Stochastically Evaluating the Validity of Partial Parse Trees in Incremental Parsing

Yoshihide Kato[1], Shigeki Matsubara[2] and Yasuyoshi Inagaki[3]
Graduate School of International Development, Nagoya University [1]
Information Technology Center, Nagoya University [2]
Furo-cho, Chikusa-ku, Nagoya, 464-8601 Japan
Faculty of Information Science and Technology, Aichi Prefectural University [3]
1522-3 Ibaragabasama, Kumabari, Nagakute-cho, Aichi-gun, 480-1198 Japan
yosihide@gsid.nagoya-u.ac.jp

Abstract

This paper proposes a method for evaluating the validity of partial parse trees constructed in incremental parsing. Our method is based on stochastic incremental parsing, and it incrementally evaluates the validity for each partial parse tree on a word-by-word basis. In our method, incremental parser returns partial parse trees at the point where the validity for the partial parse tree becomes greater than a threshold. Our technique is effective for improving the accuracy of incremental parsing.

1 Introduction

Real-time spoken language processing systems, such as simultaneous machine interpretation systems, are required to quickly respond to users' utterances. To fulfill the requirement, the system needs to understand spoken language at least incrementally (Allen et al., 2001; Inagaki and Matsubara, 1995; Milward and Cooper, 1994), that is, to analyze each input sentence from left to right and acquire the content.

Several incremental parsing methods have been proposed to date (Costa et al., 2001; Haddock, 1987; Matsubara et al., 1997; Milward, 1995; Roark, 2001). These methods construct candidate partial parse trees for initial fragments of the input sentence on a word-by-word basis. However, these methods contain local ambiguity problems that partial parse trees representing valid syntactic relations can not be determined without using information from the rest of the input sentence.

On the other hand, Marcus proposed a method of deterministically constructing valid partial parse trees by looking ahead several words (Marcus, 1980), while Kato et al. proposed an incremental parsing which delays the decision of valid partial parse trees (Kato et al., 2000). However, it is hard to say that these methods realize broad-coverage incremental parsing. The method in the literature (Marcus, 1980) uses lookahead rules, which are constructed by hand, but it is not clear whether broad coverage lookahead rules can be obtained. The incremental parsing in the literature (Kato et al., 2000), which is based on context free grammar, is infeasible to deal with large scale grammar, because the parser exhaustively searches all candidate partial parse trees in top-down fashion.

This paper proposes a probabilistic incremental parser which evaluates the validity of partial parse trees. Our method extracts a grammar from a treebank, and the incremental parsing uses a beam-search strategy so that it realizes broad-coverage parsing. To resolve local ambiguity, the parser incrementally evaluates the validity of partial parse trees on a word-by-word basis, and delays the decision of which partial parse trees should be returned, until the validity for the partial parse tree becomes greater than a threshold. Our technique is effective for improving the accuracy of incremental parsing.

This paper is organized as follows: The next section proposes a probabilistic incremental parser. Section 3 discusses the validity of partial parse tree constructed in incremental parsing. Section 4 proposes a method of incrementally evaluating the validity of partial parse tree. In section 5, we report an experimental evaluation of our method.

2 TAG-based Incremental Parsing

Our incremental parsing is based on tree adjoining grammar (TAG) (Joshi, 1985). This section proposes a TAG-based incremental parsing method.

2.1 TAG for Incremental Parsing

Firstly, we propose incremental-parsing-oriented TAG (ITAG). An ITAG comprises two sets of elementary trees just like TAG: initial trees and auxiliary trees. The difference between ITAG and TAG is the form of elementary trees. Every ITAG initial tree is *leftmost-expanded*. A tree is leftmost-expanded if it is of the following forms:

