producing the parse result t from a given sentence s is defined as

$$p(t|s) = \frac{1}{Z_s} p_0(t|s) \exp(\sum_i f_i(t,s)\lambda_i(t,s))$$

$$Z_s = \sum_{t' \in T(s)} p_0(t'|s) \exp(\sum_i f_i(t',s)\lambda_i(t',s)),$$

where $p_0(t|s)$ is a reference distribution (usually assumed to be a uniform distribution), and $T(s)$ is a set of parse candidates assigned to s. The feature function $f_i(t,s)$ represents the characteristics of t and s, while the corresponding model parameter $\lambda_i(t,s)$ 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 above model requires a set of pairs $\langle t_s, T(s)\rangle$, where t_s is the correct parse for sentence s. While t_s is provided by a treebank, $T(s)$ is computed by parsing each s in the treebank. Previous studies assumed $T(s)$ could be enumerated; however, the assumption is impractical because the size of $T(s)$ is exponentially related to the length of s. The problem of exponential explosion is inevitable in the wide-coverage parsing of real-world texts because many parse candidates are produced to support various constructions in long sentences.

3 Packed representation of HPSG parse trees

To avoid exponential explosion, we represent $T(s)$ in a packed form of HPSG parse trees. A parse tree of HPSG is represented as a set of tuples $\langle m, l, r\rangle$, where m, l, and r are the signs of mother, left daughter, and right daughter, respectively[1]. In chart parsing, partial parse candidates are stored in a *chart*, in which phrasal signs are identified and packed into an equivalence class if they are determined to be equivalent and dominate the same word sequence. A set

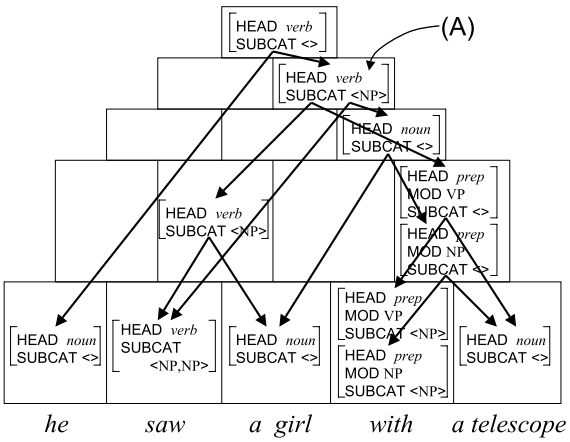

Figure 1: Chart for parsing *"he saw a girl with a telescope"*

of parse trees is then represented as a set of relations among equivalence classes.

Figure 1 shows a chart for parsing *"he saw a girl with a telescope"*, where the modifiee (*"saw"* or *"girl"*) of *"with"* is ambiguous. 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). Since the signs of the top-most nodes are equivalent, they are packed into an equivalence class. The ambiguity is represented as two pairs of arrows that come out of the node.

Formally, 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 the chart can be interpreted as an instance of *a feature forest* (Miyao and Tsujii, 2002; Geman and Johnson, 2002). A feature forest is an "and/or" graph to represent exponentially-many tree structures in a packed form. If $T(s)$ is represented in a feature forest, $p(t|T(s))$ can be estimated using dynamic programming without unpacking the chart. A feature forest is formally defined as a tuple, $\langle C, D, R, \gamma, \delta\rangle$, where C is a set of conjunctive nodes, D is a set of disjunctive nodes, $R \subseteq C$ is a set of root nodes[2], $\gamma : D \rightarrow 2^C$ is a conjunctive daughter function, and $\delta : C \rightarrow 2^D$ is a disjunctive

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

[2] For the ease of explanation, the definition of root node is slightly different from the original.

