Towards efficient probabilistic HPSG parsing: integrating semantic and syntactic preference to guide the parsing

Yoshimasa Tsuruoka†‡ **Yusuke Miyao**‡ **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,tsujii}@is.s.u-tokyo.ac.jp

Abstract

We present a framework for efficient 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 using the integrated FOMs. This paper presents a CKY algorithm with this function and experimental results of beam thresholding. We also present an iterative CKY parsing for HPSG, which should further speed up parsing in runtime.

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 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 another major concern for probabilistic HPSG. One way to obtain the Viterbi (highest probability) parse given a probabilistic model is to first perform parsing without using probabilities, and then select the highest probability parse by looking at every parse result. Existing techniques to improve parsing efficiency of unification-based grammars should be useful in the first phase (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). However, in general we must explore an exponential search space for selecting the best parse among the resulting parses of the first phase, and an exhaustive search is often impractical or impossible.

This paper presents a unified framework of 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 using the integrated FOMs.

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

We also present an extension of the CKY algorithm which further reduces the number of edges using an upper bound of the FOM of the outside Viterbi parse on each edge. This algorithm should speed up parsing in runtime, without losing the optimality of the output parse.

2 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

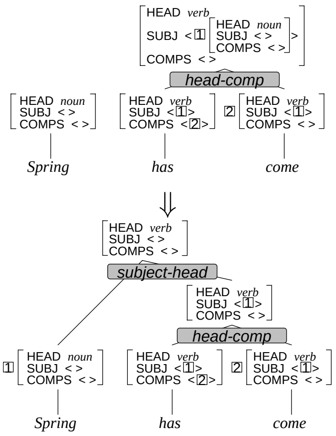

Figure 1: HPSG parsing

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 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: $\mathcal{F} \times \mathcal{F} \to \mathcal{F}$.*

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.

3 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.

3.1 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.

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. We formulate their approach as an equivalence class function and state necessary conditions for the function.

3.2 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.

```
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.

4 Iterative CKY parsing

When the grammar is large, we often need to further reduce the computational cost by pruning edges produced during parsing.

One way for pruning edges is to use a beam search strategy, in which only the best n parses are tracked. Roark (2001) and Ratnaparkhi (1999) applied this technique to PCFG parsing. Another way to reduce the number of edges produced is to use best-first or A* search strategies (Charniak et al., 1998; Caraballo and Charniak, 1998; Klein and Manning, 2003). Best-first strategies produce edges that are most likely to lead to a successful parse at each moment. While beam search and best-first strategies do not guarantee to output the Viterbi parse, A* search always outputs the Viterbi parse.

The iterative CKY algorithm, which is motivated by A* parsing for PCFG, is an extension of the CKY algorithm. It repetitively performs CKY parsing with a threshold until the successful parse is found. The threshold allows the parser to prune edges during parsing, which results in efficient parsing. By pruning edges using an upper bound of the outside Viterbi FOM, the parser guarantees to output the Viterbi parse.

```
function iterativeCKY(words, grammar, step)
{
  threshold = $t_0$
  until CKY'() returns success
  {
    CKY'(words, grammar, threshold)
    threshold = threshold - step
  }
}

function CKY'(words, grammar, threshold)
{
  # diagonal
  for $i = 1$ to num_words
    foreach $F_u \in \{F | \langle w_i, F \rangle \in L\}$
      $\alpha = log(P(F_u \rightarrow w_i))$
      $F'_u = \eta(F_u)$
      if ($\alpha > \pi[i,j][F'_u]$)
        $\beta = outside(F'_u, i, i)$
        if ($\alpha + \beta \geq threshold$)
          $\pi[i,i][F'_u] = \alpha$