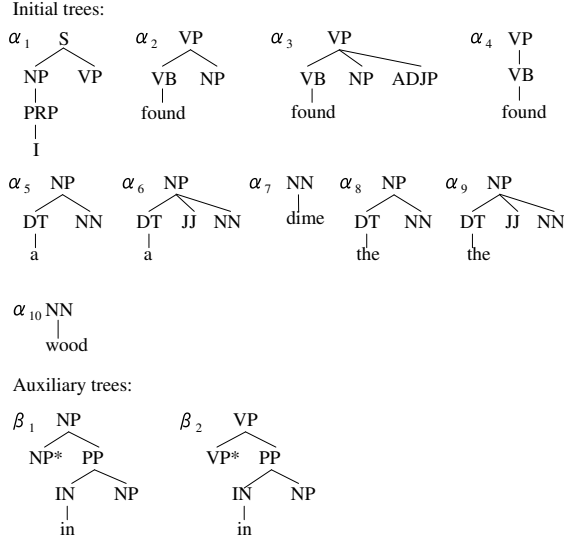1. $[t]_X$, where t is a terminal symbol and X is a nonterminal symbol.

Initial trees:

α_1 S — NP VP; NP — PRP — I

α_2 VP — VB NP; VB — found

α_3 VP — VB NP ADJP; VB — found

α_4 VP — VB; VB — found

α_5 NP — DT NN; DT — a

α_6 NP — DT JJ NN; DT — a

α_7 NN — dime

α_8 NP — DT NN; DT — the

α_9 NP — DT JJ NN; DT — the

α_{10} NN — wood

Auxiliary trees:

β_1 NP — NP* PP; PP — IN NP; IN — in

β_2 VP — VP* PP; PP — IN NP; IN — in

Figure 1: Examples of ITAG elementary trees

2. $[\sigma X_1 \cdots X_k]_X$, where σ is a leftmost expanded tree, $X_1, \ldots, X_k, X$ are nonterminal symbols.

On the other hand, every ITAG auxiliary tree is of the following form:

$$[X^* \sigma X_1 \cdots X_k]_X$$

where σ is a leftmost expanded tree and X, $X_1, \ldots, X_k$ are nonterminal symbols. X^* is called a foot node. Figure 1 shows examples of ITAG elementary trees.

These elemental trees can be combined by using two operations: *substitution* and *adjunction*.

substitution The substitution operation replaces a leftmost nonterminal leaf of a partial parse tree σ with an initial tree α having the same nonterminal symbol at its root. We write s_α for the operation of substituting α and $s_\alpha(\sigma)$ for the result of applying s_α to σ.

adjunction The adjunction operation splits a partial parse tree σ at a nonterminal node having no nonterminal leaf, and inserts an auxiliary tree β having the same nonterminal symbol at its root. We write a_β for the operation of adjoining β and $a_\beta(\sigma)$ for the result of applying a_β to σ.

The substitution operation is similar to rule expansion of top-down incremental parsing such as (Matsubara et al., 1997; Roark, 2001). Furthermore, by introducing the adjunction operation to incremental parsing, we can expect that local ambiguity of left-recursive structures is decreased (Lombardo and Sturt, 1997).

Our proposed incremental parsing is based on ITAG. When i-th word w_i is scanned, the parser combines elementary trees for w_i with partial parse trees for $w_1 \cdots w_{i-1}$ to construct the partial parse trees for $w_1 \cdots w_{i-1} w_i$.

As an example, let us consider incremental parsing of the following sentence by using ITAG shown in Figure 1:

$$\text{I found a dime in the wood.} \quad (1)$$

Table 1 shows the process of tree construction for the sentence (1). When the word "found" is scanned, partial parse trees #3, #4 and #5 are constructed by applying substitution operations to partial parse tree #2 for the initial fragment "I". When the word "in" is scanned, partial parse trees #12 and #13 are constructed by applying adjunction operations to partial parse tree #10 for the initial fragment "I found a dime". This example shows that the ITAG based incremental parsing is capable of constructing partial parse trees of initial fragments for every word input.

2.2 ITAG Extraction from Treebank

Here, we propose a method for extracting an ITAG from a treebank to realize broad-coverage incremental parsing. Our method decomposes parse trees in treebank to obtain ITAG elementary trees. The decomposition is as follows:

- for each node η_1 having no left-sibling, if the parent η_p has the same nonterminal symbol as η_1, split the parse tree at η_1 and η_p, and combine the upper tree and the lower tree. η_1 of intermediate tree is a foot node.

- for each node η_2 having only one left-sibling, if the parent η_p does not have the same nonterminal symbol as the left-sibling η_1 of η_2, split the parse tree at η_2.