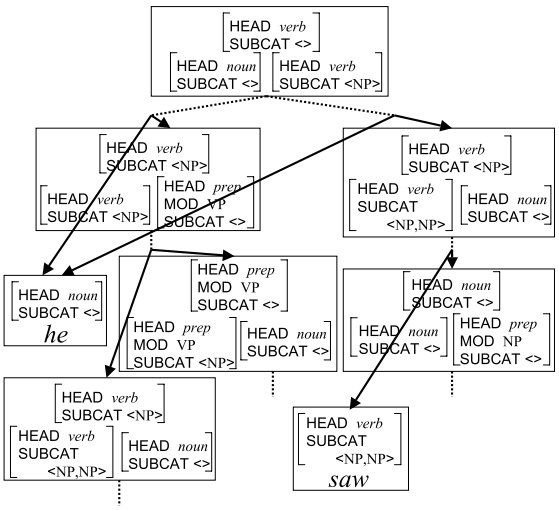

Figure 2: Packed representation of HPSG parse trees in Figure 1

daughter function. The feature functions $f_i(t, s)$ are assigned to conjunctive nodes.

The simplest way to map a chart of HPSG parse trees into a feature forest is to map each equivalence class $e \in E$ to a conjunctive node $c \in C$. However, in HPSG parsing, important features for disambiguation are combinations of a mother and its daughters, i.e., $\langle m, l, r \rangle$. Hence, 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 2 shows (a part of) the HPSG parse trees in Figure 1 represented as a feature forest. Square boxes are conjunctive nodes, dotted lines express a disjunctive daughter function, and solid arrows represent a conjunctive daughter function.

The mapping is formally defined as follows.

- $C = \{\langle e_m, e_l, e_r \rangle | e_m \in E \wedge e_l, e_r \in E. \langle e_l, e_r \rangle \in \alpha(e_m)\}$,
- $D = E$,
- $R = \{\langle e_m, e_l, e_r \rangle | e_m \in E_r \wedge \langle e_m, e_l, e_r \rangle \in C\}$,
- $\gamma = \{\langle e_m, C(e_m) \rangle | e_m \in E \wedge C(e_m) = \{\langle e_m, e_l, e_r \rangle | e_l, e_r \in E. \langle e_l, e_r \rangle \in \alpha(e_m)\}\}$, and
- $\delta = \{\langle \langle e_m, e_l, e_r \rangle, \{e_l, e_r\} \rangle | \langle e_m, e_l, e_r \rangle \in C\}$.

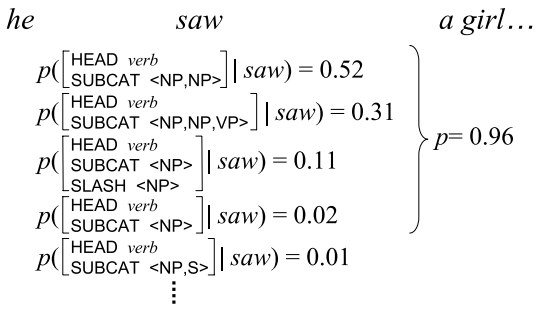

Figure 3: Filtering of lexical entries for *"saw"*

4 Filtering by preliminary distribution

The above method allows for the tractable estimation of log-linear models on exponentially-many HPSG parse trees. However, despite the development of methods to improve HPSG parsing efficiency (Oepen et al., 2002a), the exhaustive parsing of all sentences in a treebank is still expensive.

Our idea is that we can omit the computation of parse trees with low probabilities in the estimation stage because $T(s)$ can be approximated with 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, $\bar{p}(t|s) = \prod_{w \in s} p(l|w)$, where $w \in s$ is a word in the sentence s, and l is a lexical entry assigned to w. This model can be estimated without parsing 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 the lexical entry necessary to produce the correct parse is not assigned, it is additionally assigned to the word.

Figure 3 shows an example of filtering 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. This method reduces the time for

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

Table 1: Templates of atomic features

parsing a treebank, while this approximation causes bias in the training data and results in lower accuracy. The trade-off between the parsing cost and the accuracy will be examined experimentally.

We have several ways to integrate $\bar{p}$ with the estimated model $p(t|T(s))$. 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.

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 was shown to be a generalization of the reference distribution method (Johnson and Riezler, 2000).

