Fast Parsing for Probabilistic Head-driven Phrase Structure Grammars

Yoshimasa Tsuruoka†‡ Yusuke Miyao‡ Takashi Ninomiya†‡ Jun'ichi Tsujii‡†
†CREST, JST (Japan Science and Technology Agency)
Honcho 4-1-8, Kawaguchi-shi, Saitama 332-0012
‡Department of Computer Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033
{tsuruoka,yusuke,ninomi,tsujii}@is.s.u-tokyo.ac.jp

Abstract

We present a framework for fast parsing with probabilistic Head-driven Phrase Structure Grammars (HPSG). The parser can integrate semantic and syntactic preference into figures-of-merit (FOMs) with the *equivalence class function* during parsing, and reduce the search space by dynamic programming and using the integrated FOMs. We apply a beam thresholding technique to the parsing algorithm and evaluated its effectiveness on the Penn Treebank corpus. Experimental results show that the parser can achieve significant speedup compared to full parsing strategies on the expense of slight loss of recall.

1 Introduction

Probabilistic modeling of unification-based grammars including HPSG (Pollard and Sag, 1994) has received a great deal of attention for the last decade. Log-linear (or Maximum Entropy) modeling provides a promising framework for HPSG in terms of estimating model parameters (Abney, 1997; Johnson et al., 1999; Miyao et al., 2003).

The computational cost required for parsing is a major concern for probabilistic HPSG. One way to obtain the Viterbi (highest probability) parse given a sentence and a probabilistic model is to first perform full parsing, which gives all possible parses regardless of their probabilities, and then choose the highest probability parse by evaluating the probability of every parse result. However, such a naive strategy often requires prohibitive computational cost.

Geman and Johnson (2002) proposed a dynamic programming algorithm for finding the Viterbi parse in a packed parse forest generated by unification-based grammars. Their algorithm enables us to select the highest probability parse without expanding packed parse forest. The behavior of their algorithm is like the Viterbi algorithm for PCFG, and hence the correct parse of the highest probability is guaranteed. The interesting idea of their approach is to first enforce full parsing because the probabilities of non-local dependencies, which cannot be computed during parsing, can be computed after full parsing. However, their efficiency is inherently limited by the computational cost of full parsing.

This paper presents a framework for parsing to obtain the Viterbi parse given an HPSG and its probabilistic model. We define the *equivalence class function* to reduce multiple feature structures to a single feature structure that gives the same resulting figure-of-merit (FOM). With this function, the parser can integrate semantic and syntactic preference into FOMs during parsing, and reduce the search space by dynamic programming and using the integrated FOMs.

We present the CKY parsing algorithm using the equivalence class function for probabilistic HPSG, and apply a beam thresholding technique to the parsing algorithm. The performance of the parser is evaluated on the Penn Treebank corpus.

This paper is organized as follows. Section 2 describes the previous work for efficient parsing with the grammars like HPSG. Section 3 gives the definitions of the probabilistic HPSG. Section 4 presents the equivalence class function and the CKY parsing algorithm using this function. Experimental results using the Penn Treebank corpus are presented in Section 5. Section 6 offers some concluding remarks.

2 Related work

Many of the methods for improving parsing efficiency of deep linguistic analysis have been studied in the frameworks of grammars such as Lexical Functional Grammar (LFG) (Bresnan, 1982), Lexicalized Tree Adjoining Grammar (LTAG) (Shabes et al., 1988), Head-driven Phrase Structure Grammars (Pollard and Sag, 1994), and Combinatory Categorial Grammar (CCG) (Steedman, 2000). Most of them are proposed for full parsing, i.e., all-paths search without FOM (Matsumoto et al., 1983; Maxwell and Kaplan, 1993; van Noord, 1997; Kiefer et al., 1999; Malouf et al., 2000; Torisawa et al., 2000; Penn and Munteanu, 2003). The full parsing strategy is widely used in grammar development, training of parameters for the probabilistic models, or finding the most probable parse among all parses derived by full parsing. However, full parsing is far too expensive in practice because it exhaustively searches all parses derived by the given grammar, especially in case of automatically acquired wide-coverage grammars.