- for the other node η in the parse tree, split the parse tree at η.

For example, The initial trees $\alpha_1, \alpha_2, \alpha_5, \alpha_7$ α_8 and α_{10} and the auxiliary tree β_2 are extracted from the parse tree #18 in Table 1.

Our proposed tree extraction is similar to the TAG extractions proposed in the literatures (Chen and Vijay-Shanker, 2000; Chiang, 2003; Xia, 1999). The main difference between these methods is the position of nodes at which parse trees are split. While the methods in the literatures (Chen and Vijay-Shanker, 2000; Chiang, 2003; Xia, 1999) utilize a head percolation rule to split the parse trees at complement nodes, our method splits the parse trees

Table 1: Incremental parsing process of "I found a dime in the wood."

word	#	partial parse tree
	1	s
I	2	$[[[I]_{prp}]_{np}vp]_s$
found	3	$[[[I]_{prp}]_{np}[[found]_{vb}np]_{vp}]_s$
	4	$[[[I]_{prp}]_{np}[[found]_{vb}np\;adjp]_{vp}]_s$
	5	$[[[I]_{prp}]_{np}[[found]_{vb}]_{vp}]_s$
a	6	$[[[I]_{prp}]_{np}[[found]_{vb}[[a]_{dt}nn]_{np}]_{vp}]_s$
	7	$[[[I]_{prp}]_{np}[[found]_{vb}[[a]_{dt}jj\;nn]_{np}]_{vp}]_s$
	8	$[[[I]_{prp}]_{np}[[found]_{vb}[[a]_{dt}nn]_{np}adjp]_{vp}]_s$
	9	$[[[I]_{prp}]_{np}[[found]_{vb}[[a]_{dt}jj\;nn]_{np}adjp]_{vp}]_s$
dime	10	$[[[I]_{prp}]_{np}[[found]_{vb}[[a]_{dt}[dime]_{nn}]_{np}]_{vp}]_s$
	11	$[[[I]_{prp}]_{np}[[found]_{vb}[[a]_{dt}[dime]_{nn}]_{np}adjp]_{vp}]_s$
in	12	$[[[I]_{prp}]_{np}[[[found]_{vb}[[a]_{dt}[dime]_{nn}]_{np}]_{vp}[[in]_{in}np]_{pp}]_{vp}]_s$
	13	$[[[I]_{prp}]_{np}[[found]_{vb}[[[a]_{dt}[dime]_{nn}]_{np}[[in]_{in}np]_{pp}]_{np}]_{vp}]_s$
the	14	$[[[I]_{prp}]_{np}[[[found]_{vb}[[a]_{dt}[dime]_{nn}]_{np}]_{vp}[[in]_{in}[[the]_{dt}nn]_{np}]_{pp}]_{vp}]_s$
	15	$[[[I]_{prp}]_{np}[[[found]_{vb}[[a]_{dt}[dime]_{nn}]_{np}]_{vp}[[in]_{in}[[the]_{dt}jj\;nn]_{np}]_{pp}]_{vp}]_s$
	16	$[[[I]_{prp}]_{np}[[found]_{vb}[[[a]_{dt}[dime]_{nn}]_{np}[[in]_{in}[[the]_{dt}nn]_{np}]_{pp}]_{np}]_{vp}]_s$
	17	$[[[I]_{prp}]_{np}[[found]_{vb}[[[a]_{dt}[dime]_{nn}]_{np}[[in]_{in}[[the]_{dt}jj\;nn]_{np}]_{pp}]_{np}]_{vp}]_s$
wood	18	$[[[I]_{prp}]_{np}[[[found]_{vb}[[a]_{dt}[dime]_{nn}]_{np}]_{vp}[[in]_{in}[[the]_{dt}[wood]_{nn}]_{np}]_{pp}]_{vp}]_s$
	19	$[[[I]_{prp}]_{np}[[found]_{vb}[[[a]_{dt}[dime]_{nn}]_{np}[[in]_{in}[[the]_{dt}[wood]_{nn}]_{np}]_{pp}]_{np}]_{vp}]_s$

at left recursive nodes and nodes having left-sibling. The elementary trees extracted by our method are of the forms described in section 2.1, and can be combined from left to right on a word-by-word basis. The property is suitable for incremental parsing. On the other hand, the elementary trees obtained by the method based on head information does not necessarily have this property [1].