5 Features

Feature functions in the log-linear models are designed to capture the characteristics of $\langle e_m, e_l, e_r \rangle$. In this paper, we investigate combinations of the atomic features listed in Table 1. The following combinations are used for representing the characteristics of the binary/unary schema applications.

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

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

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

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

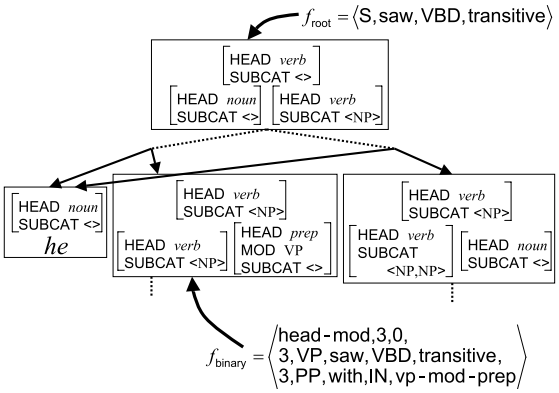

Figure 4: Example features

Figure 4 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 the lexical entry is a VP-modifying preposition.

In an actual implementation, some of the atomic features are abstracted (i.e., ignored) for smoothing. Table 2 shows a full set of templates of combined features used in the experiments. Each row represents a template of a feature function. A check means the atomic feature is incorporated while a hyphen means the feature is ignored.

Restricting the domain of feature functions to $\langle e_m, e_l, e_r \rangle$ seems to limit the flexibility of feature design. Although it is true to some extent, this 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, in our previous study (Miyao et al., 2003), we successfully developed a probabilistic model including features on nonlocal predicate-argument dependencies. However, since we could not observe significant improvements by incorporating nonlocal features, this paper investigates only the features described above.

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

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

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

Table 2: Feature templates for binary schema (left), unary schema (center), and root condition (right)

	Avg. length	LP	LR	UP	UR	F-score
Section 22 (< 40 words)	20.69	87.18	86.23	90.67	89.68	86.70
Section 22 (< 100 words)	22.43	86.99	84.32	90.45	87.67	85.63
Section 23 (< 40 words)	20.52	87.12	85.45	90.65	88.91	86.27
Section 23 (< 100 words)	22.23	86.81	84.64	90.29	88.03	85.71

Table 3: Accuracy for development/test sets

6 Experiments

We used an HPSG grammar derived from Penn Treebank (Marcus et al., 1994) Section 02-21 (39,832 sentences) by our method of grammar development (Miyao et al., 2004). The training data was the HPSG treebank derived from the same portion of the Penn Treebank[3]. For the training, we eliminated sentences with no less than 40 words and for which the parser could not produce the correct parse. The resulting training set consisted of 33,574 sentences. The treebanks derived from Sections 22 and 23 were used as the development (1,644 sentences) and final test sets (2,299 sentences). We measured the accuracy of predicate-argument dependencies output by the parser. A dependency is defined as a tuple $\langle \pi, w_h, a, w_a \rangle$, where π is the predicate type (e.g., adjective, intransitive verb), w_h is the head word of the predicate, a is the argument label (MODARG, ARG1, ..., ARG4), and w_a is the head word of the argument. Labeled precision/recall (LP/LR) is the ratio of tuples correctly identified by the parser, while unlabeled precision/recall (UP/UR) is the ratio of w_h and w_a correctly identified regardless of π and a. The F-score is the harmonic mean of LP and LR. The accuracy was measured by parsing test sentences with part-of-speech tags provided by the treebank. The Gaussian prior was used for smoothing (Chen and Rosenfeld, 1999), and its hyper-parameter was tuned for each model to maximize the F-score for the development set. The optimization algorithm was the limited-memory BFGS method (Nocedal and Wright, 1999). All the following experiments were conducted on AMD Opteron servers with a 2.0-GHz CPU and 12-GB memory.

