Deterministic shift-reduce parsing for unification-based grammars by using default unification

Takashi Ninomiya
Information Technology Center
University of Tokyo, Japan
ninomi@r.dl.itc.u-tokyo.ac.jp

Takuya Matsuzaki
Department of Computer Science
University of Tokyo, Japan
matuzaki@is.s.u-tokyo.ac.jp

Nobuyuki Shimizu
Information Technology Center
University of Tokyo, Japan
shimizu@r.dl.itc.u-tokyo.ac.jp

Hiroshi Nakagawa
Information Technology Center
University of Tokyo, Japan
nakagawa@dl.itc.u-tokyo.ac.jp

Abstract

Many parsing techniques including parameter estimation assume the use of a packed parse forest for efficient and accurate parsing. However, they have several inherent problems deriving from the restriction of locality in the packed parse forest. Deterministic parsing is one of solutions that can achieve simple and fast parsing without the mechanisms of the packed parse forest by accurately choosing search paths. We propose (i) deterministic shift-reduce parsing for unification-based grammars, and (ii) best-first shift-reduce parsing with beam thresholding for unification-based grammars. Deterministic parsing cannot simply be applied to unification-based grammar parsing, which often fails because of its hard constraints. Therefore, it is developed by using default unification, which almost always succeeds in unification by overwriting inconsistent constraints in grammars.

1 Introduction

Over the last few decades, probabilistic unification-based grammar parsing has been investigated intensively. Previous studies (Abney, 1997; Johnson et al., 1999; Kaplan et al., 2004; Malouf and van Noord, 2004; Miyao and Tsujii, 2005; Riezler et al., 2000) defined a probabilistic model of unification-based grammars, including head-driven phrase structure grammar (HPSG), lexical functional grammar (LFG) and combinatory categorial grammar (CCG), as a maximum entropy model (Berger et al., 1996). Geman and Johnson (Geman and Johnson, 2002) and Miyao and Tsujii (Miyao and Tsujii, 2002) proposed a feature forest, which is a dynamic programming algorithm for estimating the probabilities of all possible parse candidates. A feature forest can estimate the model parameters without unpacking the parse forest, i.e., the chart and its edges.

Feature forests have been used successfully for probabilistic HPSG and CCG (Clark and Curran, 2004b; Miyao and Tsujii, 2005), and its parsing is empirically known to be fast and accurate, especially with supertagging (Clark and Curran, 2004a; Ninomiya et al., 2007; Ninomiya et al., 2006). Both estimation and parsing with the packed parse forest, however, have several inherent problems deriving from the restriction of locality. First, feature functions can be defined only for local structures, which limit the parser's performance. This is because parsers segment parse trees into constituents and factor equivalent constituents into a single constituent (edge) in a chart to avoid the same calculation. This also means that the semantic structures must be segmented. This is a crucial problem when we think of designing semantic structures other than predicate argument structures, e.g., synchronous grammars for machine translation. The size of the constituents will be exponential if the semantic structures are not segmented. Lastly, we need delayed evaluation for evaluating feature functions. The application of feature functions must be delayed until all the values in the segmented constituents are instantiated. This is because values in parse trees can propagate anywhere throughout the parse tree by unification. For example, values may propagate from the root node to terminal nodes, and the final form of the terminal nodes is unknown until the parser finishes constructing the whole parse tree. Consequently, the design of grammars, semantic structures, and feature functions becomes complex. To solve the problem of locality, several approaches, such as reranking (Charniak and Johnson, 2005), shift-reduce parsing (Yamada and Matsumoto, 2003), search optimization learning (Daumé and Marcu, 2005) and sampling methods (Malouf and van Noord, 2004; Nakagawa, 2007), were studied.

In this paper, we investigate shift-reduce parsing approach for unification-based grammars without the mechanisms of the packed parse forest. Shift-reduce parsing for CFG and dependency parsing have recently been studied (Nivre and Scholz, 2004; Ratnaparkhi, 1997; Sagae and Lavie, 2005, 2006; Yamada and Matsumoto, 2003), through approaches based essentially on deterministic parsing. These techniques, however, cannot simply be applied to unification-based grammar parsing because it can fail as a result of its hard constraints in the grammar. Therefore, in this study, we propose deterministic parsing for unification-based grammars by using default unification, which almost always succeeds in unification by overwriting inconsistent constraints in the grammars. We further pursue best-first shift-reduce parsing for unification-based grammars.

Sections 2 and 3 explain unification-based grammars and default unification, respectively. Shift-reduce parsing for unification-based grammars is presented in Section 4. Section 5 discusses our experiments, and Section 6 concludes the paper.