2.3 Probabilistic ITAG

This section describes probabilistic ITAG (PITAG) which is utilized by evaluating partial parse trees in incremental parsing. PITAG assigns a probability to the event that an elementary tree is combined by substitution or adjunction with another tree.

We induce the probability by maximum likelihood estimation. Let α be an initial tree and X be the root symbol of α. The probability that α is substituted is calculated as follows:

$$P(s_\alpha) = \frac{C(s_\alpha)}{\sum_{\alpha' \in I(X)} C(s_{\alpha'})} \qquad (2)$$

where $C(s_\alpha)$ is the count of the number of times of applying substitution s_α in the treebank, and $I(X)$ is the set of initial trees whose root is labeled with X.

Let β be a auxiliary tree and X be the root symbol of β. The probability that β is adjoined is calculated as follows:

$$P(a_\beta) = \frac{C(a_\beta)}{C(X)} \qquad (3)$$

where $C(X)$ is the count of the number of occurrences of symbol X. The probability that adjunction is not applied is calculated as follows:

$$P(nil_X) = 1 - \sum_{\beta \in A(X)} P(a_\beta) \qquad (4)$$

where nil_X means that the adjunction is not applied to a node labeled with X, and $A(X)$ is the set of all auxiliary trees whose root is labeled X.

In this PITAG formalism, the probability that elementary trees are combined at each node depends only on the nonterminal symbol of that node [2].

The probability of a parse tree is calculated by the product of the probability of the operations which are used in construction of the parse tree. For example, the probability of each operation is given as shown in Table 2. The probability of the partial parse tree #12, which is constructed by using s_{α_1}, s_{α_2}, s_{α_5}, s_{α_7}, nil_{NP} and a_{β_2}, is $1 \times 0.7 \times 0.3 \times 0.5 \times 0.7 \times 0.7 = 0.05145$.

We write $P(\sigma)$ for the probability of a partial parse tree σ.

Table 2: Probability of operations

operation	probability
s_{α_1}	1.0
s_{α_2}	0.7
$s_{\alpha_7}, s_{\alpha_{10}}$	0.5
$s_{\alpha_5}, s_{\alpha_8}$	0.3
$s_{\alpha_4}, s_{\alpha_6}, s_{\alpha_9}$	0.2
s_{α_3}	0.1
a_{β_1}	0.3
a_{β_2}	0.7
nil_{NP}	0.7
nil_{VP}	0.3

Figure 2: Subsumption relation between partial parse trees

Figure 3: Valid partial parse trees

2.4 Parsing Strategies

In order to improve the efficiency of the parsing, we adapt two parsing strategies as follows:

- If two partial parse trees have the same sequence of nodes to which ITAG operations are applicable, then the lower probability tree can be safely discarded.

- The parser only keeps n-best partial parse trees.

3 Validity of Partial Parse Trees

This section gives some definitions about the validity of a partial parse tree. Before describing the validity of a partial parse tree, we define the subsumption relation between partial parse trees.

Definition 1 (subsumption relation) *Let σ and τ be partial parse trees. Then we write $\sigma \rhd \tau$, if $s_\alpha(\sigma) = \tau$, for some initial tree α or $a_\beta(\sigma) = \tau$, for some auxiliary tree β. Let $\rhd^*$ be the reflexive transitive closure of $\rhd$. We say that σ subsumes τ, if $\sigma \rhd^* \tau$.* □

That σ subsumes τ means that τ is the result of applying a substitution or an adjunction to σ. Figure 2 shows the subsumption relation between the partial parse trees constructed for the sentence (1).