Table 3 shows the accuracy for the development/test sets. Features occurring more than twice were included in the model (598,326 features). Filtering was done by the reference distribution method with $n = 10$ and $\epsilon = 0.95$. The unigram model for filtering was a log-linear model with two feature templates, $\langle$WORD, POS, LE$\rangle$ and $\langle$POS, LE$\rangle$ (24,847 features). Our results cannot be strictly compared with other grammar formalisms because each formalism represents predicate-argument dependencies differently; for reference, our results are competitive with the corresponding measures reported for Combinatory Categorial Grammar (CCG) (LP/LR = 86.6/86.3) (Clark and Curran, 2004b). Different from the results of CCG and PCFG (Collins, 1999; Charniak, 2000), the recall was clearly lower than precision. This results from the HPSG grammar having stricter feature constraints and the parser not being able to produce parse results for around one percent of the sentences. To improve recall, we need techniques of robust processing with HPSG.

[3]The programs to make the grammar and the treebank from Penn Treebank are available at `http://www-tsujii.is.s.u-tokyo.ac.jp/enju/`.

	LP	LR	Estimation time (sec.)
Filtering only	34.90	23.34	702
Product	86.71	85.55	1,758
Reference dist.	87.12	85.45	655
Feature function	84.89	83.06	1,203

Table 4: Estimation method vs. accuracy and estimation time

n, ϵ	F-score	Estimation time (sec.)	Parsing time (sec.)	Memory usage (MB)
5, 0.80	84.31	161	7,827	2,377
5, 0.90	84.69	207	9,412	2,992
5, 0.95	84.70	240	12,027	3,648
5, 0.98	84.81	340	15,168	4,590
10, 0.80	84.79	164	8,858	2,658
10, 0.90	85.77	298	13,996	4,062
10, 0.95	86.27	654	25,308	6,324
10, 0.98	86.56	1,778	55,691	11,700
15, 0.80	84.68	180	9,337	2,676
15, 0.90	85.85	308	14,915	4,220
15, 0.95	86.68	854	32,757	7,766

Table 5: Filtering threshold vs. accuracy and estimation time

Table 4 compares the estimation methods introduced in Section 4. In all of the following experiments, we show the accuracy for the test set ($<$ 40 words) only. Table 4 revealed that our simple method of filtering caused a fatal bias in training data when a preliminary distribution was used only for filtering. However, the model 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 is that a hyper-parameter of the prior was set to the same value for all the features including the feature of the preliminary distribution.

Table 5 shows 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 85.0.

Figure 5 shows the accuracy for each sentence length. It is apparent from this figure that the accuracy was significantly higher for shorter sentences ($<$ 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.

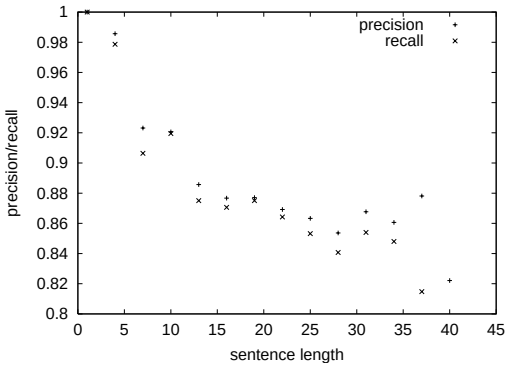

Figure 5: Sentence length vs. accuracy

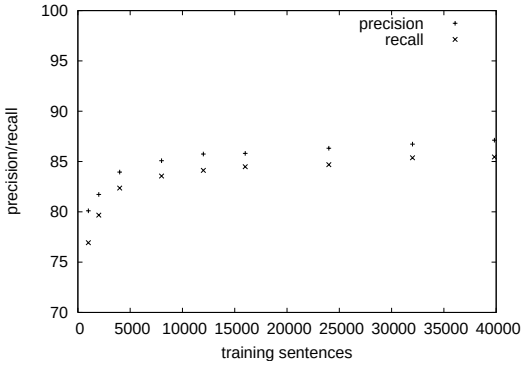

Figure 6: Corpus size vs. accuracy

Figure 6 shows the learning curve. A feature set was fixed, while the parameter of the prior was optimized for each model. High accuracy was attained even with small data, and the accuracy seemed to be saturated. This indicates that we cannot further improve the accuracy simply by increasing training data. The exploration of new types of features is necessary for higher accuracy.