2 Unification-based grammars

A unification-based grammar is defined as a pair consisting of a set of lexical entries and a set of phrase-structure rules. The lexical entries express word-specific characteristics, while the phrase-structure rules describe constructions of constituents in parse trees. Both the phrase-structure rules and the lexical entries are represented by feature structures (Carpenter, 1992), and constraints in the grammar are forced by unification. Among the phrase-structure rules, a binary rule is a partial function: $\mathcal{F} \times \mathcal{F} \rightarrow \mathcal{F}$,

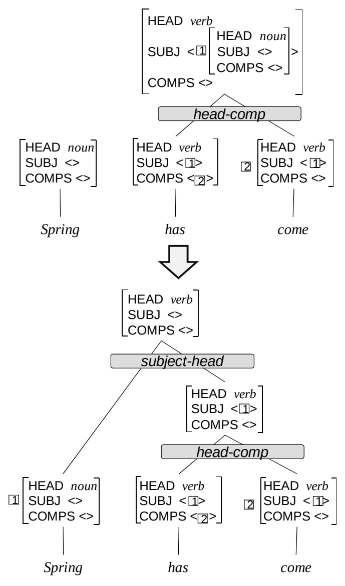
HEAD verb
SUBJ < 1 HEAD noun SUBJ <> COMPS <> >
COMPS <>
head-comp
HEAD noun SUBJ <> COMPS <>
HEAD verb SUBJ < 1 > COMPS < 2 >
2 HEAD verb SUBJ < 1 > COMPS <>
Spring
has
come
HEAD verb SUBJ <> COMPS <>
subject-head
HEAD verb SUBJ < 1 > COMPS <>
head-comp
1 HEAD noun SUBJ <> COMPS <>
HEAD verb SUBJ < 1 > COMPS < 2 >
2 HEAD verb SUBJ < 1 > COMPS <>
Spring
has
come

Figure 1: Example of HPSG parsing.

where $\mathcal{F}$ is the set of all possible feature structures. The binary rule takes two partial parse trees as daughters and returns a larger partial parse tree that consists of the daughters and their mother. A unary rule is a partial function: $\mathcal{F} \rightarrow \mathcal{F}$, which corresponds to a unary branch.

In the experiments, we used an HPSG (Pollard and Sag, 1994), which is one of the sophisticated unification-based grammars in linguistics. Generally, an HPSG has a small number of phrase-structure rules and a large number of lexical entries. Figure 1 shows an example of HPSG parsing of the sentence, “Spring has come.” The upper part of the figure shows a partial parse tree for “has come,” which is obtained by unifying each of the lexical entries for “has” and “come” with a daughter feature structure of the head-complement rule. Larger partial parse trees are obtained by repeatedly applying phrase-structure rules to lexical/phrasal partial parse trees. Finally, the parse result is output as a parse tree that dominates the sentence.

3 Default unification

Default unification was originally investigated in a series of studies of lexical semantics, in order to deal with default inheritance in a lexicon. It is also desirable, however, for robust processing, because (i) it almost always succeeds and (ii) a feature structure is relaxed such that the amount of information is maximized (Ninomiya et al., 2002). In our experiments, we tested a simplified version of Copestake’s default unification. Before explaining it, we first explain Carpenter’s two definitions of default unification (Carpenter, 1993).

(Credulous Default Unification)

$$F \overset{<}{\sqcup}_c G = \left\{ F \sqcup G' \middle| \begin{array}{c} G' \sqsubseteq G \text{ is maximal such} \\ \text{that } F \sqcup G' \text{is defined} \end{array} \right\}$$

(Skeptical Default Unification)

$$F \overset{<}{\sqcup}_s G = \sqcap (F \overset{<}{\sqcup}_c G)$$

F is called a strict feature structure, whose information must not be lost, and G is called a default feature structure, whose information can be lost but as little as possible so that F and G can be unified.

Credulous default unification is greedy, in that it tries to maximize the amount of information from the default feature structure, but it results in a set of feature structures. Skeptical default unification simply generalizes the set of feature structures resulting from credulous default unification. Skeptical default unification thus leads to a unique result so that the default information that can be found in every result of credulous default unification remains. The following is an example of skeptical default unification:

$$[\mathrm{F}{:}\,\mathbf{a}] \overset{<}{\sqcup}_{\mathrm{s}} \begin{bmatrix} \mathrm{F}{:}\, \boxed{1}\mathbf{b} \\ \mathrm{G}{:}\, \boxed{1} \\ \mathrm{H}{:}\, \mathbf{c} \end{bmatrix} = \sqcap \left\{ \begin{bmatrix} \mathrm{F}{:}\, \mathbf{a} \\ \mathrm{G}{:}\, \mathbf{b} \\ \mathrm{H}{:}\, \mathbf{c} \end{bmatrix}, \begin{bmatrix} \mathrm{F}{:}\, \boxed{1}\mathbf{a} \\ \mathrm{G}{:}\, \boxed{1} \\ \mathrm{H}{:}\, \mathbf{c} \end{bmatrix} \right\} = \begin{bmatrix} \mathrm{F}{:}\, \mathbf{a} \\ \mathrm{G}{:}\, \bot \\ \mathrm{H}{:}\, \mathbf{c} \end{bmatrix}.$$

Copestake mentioned that the problem with Carpenter's default unification is its time complexity (Copestake, 1993). Carpenter's default unification takes exponential time to find the optimal answer, because it requires checking the unifiability of the power set of constraints in a default feature structure. Copestake thus proposed another definition of default unification, as follows. Let $PV(G)$ be a function that returns a set of path values in G, and let $PE(G)$ be a function that returns a set of path equations, i.e., information about structure sharing in G.