If a partial parse tree for an initial fragment represents a syntactic relation correctly, the partial parse tree subsumes the correct parse tree for the input sentence. We say that such a partial parse tree is *valid*. The validity of a partial parse tree is defined as follows:

Definition 2 (valid partial parse tree) *Let σ be a partial parse tree and $w_1 \cdots w_n$ be an input sentence. We say that σ is valid for $w_1 \cdots w_n$ if σ subsumes the correct parse tree for $w_1 \cdots w_n$.* □
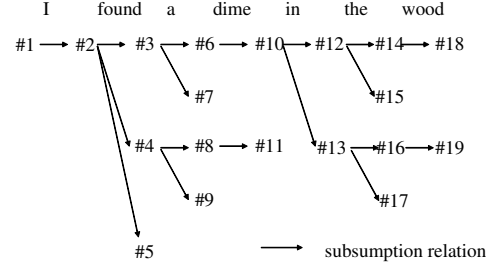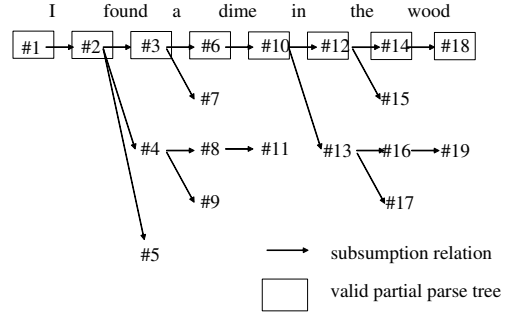
For example, assume that the #18 is correct parse tree for the sentence (1). Then partial parse tree #3 is valid for the sentence (1), because #3 $\rhd^*$ #18. On the other hand, partial parse tree #4 and #5 are not valid for (1). Figure 3 shows the valid partial parse trees for the sentence (1).

4 Evaluating the Validity of Partial Parse Tree

The validity of a partial parse tree for an initial fragment depends on the rest of the sentence. For example, the validity of the partial parse trees #3, #4 and #5 depends on the remaining input that follows the word "found." This means that the validity dynamically varies for every word input. We define a conditional validity of partial parse tree:

$$V(\sigma \mid w_1 \cdots w_j) = \frac{\sum_{\tau \in Sub(\sigma, w_1 \cdots w_j)} P(\tau)}{\sum_{\tau \in T(w_1 \cdots w_j)} P(\tau)} \quad (5)$$

where σ is a partial parse tree for an initial fragment $w_1 \cdots w_i (i \leq j)$, $T(w_1 \cdots w_j)$ is the set of constructed partial parse trees for the initial fragment $w_1 \cdots w_j$ and $Sub(\sigma, w_1 \cdots w_j)$ is the subset of $T(w_1 \cdots w_j)$ whose elements are subsumed by σ. The equation (5) represents the validity of σ on the condition $w_1 \cdots w_j$. σ is valid for input sentence if and only if some partial parse tree for $w_1 \cdots w_j$

subsumed by σ is valid. The equation 5 is the ratio of such partial parse trees to the constructed partial parse trees.

4.1 Output Partial Parse Trees

Kato et al. proposed a method of delaying the decision of which partial parse trees should be returned as the output, until the validity of partial parse trees are guaranteed (Kato et al., 2000). The idea of delaying the decision of the output is interesting. However, delaying the decision until the validity are guaranteed may cause the loss of incrementality of the parsing.

To solve the problem, in our method, the incremental parser returns high validity partial parse trees rather than validity guaranteed partial parse trees.

When the j-th word w_j is scanned, our incremental parser returns the following partial parse:

$$argmax_{\{\sigma:V(\sigma,w_1\cdots w_j)\geq\theta\}}l(\sigma) \qquad (6)$$

where θ is a threshold between $[0,1]$ and $l(\sigma)$ is the length of the initial fragment which is yielded by σ. The output partial parse tree is the one for the longest initial fragment in the partial parse trees whose validity are greater than a threshold θ.

4.2 An Example

Let us consider a parsing example for the sentence (1). We assume that the threshold $\theta = 0.8$.

Let us consider when the partial parse tree #3, which is valid for (1), is returned as output. When the word "found" is scanned, partial parse trees #3, #4 and #5 are constructed. That is, $T(\text{I found}) = \{\#3, \#4, \#5\}$. As shown in Figure 2, $Sub(\#3, \text{I found}) = \{\#3\}$. Furthermore, $P(\#3) = 0.7$, $P(\#4) = 0.1$ and $P(\#5) = 0.2$. Therefore, $Validity(\#3, \text{I found}) = 0.7/(0.7 + 0.1 + 0.2) = 0.7$. Because $Validity(\#3, \text{I found}) < \theta$, partial parse tree #3 is not returned as the output at this point. The parser only keeps #3 as a candidate partial parse tree.