Table 6 shows the accuracy with difference feature sets. The accuracy was measured by removing some of the atomic features from the final model. The last row denotes the accuracy attained by the preliminary model. The numbers in bold type represent that the difference from the final model was significant according to stratified shuffling tests (Cohen, 1995) with p-value < 0.05. The results indicate that DIST, COMMA, SPAN, WORD, and POS features contributed to the final accuracy, although the dif-

Features	LP	LR	# features
All	87.12	85.45	623,173
−RULE	86.98	85.37	620,511
−DIST	**86.74**	**85.09**	603,748
−COMMA	**86.55**	**84.77**	608,117
−SPAN	**86.53**	**84.98**	583,638
−SYM	86.90	85.47	614,975
−WORD	**86.67**	**84.98**	116,044
−POS	**86.36**	**84.71**	430,876
−LE	87.03	85.37	412,290
−DIST,SPAN	**85.54**	**84.02**	294,971
−DIST,SPAN, COMMA	**83.94**	**82.44**	286,489
−RULE,DIST, SPAN,COMMA	**83.61**	**81.98**	283,897
−WORD,LE	**86.48**	**84.91**	50,258
−WORD,POS	**85.56**	**83.94**	64,915
−WORD,POS,LE	**84.89**	**83.43**	33,740
−SYM,WORD, POS,LE	**82.81**	**81.48**	26,761
None	**78.22**	**76.46**	24,847

Table 6: Accuracy with different feature sets

Error cause	# of errors
Argument/modifier distinction	58
temporal noun	21
to-infinitive	15
others	22
Attachment	53
prepositional phrase	18
to-infinitive	10
relative clause	8
others	17
Lexical ambiguity	42
participle/adjective	15
preposition/modifier	14
others	13
Comma	19
Coordination	14
Noun phrase identification	13
Zero-pronoun resolution	9
Others	17

Table 7: Error analysis

ferences were slight. In contrast, RULE, SYM, and LE features did not affect the accuracy. However, if each of them was removed together with another feature, the accuracy decreased drastically. This implies that such features had overlapping information.

Table 7 shows the manual classification of the causes of errors in 100 sentences randomly chosen from the development set. In our evaluation, one error source may cause multiple errors of dependencies. For example, if a wrong lexical entry was assigned to a verb, all the argument dependencies of the verb are counted as errors. The numbers in the table include such double-counting. Major causes were classified into three types: argument/modifier distinction, attachment ambiguity, and lexical ambiguity. While attachment/lexical ambiguities are well-known causes, the other is peculiar to deep parsing. Most of the errors cannot be resolved by features we investigated in this study, and the design of other features is crucial for further improvements.

7 Discussion and related work

Experiments on deep parsing of Penn Treebank have been reported for Combinatory Categorial Grammar (CCG) (Clark and Curran, 2004b) and Lexical Functional Grammar (LFG) (Kaplan et al., 2004). They developed log-linear models on a packed representation of parse forests, which is similar to our representation. Although HPSG exploits further complicated feature constraints and requires high computational cost, our work has proved that log-linear models can be applied to HPSG parsing and attain accurate and wide-coverage parsing.

Clark and Curran (2004a) described a method of reducing the cost of parsing a training treebank in the context of CCG parsing. They first assigned to each word a small number of *supertags*, which correspond to lexical entries in our case, and parsed *supertagged sentences*. Since they did not mention the probabilities of supertags, their method corresponds to our "filtering only" method. However, they also applied the same supertagger in a parsing stage, and this seemed to be crucial for high accuracy. This means that they estimated the probability of producing a parse tree from a supertagged sentence.