Deep linguistic analysis by unification-based grammars is employed in the VERBMOBIL spontaneous speech translation system (Wahlster, 1993; Kasper et al., 1996). Because their system supposes real-time processing of spontaneous speech, efficiency and robustness were their great concern. Kasper et al. (1996) proposed a probabilistic model where probabilities are assigned to the CFG backbone of the unification-based grammar, and the most probable parse is found by PCFG parsing. After PCFG parsing, the most probable CFG parse is selected and re-parsed by the original unification-based grammar. This process is repeated until parsing by the original unification-based grammar succeeds. Their probabilistic model is based on PCFG and not the probabilistic unification-based grammar parsing.

Many algorithms for improving efficiency of PCFG parsing are extensively studied including grammar compilation (Tomita, 1986; Nederhof, 2000), the Viterbi algorithm, controlling search strategies without FOM such as left-corner parsing (Rosenkrantz and Lewis II, 1970) or head-corner parsing (Kay, 1970; van Noord, 1997), and with FOM such as the beam search, the best-first search or A* search (Chitrao and Grishman, 1990; Caraballo and Charniak, 1998; Collins, 1999; Ratnaparkhi, 1999; Roark, 2001; Klein and Manning, 2003). The beam search or the best-first search significantly reduces the time required for finding the best parse at the cost of losing guarantee of correct parse. However, their algorithms cannot simply be applied to the probabilistic unification-based grammar parsing because they assume the locality of probabilities, which cannot be assumed in probabilistic models of unification-based grammars. For example, non-local constraints in unification-based grammars, such as constraints for wh-movement or predicate argument relations, break the locality of probabilities. This is mainly because the PCFG parsing has a coincidence of the locality of the probability and the locality of the process, which the probabilistic unification-based grammar parsing does not have.

3 Probabilistic model of HPSG

In HPSG, a small number of schemata explain general grammatical constraints, while a large number of lexical entries express word-specific characteristics. Both schemata and lexical entries are represented by typed feature structures, and constraints represented by feature structures are checked with *unification* (for details, see (Pollard and Sag, 1994)). Figure 1 shows an example of HPSG parsing of the sentence "*Spring has come.*" First, each of the lexical entries for "*has*" and "*come*" is unified with a daughter feature structure of the Head-Complement Schema. Unification provides the phrasal sign of the mother. The sign of the larger constituent is obtained by repeatedly applying schemata to lexical/phrasal signs. Finally, the parse result is output as a phrasal sign that dominates the entire sentence.

Given set $\mathcal{W}$ of words and set $\mathcal{F}$ of feature

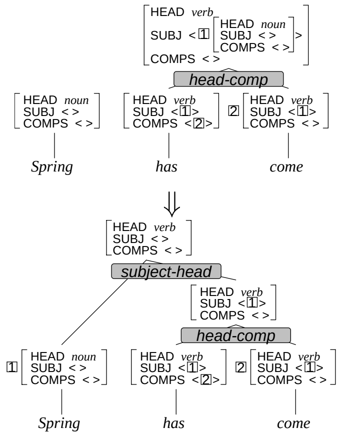

Figure 1: HPSG parsing

structures, an HPSG grammar is formulated as follows.

Definition 1 (HPSG grammar) *An HPSG grammar is a tuple, $G = \langle L, R \rangle$, where*

- $L = \{l = \langle w, F \rangle | w \in \mathcal{W}, F \in \mathcal{F}\}$ *is a set of lexical entries, and*
- R *is a set of grammar rules, i.e., $r \in R$ is a partial function: $\mathcal{F} \times \mathcal{F} \rightarrow \mathcal{F}$.*

Given a sentence, an HPSG grammar computes a set of phrasal signs, i.e., feature structures, as a result of parsing.

Existing studies (Abney, 1997; Johnson et al., 1999; Miyao et al., 2003) define a probability of feature structure F with *log-linear model* or *maximum entropy model* as follows.

Definition 2 (Probabilistic HPSG) *Probability $p(F|\mathbf{w})$ of feature structure F assigned to given sentence $\mathbf{w}$ is defined as follows.*

$$p(F|\mathbf{w}) = \frac{1}{Z_{\mathbf{w}}} \exp\left(\sum_i \lambda_i \sigma(s_i, F)\right)$$

$$Z_{\mathbf{w}} = \sum_{F'} \exp\left(\sum_i \lambda_i \sigma(s_i, F')\right)$$

where λ_i is a model parameter, s_i is a fragment of a feature structure, and $\sigma(s_i, F)$ is a function to return the number of appearances of feature structure fragment s_i in F.

Intuitively, a probability is defined as a normalized product of the weights $\exp(\lambda_i)$ when fragment s_i appears in the feature structure F. The probability represents syntactic/semantic preference expressed in a feature structure. For example, in the probabilistic model of predicate-argument structures (Miyao et al., 2003), s_i was designed to be each predicate-argument relation.