When the next word "a" is scanned, partial parse trees #6, #7, #8 and #9 are constructed, where $P(\#6) = 0.21$, $P(\#7) = 0.14$, $P(\#8) = 0.03$ and $P(\#9) = 0.02$. $Sub(\#3, \text{I found a}) = \{\#6, \#7\}$. Therefore, $Validity(\#3, \text{I found a}) = (0.21 + 0.14)/(0.21+0.14+0.03+0.02) = 0.875$. Because $Validity(\#3, \text{I found a}) \geq \theta$, partial parse tree #3 is returned as the output.

Table 3 shows the output partial parse tree for every word input.

Our incremental parser delays the decision of the output as shown in this example.

Table 3: Output partial parse trees

input word	output partial parse tree
I	#2
found	
a	#3
dime	#10
in	#12
the	
wood	#18

5 Experimental Results

To evaluate the performance of our proposed method, we performed a parsing experiment. The parser was implemented in GNU Common Lisp on a Linux PC. In the experiment, the inputs of the incremental parser are POS sequences rather than word sequences. We used 47247 initial trees and 2931 auxiliary trees for the experiment. The elementary trees were extracted from the parse trees in sections 02-21 of the Wall Street Journal in Penn Treebank (Marcus et al., 1993), which is transformed by using parent-child annotation and left factoring (Roark and Johnson, 1999). We set the beam-width at 500.

The labeled precision and recall of the parsing are 80.8% and 78.5%, respectively for the section 23 in Penn Treebank. We used the set of sentences for which the outputs of the incremental parser are identical to the correct parse trees in the Penn Treebank. The number of these sentences is 451. The average length of these sentences is 13.5 words.

We measured the delays and the precisions for validity thresholds 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0.

We define the degree of delay as follows: Let $s = w_1 \cdots w_n$ be an input sentence and $o_j(s)$ be the partial parse tree that is the output when the j-th word w_j is scanned. We define the degree of delay when j-th word is scanned as follows:

$$D(j,s) = j - l(o_j(s)) \qquad (7)$$

We define maximum delay $D_{max}(s)$ and average delay $D_{ave}(s)$ as follows:

$$D_{max}(s) = \max_{1\leq j\leq n} D(j,s) \qquad (8)$$

$$D_{ave}(s) = \frac{1}{n}\sum_{j=1}^{n} D(j,s) \qquad (9)$$

The precision is defined as the percentage of valid partial parse trees in the output.

Moreover, we measured the precision of the parsing whose delay is always 0 and which returns the

Table 4: Precisions and delays

	precision(%)	D_{max}	D_{ave}
$\theta = 1.0$	100.0	11.9	6.4
$\theta = 0.9$	97.3	7.5	2.9
$\theta = 0.8$	95.4	6.4	2.2
$\theta = 0.7$	92.5	5.5	1.8
$\theta = 0.6$	88.4	4.5	1.3
$\theta = 0.5$	83.0	3.4	0.9
baseline	73.6	0.0	0.0
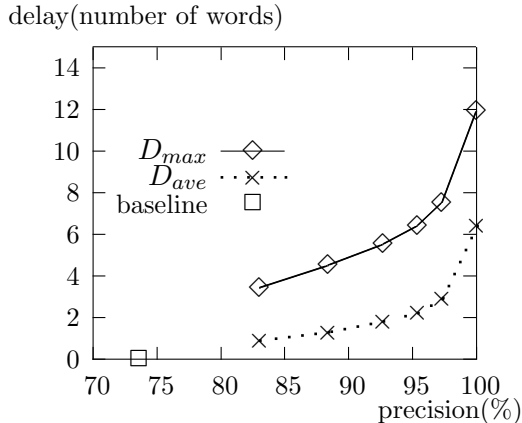
Figure 4: Relation between precision and delay

partial parse tree having highest probability. We call it the parsing baseline.

Table 4 shows the precisions and delays. Figure 4 illustrates the relation between the precisions and delays.