(Copestake's default unification)

$$F \overset{<}{\sqcup}_a G = H \sqcup \bigsqcup \left\{ F \middle| \begin{array}{c} F \in PV(G) \text{and there is no } F' \in PV(G) \\ \text{such that } H \sqcup F' \text{is defined and} \\ H \sqcup F \sqcup F' \text{is not defined} \end{array} \right\},$$

where $H = F \sqcup \bigsqcup PE(G)$.

Copestake's default unification works efficiently because all path equations in the default feature structure are unified with the strict feature structures, and because the unifiability of path values is checked one by one for each node in the result of unifying the path equations. The implementation is almost the same as that of normal unification, but each node of a feature structure has a set of values marked as "strict" or "default." When types are involved, however, it is not easy to find unifiable path values in the default feature structure. Therefore, we implemented a more simply typed version of Corpestake's default unification.

```
procedure forced_unification(p, q)
  queue := {⟨p, q⟩};
  while( queue is not empty )
    ⟨p, q⟩ := shift(queue);
    p := deref(p); q := deref(q);
    if p ≠ q
      θ(p) ≔ θ(p) ∪ θ(q);
      θ(q) ≔ ptr(p);
      forall f ∈ feat(p)∪ feat(q)
        if f ∈ feat(p) ∧ f ∈ feat(q)
          queue := queue ∪ ⟨δ(f, p), δ(f, q)⟩;
        if f ∉ feat(p) ∧ f ∈ feat(q)
          δ(f, p) ≔ δ(f, q);
procedure mark(p, m)
  p := deref(p);
  if p has not been visited
    θ(p) := {⟨θ(p), m⟩};
    forall f ∈ feat(p)
      mark(δ(f, p), m);
procedure collapse_defaults(p)
  p := deref(p);
  if p has not been visited
    ts := ⊥; td := ⊥;
    forall ⟨t, strict⟩ ∈ θ(p)
      ts := ts ⊔ t;
    forall ⟨t, default⟩ ∈ θ(p)
      td := td ⊔ t;
    if ts is not defined
      return false;
    if ts ⊔ td is defined
      θ(p) := ts ⊔ td;
    else
      θ(p) := ts;
    forall f ∈ feat(p)
      collapse_defaults(δ(f, p));
procedure default_unification(p, q)
  mark(p, strict);
  mark(q, default);
  forced_unification(p, q);
  collapse_defaults(p);
```

θ(p) is (i) a single type, (ii) a pointer, or (iii) a set of pairs of types and markers in the feature structure node p.
A marker indicates that the types in a feature structure node originally belong to the strict feature structures or the default feature structures.
A pointer indicates that the node has been unified with other nodes and it points the unified node. A function deref traverses pointer nodes until it reaches to non-pointer node.
δ(f, p) returns a feature structure node which is reached by following a feature f from p.

Figure 2: Algorithm for the simply typed version of Corpestake's default unification.

Figure 2 shows the algorithm by which we implemented the simply typed version. First, each node is marked as "strict" if it belongs to a strict feature structure and as "default" otherwise. The marked strict and default feature structures

Common features: Sw(i), Sp(i), Shw(i), Shp(i), Snw(i), Snp(i), Ssy(i), Shsy(i), Snsy(i), wi-1, wi,wi+1, pi-2, pi-1, pi, pi+1, pi+2, pi+3
Binary reduce features: d, c, spl, syl, hwl, hpl, hll, spr, syr, hwr, hpr, hlr
Unary reduce features: sy, hw, hp, hl

Sw(i) … head word of i-th item from the top of the stack
Sp(i) … head POS of i-th item from the top of the stack
Shw(i) … head word of the head daughter of i-th item from the top of the stack
Shp(i) … head POS of the head daughter of i-th item from the top of the stack
Snw(i) … head word of the non-head daughter of i-th item from the top of the stack
Snp(i) … head POS of the non-head daughter of i-th item from the top of the stack
Ssy(i) … symbol of phrase category of the i-th item from the top of the stack
Shsy(i) … symbol of phrase category of the head daughter of the i-th item from the top of the stack
Snsy(i) … symbol of phrase category of the non-head daughter of the i-th item from the top of the stack
d … distance between head words of daughters
c … whether a comma exists between daughters and/or inside daughter phrases
sp … the number of words dominated by the phrase
sy … symbol of phrase category
hw … head word
hp … head POS
hl … head lexical entry

Figure 3: Feature templates.