4 CKY parsing for probabilistic HPSG

The CKY algorithm, which is essentially a bottom-up parser, is a natural choice for non-probabilistic HPSG parsers. Because the large portion of constraints is expressed in lexical entries in HPSG, bottom-up parsers can utilize those constraints to reduce the search space in early stages of parsing.

For PCFG, extending the CKY algorithm to output the Viterbi parse is straightforward (Ney, 1991; Jurafsky and Martin, 2000). The parser can efficiently calculate the Viterbi parse by taking the maximum of the probabilities of the same nonterminal symbol in each cell.

To achieve such efficiency in CKY parsing for probabilistic HPSG, we need a function to reduce multiple feature structures that are equivalent in terms of resulting FOMs to a single feature structure.

4.1 Equivalence class function

First, we define the FOM of feature structure F as

$$\Delta(F) = \sum_i \lambda_i \sigma(s_i, F),$$

and $\delta(r, F_1, F_2)$ as

$$\delta(r, F_1, F_2) = \Delta(F) - \Delta(F_1) - \Delta(F_2),$$

$$F = r(F_1, F_2),$$

where r is a grammar rule, and F_1 and F_2 are the daughters of F.

We define *equivalence class function* η as a function $\eta : \mathcal{F} \rightarrow \mathcal{F}$ that satisfies the following conditions for all F_1, F_2 and r.

Condition 1:

$$^{\exists}F.(F = \eta(r(F_1, F_2)) \Leftrightarrow F = \eta(r(\eta(F_1), \eta(F_2))))$$

Condition 2:

$$^{\exists}d.(d = \delta(r, F_1, F_2) \Leftrightarrow d = \delta(r, \eta(F_1), \eta(F_2))$$

The first condition guarantees that the parsing with reduced feature structures will not overgenerate nor undergenerate. The second assures that the FOM computed with reduced feature structures is equivalent to the original one.

If we can construct the equivalence class function for a given grammar, we can calculate the FOM of the mother as

$$\Delta(F) = \delta(r, \eta(F_1), \eta(F_2)) + \Delta(F_1) + \Delta(F_2).$$

This equation indicates that we can employ dynamic programming on reduced feature structures in CKY parsing. The remaining issue is how to construct the equivalence class function, which depends on the grammar and the FOM model.

Designing the equivalence class function for local features is straightforward. The function can simply discard the portions of the feature structure for the local features which have no influence on the succeeding parsing processes and FOMs.

As for non-local features, Miyao et al. (2003) implicitly defined an equivalence class function for predicate argument structures. In their definition, instantiated arguments were removed from predicate argument structures in each step of parsing, because instantiated arguments were no more required for further processing in their probabilistic model. Owing to this function, predicate argument structures could be represented with a compact packed structure, which allowed the tractable estimation of model parameters.

In this work we follow their approach and define the equivalence class function as the one that removes instantiated arguments from the feature structure. In other words, the equivalence class function effectively delays the evaluation of probabilities of non-local dependencies until they are fixed.

The computational cost of this parsing framework heavily depends on how efficiently we can compute $\delta(r, \eta(F_1), \eta(F_2))$ in each step. The probabilistic model used in this paper define each non-local feature as a pair of predicate-argument relations. Since the probabilistic model is linear,

```
function CKY(words, grammar)
{
  # diagonal
  for i = 1 to num_words
    foreach F_u ∈ {F|⟨w_i, F⟩ ∈ L}
      α = log(P(F_u → w_i))
      F'_u = η(F_u)
      if (α > π[i, i][F'_u]) then
        π[i, i][F'_u] = α

  # the rest of the matrix
  for j = 2 to num_words
    for i = 1 to num_words-j+1
      for k = 1 to j-1
        foreach F_s ∈ π[i, k]
        foreach F_t ∈ π[i + k, j − k]
        if F = r(F_s, F_t) has succeeded
          α = Δ(F_s) + Δ(F_t) + δ(r, F_s, F_t)
          F' = η(F)
          if (α > π[i, j][F']) then
              π[i, j][F'] = α
}
```

Figure 2: Pseudocode of CKY parsing for probabilistic HPSG.

we can compute $\delta(r, \eta(F_1), \eta(F_2))$ for non-local features by just summing up the weights of newly instantiated arguments.