Another approach to estimating log-linear models for HPSG is to extract a small *informative sample* from the original set $T(s)$ (Osborne, 2000). Malouf and van Noord (2004) successfully applied this method to German HPSG. The problem with this method was in the approximation of exponentially many parse trees by a polynomial-size sample. However, their method has the advantage that any features on a parse tree can be incorporated into the model. The trade-off between approximation and locality of features is an outstanding problem.

Other discriminative classifiers were applied to the disambiguation in HPSG parsing (Baldridge and Osborne, 2003; Toutanova et al., 2004). The problem of exponential explosion is also inevitable for

their methods. An approach similar to ours may be applied to them, following the study on the learning of a discriminative classifier for a packed representation (Taskar et al., 2004).

As discussed in Section 6, exploration of other features is indispensable to further improvements. A possible direction is to encode larger contexts of parse trees, which were shown to improve the accuracy (Toutanova and Manning, 2002; Toutanova et al., 2004). Future work includes the investigation of such features, as well as the abstraction of lexical dependencies like semantic classes.

References

S. P. Abney. 1997. Stochastic attribute-value grammars. *Computational Linguistics*, 23(4).

J. Baldridge and M. Osborne. 2003. Active learning for HPSG parse selection. In *CoNLL-03*.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. NAACL-2000*, pages 132–139.

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

S. Clark and J. R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In *Proc. COLING-04*.

S. Clark and J. R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In *Proc. 42th ACL*.

P. R. Cohen. 1995. *Empirical Methods for Artificial Intelligence*. MIT Press.

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

S. Geman and M. Johnson. 2002. Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *Proc. 40th ACL*.

M. Johnson and S. Riezler. 2000. Exploiting auxiliary distributions in stochastic unification-based grammars. In *Proc. 1st NAACL*.

M. Johnson, S. Geman, S. Canon, Z. Chi, and S. Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proc. ACL'99*, pages 535–541.

R. M. Kaplan, S. Riezler, T. H. King, J. T. Maxwell III, and A. Vasserman. 2004. Speed and accuracy in shallow and deep stochastic parsing. In *Proc. HLT/NAACL'04*.

R. Malouf and G. van Noord. 2004. Wide coverage parsing with stochastic attribute value grammars. In *Proc. IJCNLP-04 Workshop "Beyond Shallow Analyses"*.

R. Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proc. CoNLL-2002*.

M. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*.

Y. Miyao and J. Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proc. HLT 2002*.

Y. Miyao, T. Ninomiya, and J. Tsujii. 2003. Probabilistic modeling of argument structures including non-local dependencies. In *Proc. RANLP 2003*, pages 285–291.

Y. Miyao, T. Ninomiya, and J. Tsujii. 2004. Corpus-oriented grammar development for acquiring a Head-driven Phrase Structure Grammar from the Penn Treebank. In *Proc. IJCNLP-04*.

J. Nocedal and S. J. Wright. 1999. *Numerical Optimization*. Springer.

S. Oepen, D. Flickinger, J. Tsujii, and H. Uszkoreit, editors. 2002a. *Collaborative Language Engineering: A Case Study in Efficient Grammar-Based Processing*. CSLI Publications.

S. Oepen, K. Toutanova, S. Shieber, C. Manning, D. Flickinger, and T. Brants. 2002b. The LinGO, Redwoods treebank. motivation and preliminary applications. In *Proc. COLING 2002*.

M. Osborne. 2000. Estimation of stochastic attribute-value grammar using an informative sample. In *Proc. COLING 2000*.

C. Pollard and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

S. Riezler, T. H. King, R. M. Kaplan, R. Crouch, J. T. Maxwell III, and M. Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proc. 40th ACL*.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *EMNLP 2004*.

K. Toutanova and C. D. Manning. 2002. Feature selection for a rich HPSG grammar using decision trees. In *Proc. CoNLL-2002*.

K. Toutanova, P. Markova, and C. Manning. 2004. The leaf projection path view of parse trees: Exploring string kernels for HPSG parse selection. In *EMNLP 2004*.