Shift Features
[Sw(0)] [Sw(1)] [Sw(2)] [Sw(3)] [Sp(0)] [Sp(1)] [Sp(2)] [Sp(3)] [Shw(0)] [Shw(1)] [Shp(0)] [Shp(1)] [Snw(0)] [Snw(1)] [Snp(0)] [Snp(1)] [Ssy(0)] [Ssy(1)] [Shsy(0)] [Shsy(1)] [Snsy(0)] [Snsy(1)] [d] [wi-1] [wi] [wi+1] [pi-2] [pi-1] [pi] [pi+1] [pi+2] [pi+3] [wi-1, wi] [wi, wi+1] [pi-1, wi] [pi, wi] [pi+1, wi] [pi, pi+1, pi+2, pi+3] [pi-2, pi-1, pi] [pi-1, pi, pi+1] [pi, pi+1, pi+2] [pi-2, pi-1] [pi-1, pi] [pi, pi+1] [pi+1, pi+2]

Binary Reduce Features
[Sw(0)] [Sw(1)] [Sw(2)] [Sw(3)] [Sp(0)] [Sp(1)] [Sp(2)] [Sp(3)] [Shw(0)] [Shw(1)] [Shp(0)] [Shp(1)] [Snw(0)] [Snw(1)] [Snp(0)] [Snp(1)] [Ssy(0)] [Ssy(1)] [Shsy(0)] [Shsy(1)] [Snsy(0)] [Snsy(1)] [d] [wi-1] [wi] [wi+1] [pi-2] [pi-1] [pi] [pi+1] [pi+2] [pi+3] [d,c,hw,hp,hl] [d,c,hw,hp] [d, c, hw, hl] [d, c, sy, hw] [c, sp, hw, hp, hl] [c, sp, hw, hp] [c, sp, hw,hl] [c, sp, sy, hw] [d, c, hp, hl] [d, c, hp] [d, c, hl] [d, c, sy] [c, sp, hp, hl] [c, sp, hp] [c, sp, hl] [c, sp, sy]

Unary Reduce Features
[Sw(0)] [Sw(1)] [Sw(2)] [Sw(3)] [Sp(0)] [Sp(1)] [Sp(2)] [Sp(3)] [Shw(0)] [Shw(1)] [Shp(0)] [Shp(1)] [Snw(0)] [Snw(1)] [Snp(0)] [Snp(1)] [Ssy(0)] [Ssy(1)] [Shsy(0)] [Shsy(1)] [Snsy(0)] [Snsy(1)] [d] [wi-1] [wi] [wi+1] [pi-2] [pi-1] [pi] [pi+1] [pi+2] [pi+3] [hw, hp, hl] [hw, hp] [hw, hl] [sy, hw] [hp, hl] [hp] [hl] [sy]

Figure 4: Combinations of feature templates.

are unified, whereas the types in the feature structure nodes are not unified but merged as a set of types. Then, all types marked as "strict" are unified into one type for each node. If this fails, the default unification also returns unification failure as its result. Finally, each node is assigned a single type, which is the result of type unification for all types marked as both "default" and "strict" if it succeeds or all types marked only as "strict" otherwise.

4 Shift-reduce parsing for unification-based grammars

Non-deterministic shift-reduce parsing for unification-based grammars has been studied by Briscoe and Carroll (Briscoe and Carroll, 1993). Their algorithm works non-deterministically with the mechanism of the packed parse forest, and hence it has the problem of locality in the packed parse forest. This section explains our shift-reduce parsing algorithms, which are based on deterministic shift-reduce CFG parsing (Sagae and Lavie, 2005) and best-first shift-reduce CFG parsing (Sagae and Lavie, 2006). Sagae's parser selects the most probable shift/reduce actions and non-terminal symbols without assuming explicit CFG rules. Therefore, his parser can proceed deterministically without failure. However, in the case of unification-based grammars, a deterministic parser can fail as a result of its hard constraints in the grammar. We propose two new shift-reduce parsing approaches for unification-based grammars: deterministic shift-reduce parsing and shift-reduce parsing by backtracking and beam search. The major difference between our algorithm and Sagae's algorithm is that we use default unification. First, we explain the deterministic shift-reduce parsing algorithm, and then we explain the shift-reduce parsing with backtracking and beam search.

4.1 Deterministic shift-reduce parsing for unification-based grammars

The deterministic shift-reduce parsing algorithm for unification-based grammars mainly comprises two data structures: a stack S, and a queue W. Items in S are partial parse trees, including a lexical entry and a parse tree that dominates the whole input sentence. Items in W are words and POSs in the input sentence. The algorithm defines two types of parser actions, shift and reduce, as follows.

- Shift: A shift action removes the first item (a word and a POS) from W. Then, one lexical entry is selected from among the candidate lexical entries for the item. Finally, the selected lexical entry is put on the top of the stack.

- Binary Reduce: A binary reduce action removes two items from the top of the stack. Then, partial parse trees are derived by applying binary rules to the first removed item and the second removed item as a right daughter and left daughter, respectively. Among the candidate partial parse trees, one is selected and put on the top of the stack.
- Unary Reduce: A unary reduce action removes one item from the top of the stack. Then, partial parse trees are derived by applying unary rules to the removed item. Among the candidate partial parse trees, one is selected and put on the top of the stack.

Parsing fails if there is no candidate for selection (i.e., a dead end). Parsing is considered successfully finished when W is empty and S has only one item which satisfies the sentential condition: the category is verb and the subcategorization frame is empty. Parsing is considered a non-sentential success when W is empty and S has only one item but it does not satisfy the sentential condition.