4.2 CKY parsing algorithm

Figure 2 shows a CKY parsing algorithm for Probabilistic HPSG. The algorithm is almost identical to the CKY for PCFG. Note that a feature structure is reduced by the equivalence class function just before whose FOM is compared with that of the corresponding feature structure which is already in the chart.

After filling the chart, the parser can obtain the Viterbi parse by starting from the cell $\pi[1, \textit{num\_words}]$ and traversing the chart along the links to the daughters. It should be noted that no additional computation is required to construct the best predicate-argument relations for the sentence, which is often quite expensive in full-parsing strategies. All the computations required to build the entire Viterbi parse including argument structures are done during filling the chart.

4.3 Beam thresholding

The CKY algorithm with the equivalent class function enables us to employ various pruning techniques for efficient parsing.

Beam thresholding is a simple and effective

γ	phrasal category
pos	part-of-speech of the head word
lc	lexical entry of the head word
r	schema
δ	distance between the head words of daughters
pc	existence of punctuation between daughters
ρ	argument position in a predicate-argument structure

Table 1: Notations in the description of features

technique to prune edges during parsing. In each cell of the chart, the method keeps only a portion of the edges which have higher FOMs compared to the other edges in the same cell.

In this work, we tried two selection schemes for deciding the edges to be kept in each cell.

- Thresholding by number of edges

  Each cell keeps the top n edges according to their FOMs.

- Thresholding by beam width

  Each cell keeps the edges whose FOM is greater than $\alpha_{max} - w$, where α_{max} is the highest FOM among the edges in the cell.

5 Experiments

5.1 Grammar and probabilistic models

An HPSG grammar used in the experiments was extracted from the Penn Treebank (Marcus et al., 1994) by the method of Miyao et al. (Miyao et al., 2004). The grammar acquired from the Penn Treebank Sections 02–21 (39,598 sentences) consisted of 826 lexical entry templates for 10,809 words. In average, 2.62 lexical entries were assigned to a word.

In order to investigate the effect of beam-thresholding with a different kind of probabilistic models, we prepared two probabilistic models. One was a model using only syntactic features (the *syntax* model). Syntactic features capture the characteristics of each branching in an HPSG derivation. Formally, a syntactic feature represents the occurrence of the branching $\langle\langle\gamma_h, pos_h, lc_h\rangle, \langle\gamma_n, pos_n, lc_n\rangle, r, \delta, pc\rangle$, where γ_h and γ_n are for head/non-head daughters and other notations are represented in Table 1. The other was a model using semantic features in addition to the syntactic features (the *full* model). Semantic features capture the characteristics of each

	# Features	Data size	Estimation time
syntax	54,345	8.66 GB	76 min
full	56,440	13.15 GB	110 min

Table 2: Space/computational costs of model estimation

predicate-argument relation, which is formally represented with $\langle\langle pos_h, lc_h\rangle, \langle pos_n, lc_n\rangle, \rho, \delta\rangle$. Note that the syntax model concerns tree structures of HPSG derivations, while the full model treats predicate-argument structures that can include re-entrant structures. Comparing the behaviors of these models, we can investigate the effect of the equivalent class function in the model involving re-entrant structures. We implemented the equivalence class function similar to the existing study on the probabilistic modeling of predicate-argument structures (Miyao et al., 2003)

Table 2 shows the space/computational costs of model estimation. The parameters of the two models were estimated by using the limited-memory BFGS algorithm (Nocedal, 1980) with a Gaussian distribution as a prior probability distribution for smoothing (Chen and Rosenfeld, 1999). All of the experiments were performed on servers with 1.26-GHz Pentium-III CPU and 4-GB memory.

The sentences in section 22 were used for evaluation. We parsed all the sentences that had less than 40 words.

5.2 Beam thresholding schemes

We first performed experiments to evaluate the beam thresholding schemes presented in section 4.3. We evaluated the accuracy of predicate-argument relations of the output Viterbi parses and the total time required for parsing the sentences. Each predicate-argument relation was counted as correct if the type of predicate, the argument position, the head word, and the argument word were all correct.