The experimental result demonstrates that there is a precision/delay trade-off. Our proposed method increases the precision in comparison with the baseline, while returning the output is delayed. When $\theta = 1$, it is guaranteed that the output partial parse trees are valid, that is, our method is similar to the method in the literature (Kato et al., 2000). In comparison with this case, our method when $\theta < 1$ dramatically decreases the delay.

Although the result does not necessarily demonstrates that our method is the best one, it achieves both high-accuracy and short-delay to a certain extent.

6 Concluding Remarks

In this paper, we have proposed a method of evaluating the validity that a partial parse tree constructed in incremental parsing becomes valid. The method is based on probabilistic incremental parsing. When a word is scanned, the method incrementally calculates the validity for each partial parse tree and re-

turns the partial parse tree whose validity is greater than a threshold. Our method delays the decision of which partial parse tree should be returned.

To evaluate the performance of our method, we conducted a parsing experiment using the Penn Treebank. The experimental result shows that our method improves the accuracy of incremental parsing.

The experiment demonstrated a precision/delay trade-off. To evaluate overall performance of incremental parsing, we would like to investigate a single measure into which delay and precision are combined.

Acknowledgement

References

J. Allen, G. Ferguson, and A. Stent. 2001. An Architecture for More Realistic Conversational Systems. In *Proceedings of International Conference of Intelligent User Interfaces*, pages 1–8.

J. Carroll and D. Weir. 2003. Encoding Frequency Information in Stochastic Parsing Models. In R. Bod, R. Scha, and K. Sima'an, editors, *Data-Oriented Parsing*, pages 43–60. CSLI Publications, Stanford.

J. Chen and K. Vijay-Shanker. 2000. Automated Extraction of TAGs from the Penn Treebank. In *Proceedings of the 6th International Workshop on Parsing Technologies*, pages 65–76.

D. Chiang. 2003. Statistical Parsing with an Automatically Extracted Tree Adjoining Grammar. In R. Bod, R. Scha, and K. Sima'an, editors, *Data-Oriented Parsing*, pages 299–316. CSLI Publications, Stanford.

F. Costa, V. Lombardo, P. Frasconi, and Soda G. 2001. Wide Coverage Incremental Parsing by Learning Attachment Preferences. In *Proceedings of the 7th Congress of the Italian Association for Artificial Intelligence*, pages 297–307.

N. J. Haddock. 1987. Incremental Interpretation and Combinatory Categorial Grammar. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 661–663.

Y. Inagaki and S. Matsubara. 1995. Models for Incremental Interpretation of Natural Language. In *Proceedings of the 2nd Symposium on Natural Language Processing*, pages 51–60.

A. K. Joshi. 1985. Tree Adjoining Grammar: How Much Context-Sensitivity is required to provide

reasonable structural descriptions? In D. R. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press, Cambridge.

Y. Kato, S. Matsubara, K. Toyama, and Y. Inagaki. 2000. Spoken Language Parsing based on Incremental Disambiguation. In *Proceedings of the 6th International Conference on Spoken Language Processing*, volume 2, pages 999–1002.

V. Lombardo and P. Sturt. 1997. Incremental Processing and Infinite Local Ambiguity. In *Proceedings of the 19th Annual Conference of the Cognitive Science Siciety*, pages 448–453.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):310–330.

M Marcus. 1980. *A Theory of Syntactic Recognition for Natural Language*. MIT Press, Cambrige, MA.

S. Matsubara, S. Asai, K. Toyama, and Y. Inagaki. 1997. Chart-based Parsing and Transfer in Incremental Spoken Language Translation. In *Proceedings of the 4th Natural Language Processing Pacific Rim Symposium*, pages 521–524.

D. Milward and R. Cooper. 1994. Incremental Interpretation: Applications, Theory, and Relationship to Dynamic Semantics. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 748–754.

D. Milward. 1995. Incremental Interpretation of Categorial Grammar. In *Proceedings of the 7th Conference of European Chapter of the Association for Computational Linguistics*, pages 119–126.

B. Roark and M. Johnson. 1999. Efficient Probabilistic Top-down and Left-corner Parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 421–428.

B. Roark. 2001. Probabilistic Top-Down Parsing and Language Modeling. *Computational Linguistics*, 27(2):249–276.

F. Xia. 1999. Extracting Tree Adjoining Grammars from Bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium*, pages 398–403.