In our experiments, we used a maximum entropy classifier to choose the parser's action. Figure 3 lists the feature templates for the classifier, and Figure 4 lists the combinations of feature templates. Many of these features were taken from those listed in (Ninomiya et al., 2007), (Miyao and Tsujii, 2005) and (Sagae and Lavie, 2005), including global features defined over the information in the stack, which cannot be used in parsing with the packed parse forest. The features for selecting shift actions are the same as the features used in the supertagger (Ninomiya et al., 2007). Our shift-reduce parsers can be regarded as an extension of the supertagger.

The deterministic parsing can fail because of its grammar's hard constraints. So, we use default unification, which almost always succeeds in unification. We assume that a head daughter (or, an important daughter) is determined for each binary rule in the unification-based grammar. Default unification is used in the binary rule application in the same way as used in Ninomiya's offline robust parsing (Ninomiya et al., 2002), in which a binary rule unified with the head daughter is the strict feature structure and the non-head daughter is the default feature structure, i.e., $(R \sqcup H) \overset{<}{\sqcup} NH$, where R is a binary rule, H is a head daughter and NH is a non-head daughter. In the experiments, we used the simply typed version of Copestake's default unification in the binary rule application[1]. Note that default unification was always used instead of normal unification in both training and evaluation in the case of the parsers using default unification. Although Copestake's default unification almost always succeeds, the binary rule application can fail if the binary rule cannot be unified with the head daughter, or inconsistency is caused by path equations in the default feature structures. If the rule application fails for all the binary rules, backtracking or beam search can be used for its recovery as explained in Section 4.2. In the experiments, we had no failure in the binary rule application with default unification.

4.2 Shift-reduce parsing by backtracking and beam-search

Another approach for recovering from the parsing failure is backtracking. When parsing fails or ends with non-sentential success, the parser's state goes back to some old state (backtracking), and it chooses the second best action and tries parsing again. The old state is selected so as to minimize the difference in the probabilities for selecting the best candidate and the second best candidate. We define a maximum number of backtracking steps while parsing a sentence. Backtracking repeats until parsing finishes with sentential success or reaches the maximum number of backtracking steps. If parsing fails to find a parse tree, the best continuous partial parse trees are output for evaluation.

From the viewpoint of search algorithms, parsing with backtracking is a sort of depth-first search algorithms. Another possibility is to use the best-first search algorithm. The best-first parser has a state priority queue, and each state consists of a tree stack and a word queue, which are the same stack and queue explained in the shift-reduce parsing algorithm. Parsing proceeds by applying shift-reduce actions to the best state in the state queue. First, the best state is re-

[1] We also implemented Ninomiya's default unification, which can weaken path equation constraints. In the preliminary experiments, we tested binary rule application given as $(R \sqcup H) \overset{<}{\sqcup} NH$ with Copestake's default unification, $(R \sqcup H) \overset{<}{\sqcup} NH$ with Ninomiya's default unification, and $(H \sqcup NH) \overset{<}{\sqcup} R$ with Ninomiya's default unification. However, there was no significant difference of F-score among these three methods. So, in the main experiments, we only tested $(R \sqcup H) \overset{<}{\sqcup} NH$ with Copestake's default unification because this method is simple and stable.

		Section 23 (Gold POS)								
		LP (%)	LR (%)	LF (%)	Avg. Time (ms)	# of backtrack	Avg. # of states	# of dead end	# of non-sentential success	# of sentential success
Previous studies	(Miyao and Tsujii, 2005)	87.26	86.50	86.88	604	-	-	-	-	-
	(Ninomiya et al., 2007)	89.78	89.28	89.53	234	-	-	-	-	-
Ours	det	76.45	82.00	79.13	122	0	-	867	35	1514
	det+du	87.78	87.45	87.61	256	0	-	0	117	2299
	back40	81.93	85.31	83.59	519	18986	-	386	23	2007
	back10 + du	87.79	87.46	87.62	267	574	-	0	45	2371
	beam(7.4)	86.17	87.77	86.96	510	-	226	369	30	2017
	beam(20.1)+du	88.67	88.79	88.48	457	-	205	0	16	2400
	beam(403.4)	89.98	89.92	89.95	10246	-	2822	71	14	2331

		Section 23 (Auto POS)								
		LP (%)	LR (%)	LF (%)	Avg. Time (ms)	# of backtrack	Avg. # of states	# of dead end	# of non sentential success	# of sentential success
Previous studies	(Miyao and Tsujii, 2005)	84.96	84.25	84.60	674	-	-	-	-	-
	(Ninomiya et al., 2007)	87.28	87.05	87.17	260	-	-	-	-	-
	(Matsuzaki et al., 2007)	86.93	86.47	86.70	30	-	-	-	-	-
	(Sagae et al., 2007)	88.50	88.00	88.20	-	-	-	-	-	-
Ours	det	74.13	80.02	76.96	127	0	-	909	31	1476
	det+du	85.93	85.72	85.82	252	0	-	0	124	2292
	back40	78.71	82.86	80.73	568	21068	-	438	27	1951
	back10 + du	85.96	85.75	85.85	270	589	-	0	46	2370
	beam(7.4)	83.84	85.82	84.82	544	-	234	421	33	1962
	beam(20.1)+du	86.59	86.36	86.48	550	-	222	0	21	2395
	beam(403.4)	87.70	87.86	87.78	16822	-	4553	89	16	2311