Figure 3 to 4 show the precision (or recall) and the total time for parsing the entire section using the syntax model. It is somewhat interesting that the beam thresholding strategy does not degrade the precision of the output. The main reason is that the parser does not output any predicate-argument relations for a sentence if the parsing has failed by

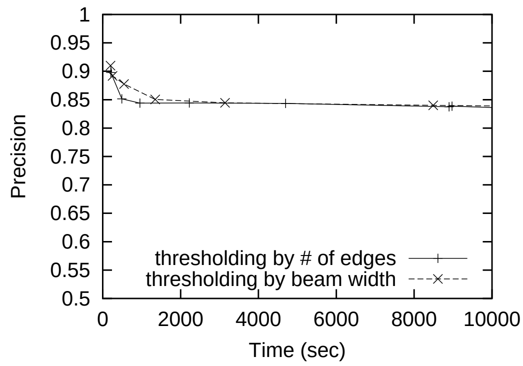

Figure 3: Precision versus time in the syntax model

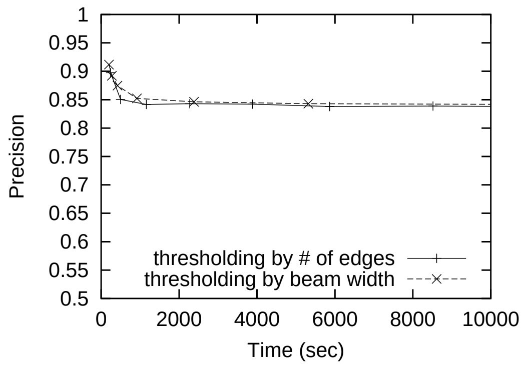

Figure 5: Precision versus time in the full model

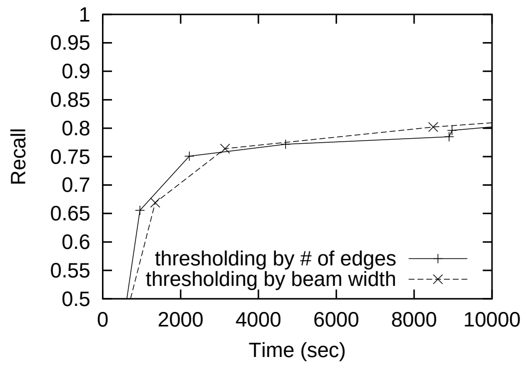

Figure 4: Recall versus time in the syntax model

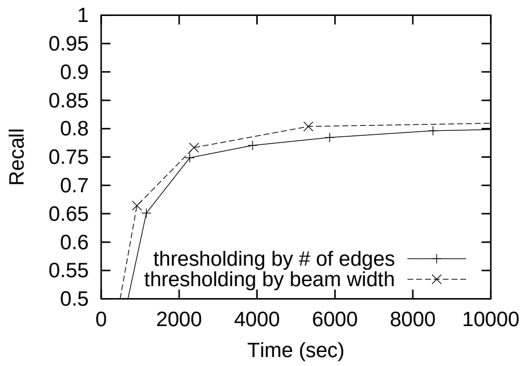

Figure 6: Recall versus time in the full model

pruning the edges that are necessary to construct a complete parse of the sentence.

On the other hand, the recall significantly deteriorated as we tightened the threshold. The two thresholding schemes showed comparable performance.

The results in the full model are shown in Figure 5 to 6. They show similar trends to those in the syntax model. With regard to the recall, thersholding by beam width gives slightly better results than that by number of edges.

5.3 Comparing with the baseline parser

One simple way to obtain the best parse of a sentence is to first parse the sentence without using the probabilistic model, then search the best parse among the parse results. We have implemented this baseline parser with the CKY parsing algorithm for non-probabilistic HPSG combined with a CFG filtering technique (Torisawa et al., 2000), which is one of the most efficient techniques to perform full parsing.

We compared our parsing algorithm proposed in this paper with the baseline parser with regard to the speed and accuracy of parsing. Table 3 show the results. The parameter setting of each beam thresholding scheme was chosen in such a way that the time of parsing is around 10% of the baseline parser. We also performed parsing by combining the two thresholding schemes. The parser with the combined thresholding achieved significant speedup compared with the baseline parser. In this parameter setting, the loss of recall was quite small (around 5%). We have also conducted the same experiments with the full model (Table 4). Again, the parser with the combined thresholding showed considerable speedup.

In these experiments, the baseline parser spent about one-third of the time on full parsing. This gives the upper-bound of the speedup of any parsing algorithms including (Geman and Johnson,

2002) that require full parsing as the first phase. Our parsing algorithm still offers considerable speedup when compared to the upper bounds. The rest of the time was spent on selecting the best parse.