  # 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 $F_s \in \pi[i,k]$
        foreach $F_t \in \pi[i+k, j-k]$
        if $F = r(F_s, F_t)$ has succeeded
          $\alpha = \Delta(F_s) + \Delta(F_t) + \delta(r, F_s, F_t)$
          $F' = \eta(F)$
          if ($\alpha > \pi[i,j][F']$) then
            $\beta = outside(F', i, j)$
            if ($\alpha + \beta \geq threshold$)
              $\pi[i,j][F'] = \alpha$
}
```

Figure 3: iterative CKY parsing algorithm for probabilistic HPSG.

4.1 Algorithm

Figure 3 shows the algorithm of iterative CKY parsing for Probabilistic HPSG.

The main function **iterativeCKY(...)** repetitively calls the function **CKY'(...)** giving a threshold to it until it returns a successful parse. The threshold starts with some initial value and is decreased by a predefined *step* at each iteration.

The function **CKY'(...)** prunes the edges that do not satisfy the condition

$$\alpha_e + \beta_e \geq \text{threshold}, \qquad (1)$$

where α_e is the inside Viterbi FOM[1] of the edge, which is calculated in a bottom-up manner, and β_e is the upper bound of the outside Viterbi FOM of the edge. Therefore, the sum of α_e and β_e is the highest FOM among those of all possible successful parse trees that contain the edge. In other words, the sum is the most optimistic estimate.

After filling the dynamic programming table, the algorithm checks whether the parsing has successfully finished by looking at the upper right corner of the matrix. If it is, it returns *success*; otherwise, it returns *failure*.

It should be noted that if this function returns *success*, the obtained parse is optimal because pruning is done on the most optimistic estimate. The algorithm prunes only the edges that would not lead to a successful parse within the given threshold. The edges needed to build the optimal parse are never pruned.

When this function returns *failure*, it is called again with a relaxed threshold.

It is important that the value of β_e can be computed off-line. Therefore, what we need to do in runtime is just retrieve the information from the precomputed table. The runtime overhead is very small.

One simple way to compute β_e is calculating the FOM of an edge by allowing the outside edges to be ANY lexical entries. For PCFG, Klein (2003) presented an efficient algorithm to compute outside Viterbi probabilities in a recursive manner.

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

This section provides the experimental results of beam thresholding on the CKY algorithm presented in Section 3.

5.1 Grammar and probabilistic models

The HPSG grammar used in the experiments was constructed from the Penn Treebank (Marcus et al.,

[1] We use the term "inside (or outside) Viterbi FOM" to refer to the FOM of the Viterbi parse inside (or outside) an edge. Do not confuse it with the *inside (or outside) probability* which refers to the "sum" of the probabilities of all possible parses inside (or outside) an edge.

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

# Features	Data size	Estimation time
56,440	13.15 GB	110 min

Table 2: Space/computational costs of model estimation

1994) by the method of Miyao et al. (Miyao et al., 2004). The grammar acquired from 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.

The probabilistic model has two types of features: syntactic features and semantic features. The 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 semantic features capture the characteristics of each 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 model treats predicate-argument structures that can include 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 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 200 sentences that had less than 40 words and could be strictly covered by the grammar (i.e., the grammar included all lexical entries to output the correct parse for the sentence).

Parser	Precision	Recall	Time (sec)
Baseline	86.9%	87.2%	26.64
Beam search	87.5%	83.4%	0.85

Table 3: Comparing with the baseline parser

5.2 Beam thresholding

Beam thresholding is conducted on each cell in the CKY chart. The parser keeps only top ten edges according to their FOMs. Additionally, it discards the edges whose FOMs are lower than the top FOM of the cell by 6.0 (in log-probability).

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. For comparison, 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).

Table 3 shows the results. The parser with beam thresholding achieved about 30 times speedup compared with the baseline parser. The loss of recall was 3.8%.

6 Conclusion

This paper presented a framework for efficient parsing with probabilistic HPSG with the equivalence class function. The proposed framework enables us to compute the FOMs of partial parse results during parsing. With this function, the search space can be significantly reduced by various pruning techniques including beam search and best-first search strategies.

We also described an iterative CKY parsing algorithm for probabilistic HPSG, which can reduce the number of edges produced during parsing without losing the optimality of the output parse.

References

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

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

Eugene Charniak, Sharon Goldwater, and Mark Johnson. 1998. Edge-based best-first chart parsing. In *Proceedings of the Sixth Workshop on Very Large Corpora*.

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

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.

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 the HLT-NAACL*, pages 119–126.

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*.

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.

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).