Table 1: Experimental results for Section 23.

moved from the state queue, and then shift-reduce actions are applied to the state. The newly generated states as results of the shift-reduce actions are put on the queue. This process repeats until it generates a state satisfying the sentential condition. We define the probability of a parsing state as the product of the probabilities of selecting actions that have been taken to reach the state. We regard the state probability as the objective function in the best-first search algorithm, i.e., the state with the highest probabilities is always chosen in the algorithm. However, the best-first algorithm with this objective function searches like the breadth-first search, and hence, parsing is very slow or cannot be processed in a reasonable time. So, we introduce beam thresholding to the best-first algorithm. The search space is pruned by only adding a new state to the state queue if its probability is greater than 1/b of the probability of the best state in the states that has had the same number of shift-reduce actions. In what follows, we call this algorithm beam search parsing.

In the experiments, we tested both backtracking and beam search with/without default unification. Note that, the beam search parsing for unification-based grammars is very slow compared to the shift-reduce CFG parsing with beam search. This is because we have to copy parse trees, which consist of a large feature structures, in every step of searching to keep many states on the state queue. In the case of backtracking, copying is not necessary.

5 Experiments

We evaluated the speed and accuracy of parsing with Enju 2.3β, an HPSG for English (Miyao and Tsujii, 2005). The lexicon for the grammar was extracted from Sections 02-21 of the Penn Treebank (39,832 sentences). The grammar consisted of 2,302 lexical entries for 11,187 words. Two probabilistic classifiers for selecting shift-reduce actions were trained using the same portion of the treebank. One is trained using normal unification, and the other is trained using default unification.

We measured the accuracy of the predicate argument relation output of the parser. A predicate-argument relation is defined as a tuple $\langle \sigma, w_h, a, w_a \rangle$, where σ is the predicate type (e.g.,

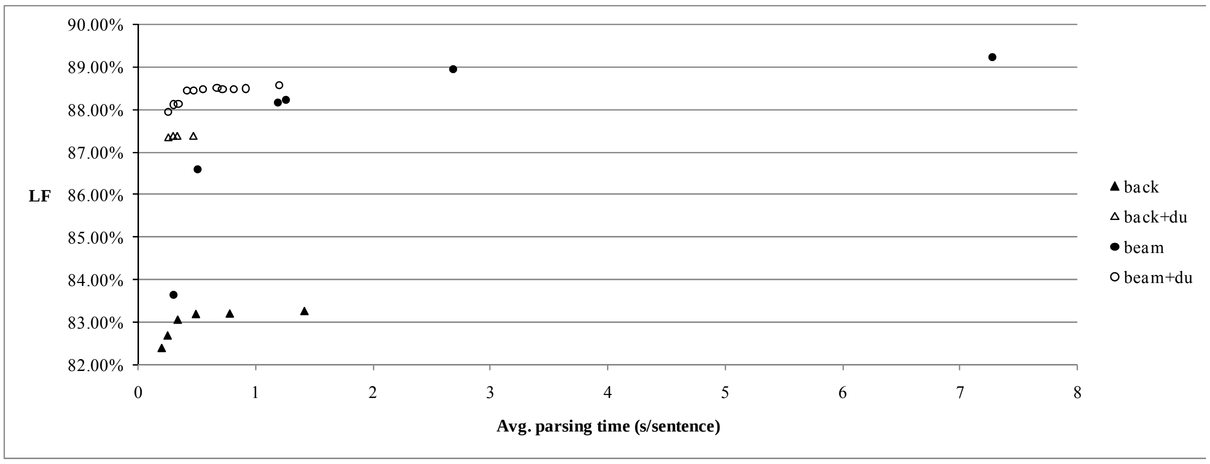

Figure 5: The relation between LF and the average parsing time (Section 22, Gold POS).

adjective, intransitive verb), w_h is the head word of the predicate, a is the argument label (MOD-ARG, ARG1, …, ARG4), and w_a is the head word of the argument. The labeled precision (LP) / labeled recall (LR) is the ratio of tuples correctly identified by the parser, and the labeled F-score (LF) is the harmonic mean of the LP and LR. This evaluation scheme was the same one used in previous evaluations of lexicalized grammars (Clark and Curran, 2004b; Hockenmaier, 2003; Miyao and Tsujii, 2005). The experiments were conducted on an Intel Xeon 5160 server with 3.0-GHz CPUs. Section 22 of the Penn Treebank was used as the development set, and the performance was evaluated using sentences of ≤ 100 words in Section 23. The LP, LR, and LF were evaluated for Section 23.