6 Conclusion

This paper presented a framework for efficient parsing with probabilistic HPSG with the equivalence class function. The proposed framework enables us to perform dynamic programming on partial parses and compute their probabilities during parsing. It therefore does not require full parsing. With this function, the search space can be significantly reduced by various pruning techniques including beam search and best-first search strategies.

We have built a parser using this framework and a beam thresholding technique. Experimental results on the Penn Treebank corpus show that the parser can achieve significant speedup with slight loss of recall compared to the conventional parser with CFG filtering.

References

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

Joan Bresnan. 1982. *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA.

Sharon A. Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24(2):275–298.

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

Mahesh V. Chitrao and Ralph Grishman. 1990. Edge-based best-first chart parsing. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 263–266.

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

Stuart Geman and Mark Johnson. 2002. Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *Proceedings of the 40st Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pages 279–286.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of ACL '99*, pages 535–541.

Dainiel Jurafsky and James H. Martin. 2000. *Speech and Language Processing*. Prentice Hall.

Walter Kasper, Hans-Ulrich Krieger, Jörg Spilker, and Hans Weber. 1996. From word hypotheses to logical form: An efficient interleaved approach. In *Proceedings of Natural Language Processing and Speech Technology. Results of the 3rd KONVENS Conference*, pages 77–88.

Martin Kay. 1970. Head driven parsing. In *Proceedings of Workshop on Parsing Technologies*, pages 139–152.

B. Kiefer, H.-U. Krieger, J. Carroll, and R. Malouf. 1999. A bag of useful techniques for efficient and robust parsing. In *Proc. of ACL-1999*, pages 473–480, June.

Dan Klein and Christopher D. Manning. 2003. A* parsing: Fast exact viterbi parse selection. In *Proceedings of HLT-NAACL'03*.

R. Malouf, J. Carroll, and A. Copestake. 2000. Efficient feature structure operations without compilation. *Journal of Natural Language Engineering*, 6(1):29–46.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Y. Matsumoto, H. Tanaka, H. Hirakawa, H. Miyoshi, and H. Yasukawa. 1983. BUP: A bottom up parser embedded in Prolog. *New Generation Computing*, 1(2).

John Maxwell and Ron Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–589.

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

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2004. Corpus-oriented grammar development for acquiring a Head-driven Phrase Structure Grammar from the Penn Treebank. In *Proceedings of IJCNLP-04*.

Parser	Precision	Recall	Time per Sentence (sec)
Baseline	83.8%	82.5%	43.75
Beam thresholding (n = 14)	83.8%	79.6%	5.82
Beam thresholding (w = 6.0)	84.0%	80.1%	5.50
Combined (n = 14, w = 6.0)	84.3%	77.2%	1.69

Table 3: Comparing with the baseline parser in the syntax model

Parser	Precision	Recall	Time per Sentence (sec)
Baseline	84.0%	82.7%	40.61
Beam thresholding (n = 14)	83.9%	78.4%	5.52
Beam thresholding (w = 6.0)	84.3%	80.4%	3.46
Combined (n = 14, w = 6.0)	84.4%	77.5%	2.19

Table 4: Comparing with the baseline parser in the full model

M.-J. Nederhof. 2000. Practical experiments with regular approximation of context-free languages. *Computational Linguistics*, 26(1).

H. Ney. 1991. Dynamic programming parsing for context-free grammars in continuous speech recognition. *IEEE Transactions on Signal Processing*, 39(2):336–340.

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

Gerald Penn and Cosmin Munteanu. 2003. A tabulation-based parsing method that reduces copying. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*.

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

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

D.J. Rosenkrantz and P.M. Lewis II. 1970. Deterministic left corner parsing. In *IEEE Conference Record of the 11th Annual Symposium on Switching and Automata Theory*, pages 139–152.

Yves Shabes, Anne Abeillè, and Aravind K. Joshi. 1988. Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In *Proceedings of 12th COLING*, pages 578–583.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press.

M. Tomita. 1986. *Efficient Parsing for Natural Language*. Kluwer Academic Publishers.

Kentaro Torisawa, Kenji Nishida, Yusuke Miyao, and Jun'ichi Tsujii. 2000. An HPSG parser with CFG filtering. *Journal of Natural Language Engineering*, 6(1):63–80.

Gertjan van Noord. 1997. An efficient implementation of the head-corner parser. *Computational Linguistics*, 23(3).

Wolfgang Wahlster. 1993. Verbmobil: Translation of face-to-face dialogs. In *Proceedings of the MT Summit IV*, pages 127–135.