Table 1 lists the results of parsing for Section 23. In the table, "Avg. time" is the average parsing time for the tested sentences. "# of backtrack" is the total number of backtracking steps that occurred during parsing. "Avg. # of states" is the average number of states for the tested sentences. "# of dead end" is the number of sentences for which parsing failed. "# of non-sentential success" is the number of sentences for which parsing succeeded but did not generate a parse tree satisfying the sentential condition. "det" means the deterministic shift-reduce parsing proposed in this paper. "back$n$" means shift-reduce parsing with backtracking at most n times for each sentence. "du" indicates that default unification was used. "beam$b$" means best-first shift-reduce parsing with beam threshold b. The upper half of the table gives the results obtained using gold POSs, while the lower half gives the results obtained using an automatic POS tagger. The maximum number of backtracking steps and the beam threshold were determined by observing the performance for the development set (Section 22) such that the LF was maximized with a parsing time of less than 500 ms/sentence (except "beam(403.4)"). The performance of "beam(403.4)" was evaluated to see the limit of the performance of the beam-search parsing.

Deterministic parsing without default unification achieved accuracy with an LF of around 79.1% (Section 23, gold POS). With backtracking, the LF increased to 83.6%. Figure 5 shows the relation between LF and parsing time for the development set (Section 22, gold POS). As seen in the figure, the LF increased as the parsing time increased. The increase in LF for deterministic parsing without default unification, however, seems to have saturated around 83.3%. Table 1 also shows that deterministic parsing with default unification achieved higher accuracy, with an LF of around 87.6% (Section 23, gold POS), without backtracking. Default unification is effective: it ran faster and achieved higher accuracy than deterministic parsing with normal unification. The beam-search parsing without default unification achieved high accuracy, with an LF of around 87.0%, but is still worse than deterministic parsing with default unification. However, with default unification, it achieved the best performance, with an LF of around 88.5%, in the settings of parsing time less than 500ms/sentence for Section 22.

For comparison with previous studies using the packed parse forest, the performances of Miyao's parser, Ninomiya's parser, Matsuzaki's parser and Sagae's parser are also listed in Table 1. Miyao's parser is based on a probabilistic model estimated only by a feature forest. Ninomiya's parser is a mixture of the feature forest

Path	Strict type	Default type	Freq
SYNSEM:LOCAL:CAT:HEAD:MOD:	cons	nil	434
SYNSEM:LOCAL:CAT:HEAD:MOD:hd:CAT:HEAD:MOD:	cons	nil	237
SYNSEM:LOCAL:CAT:VAL:SUBJ:	nil	cons	231
SYNSEM:LOCAL:CAT:HEAD:MOD:hd:CAT:VAL:SUBJ:	nil	cons	125
SYNSEM:LOCAL:CAT:HEAD:	verb	noun	110
SYNSEM:LOCAL:CAT:VAL:SPR:hd:LOCAL:CAT:VAL:SPEC:hd:LOCAL:CAT:HEAD:MOD:	cons	nil	101
SYNSEM:LOCAL:CAT:HEAD:MOD:hd:CAT:VAL:SPR:hd:LOCAL:CAT:VAL:SPEC:hd:LOCAL:CAT:HEAD:MOD:	cons	nil	96
SYNSEM:LOCAL:CAT:HEAD:MOD:	nil	cons	92
SYNSEM:LOCAL:CAT:HEAD:MOD:hd:CAT:HEAD:	verb	noun	91
SYNSEM:LOCAL:CAT:VAL:SUBJ:	cons	nil	79
SYNSEM:LOCAL:CAT:HEAD:	noun	verbal	77
SYNSEM:LOCAL:CAT:HEAD:MOD:hd:CAT:HEAD:	noun	verbal	77
SYNSEM:LOCAL:CAT:HEAD:	nominal	verb	75
SYNSEM:LOCAL:CAT:VAL:CONJ:hd:LOCAL:CAT:HEAD:MOD:	cons	nil	74
SYNSEM:LOCAL:CAT:VAL:CONJ:tl:hd:LOCAL:CAT:HEAD:MOD:	cons	nil	69
SYNSEM:LOCAL:CAT:VAL:CONJ:tl:hd:LOCAL:CAT:VAL:SUBJ:	nil	cons	64
SYNSEM:LOCAL:CAT:VAL:CONJ:hd:LOCAL:CAT:VAL:SUBJ:	nil	cons	64
SYNSEM:LOCAL:CAT:VAL:COMPS:hd:LOCAL:CAT:HEAD:	nominal	verb	63
SYNSEM:LOCAL:CAT:HEAD:MOD:hd:CAT:VAL:SUBJ:	cons	nil	63
…	…	…	…
Total			10,598

Table 2: Path values overwritten by default unification in Section 22.

and an HPSG supertagger. Matsuzaki's parser uses an HPSG supertagger and CFG filtering. Sagae's parser is a hybrid parser with a shallow dependency parser. Though parsing without the packed parse forest is disadvantageous to the parsing with the packed parse forest in terms of search space complexity, our model achieved higher accuracy than Miyao's parser.

"beam(403.4)" in Table 1 and "beam" in Figure 5 show possibilities of beam-search parsing. "beam(403.4)" was very slow, but the accuracy was higher than any other parsers except Sagae's parser.

Table 2 shows the behaviors of default unification for "det+du." The table shows the 20 most frequent path values that were overwritten by default unification in Section 22. In most of the cases, the overwritten path values were in the selection features, i.e., subcategorization frames (COMPS:, SUBJ:, SPR:, CONJ:) and modifiee specification (MOD:). The column of 'Default type' indicates the default types which were overwritten by the strict types in the column of 'Strict type,' and the last column is the frequency of overwriting. 'cons' means a non-empty list, and 'nil' means an empty list. In most of the cases, modifiee and subcategorization frames were changed from empty to non-empty and vice versa. From the table, overwriting of head information was also observed, e.g., 'noun' was changed to 'verb.'

6 Conclusion and Future Work

We have presented shift-reduce parsing approach for unification-based grammars, based on deterministic shift-reduce parsing. First, we presented deterministic parsing for unification-based grammars. Deterministic parsing was difficult in the framework of unification-based grammar parsing, which often fails because of its hard constraints. We introduced default unification to avoid the parsing failure. Our experimental results have demonstrated the effectiveness of deterministic parsing with default unification. The experiments revealed that deterministic parsing with default unification achieved high accuracy, with a labeled F-score (LF) of 87.6% for Section 23 of the Penn Treebank with gold POSs. Second, we also presented the best-first parsing with beam search for unification-based grammars. The best-first parsing with beam search achieved the best accuracy, with an LF of 87.0%, in the settings without default unification. Default unification further increased LF from 87.0% to 88.5%. By widening the beam width, the best-first parsing achieved an LF of 90.0%.

References

Abney, Steven P. 1997. Stochastic Attribute-Value Grammars. Computational Linguistics, 23(4), 597-618.

Berger, Adam, Stephen Della Pietra, and Vincent Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. Computational Linguistics, 22(1), 39-71.

Briscoe, Ted and John Carroll. 1993. Generalized probabilistic LR-Parsing of natural language (corpora) with unification-based grammars. Computational Linguistics, 19(1), 25-59.

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

Carpenter, Bob. 1993. Skeptical and Credulous Default Unification with Applications to Templates and Inheritance. In Inheritance, Defaults, and the Lexicon. Cambridge: Cambridge University Press.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In proc. of ACL'05, pp. 173-180.

Clark, Stephen and James R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In proc. of COLING-04, pp. 282-288.

Clark, Stephen and James R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In proc. of ACL'04, pp. 104-111.

Copestake, Ann. 1993. Defaults in Lexical Representation. In Inheritance, Defaults, and the Lexicon. Cambridge: Cambridge University Press.

Daumé, Hal III and Daniel Marcu. 2005. Learning as Search Optimization: Approximate Large Margin Methods for Structured Prediction. In proc. of ICML 2005.

Geman, Stuart and Mark Johnson. 2002. Dynamic programming for parsing and estimation of stochastic unification-based grammars. In proc. of ACL'02, pp. 279-286.

Hockenmaier, Julia. 2003. Parsing with Generative Models of Predicate-Argument Structure. In proc. of ACL'03, pp. 359-366.

Johnson, Mark, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for Stochastic ``Unification-Based" Grammars. In proc. of ACL '99, pp. 535-541.

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

Malouf, Robert and Gertjan van Noord. 2004. Wide Coverage Parsing with Stochastic Attribute Value Grammars. In proc. of IJCNLP-04 Workshop ``Beyond Shallow Analyses".

Matsuzaki, Takuya, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Efficient HPSG Parsing with Supertagging and CFG-filtering. In proc. of IJCAI 2007, pp. 1671-1676.

Miyao, Yusuke and Jun'ichi Tsujii. 2002. Maximum Entropy Estimation for Feature Forests. In proc. of HLT 2002, pp. 292-297.

Miyao, Yusuke and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In proc. of ACL'05, pp. 83-90.

Nakagawa, Tetsuji. 2007. Multilingual dependency parsing using global features. In proc. of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 915-932.

Ninomiya, Takashi, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2007. A log-linear model with an n-gram reference distribution for accurate HPSG parsing. In proc. of IWPT 2007, pp. 60-68.

Ninomiya, Takashi, Takuya Matsuzaki, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. 2006. Extremely Lexicalized Models for Accurate and Fast HPSG Parsing. In proc. of EMNLP 2006, pp. 155-163.

Ninomiya, Takashi, Yusuke Miyao, and Jun'ichi Tsujii. 2002. Lenient Default Unification for Robust Processing within Unification Based Grammar Formalisms. In proc. of COLING 2002, pp. 744-750.

Nivre, Joakim and Mario Scholz. 2004. Deterministic dependency parsing of English text. In proc. of COLING 2004, pp. 64-70.

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

Ratnaparkhi, Adwait. 1997. A linear observed time statistical parser based on maximum entropy models. In proc. of EMNLP'97.

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 proc. of ACL'00, pp. 480-487.

Sagae, Kenji and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In proc. of IWPT 2005.

Sagae, Kenji and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In proc. of COLING/ACL on Main conference poster sessions, pp. 691-698.

Sagae, Kenji, Yusuke Miyao, and Jun'ichi Tsujii. 2007. HPSG parsing with shallow dependency constraints. In proc. of ACL 2007, pp. 624-631.

Yamada, Hiroyasu and Yuji Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In proc. of IWPT-2003.