Incremental Parsing with Monotonic Adjoining Operation

Yoshihide Kato and **Shigeki Matsubara**
Information Technology Center, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464-8601 Japan
{yosihide,matubara}@el.itc.nagoya-u.ac.jp

Abstract

This paper describes an incremental parser based on an adjoining operation. By using the operation, we can avoid the problem of infinite local ambiguity in incremental parsing. This paper further proposes a restricted version of the adjoining operation, which preserves lexical dependencies of partial parse trees. Our experimental results showed that the restriction enhances the accuracy of the incremental parsing.

1 Introduction

Incremental parser reads a sentence from left to right, and produces partial parse trees which span all words in each initial fragment of the sentence. Incremental parsing is useful to realize real-time spoken language processing systems, such as a simultaneous machine interpretation system, an automatic captioning system, or a spoken dialogue system (Allen et al., 2001).

Several incremental parsing methods have been proposed so far (Collins and Roark, 2004; Roark, 2001; Roark, 2004). In these methods, the parsers can produce the candidates of partial parse trees on a word-by-word basis. However, they suffer from the problem of infinite local ambiguity, i.e., they may produce an infinite number of candidates of partial parse trees. This problem is caused by the fact that partial parse trees can have arbitrarily nested left-recursive structures and there is no information to predict the depth of nesting.

To solve the problem, this paper proposes an incremental parsing method based on an adjoining operation. By using the operation, we can avoid the problem of infinite local ambiguity. This approach has been adopted by Lombardo and Sturt (1997) and Kato et al. (2004). However, this raises another problem that their adjoining operations cannot preserve lexical dependencies of partial parse trees. This paper proposes a restricted version of the adjoining operation which preserves lexical dependencies. Our experimental results showed that the restriction enhances the accuracy of the incremental parsing.

2 Incremental Parsing

This section gives a description of Collins and Roark's incremental parser (Collins and Roark, 2004) and discusses its problem.

Collins and Roark's parser uses a grammar defined by a 6-tuple $G = (V, T, S, \#, C, B)$. V is a set of nonterminal symbols. T is a set of terminal symbols. S is called a start symbol and $S \in V$. $\#$ is a special symbol to mark the end of a constituent. The rightmost child of every parent is labeled with this symbol. This is necessary to build a proper probabilistic parsing model. C is a set of *allowable chains*. An allowable chain is a sequence of nonterminal symbols followed by a terminal symbol. Each chain corresponds to a label sequence on a path from a node to its leftmost descendant leaf. B is a set of *allowable triples*. An allowable triple is a tuple $\langle X, Y, Z \rangle$ where $X, Y, Z \in V$. The triple specifies which nonterminal symbol Z is allowed to follow a nonterminal symbol Y under a parent X.

For each initial fragment of a sentence, Collins and Roark's incremental parser produces partial parse trees which span all words in the fragment.

Let us consider the parsing process as shown in Figure 1. For the first word "we", the parser produces the partial parse tree (a), if the allowable chain $\langle \text{S} \rightarrow \text{NP} \rightarrow \text{PRP} \rightarrow \text{we} \rangle$ exists in C. For other chains which start with S and end with "we", the parser produces partial parse trees by using the chains. For the next word, the parser attaches the chain $\langle \text{VP} \rightarrow \text{VBP} \rightarrow \text{describe} \rangle$ to the partial parse tree (a) [1]. The attachment is possible when the allowable triple $\langle$S, NP, VP$\rangle$ exists in B.

[1] More precisely, the chain is attached after attaching end-of-constituent # under the NP node.

Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, pages 41–44,
Suntec, Singapore, 4 August 2009.

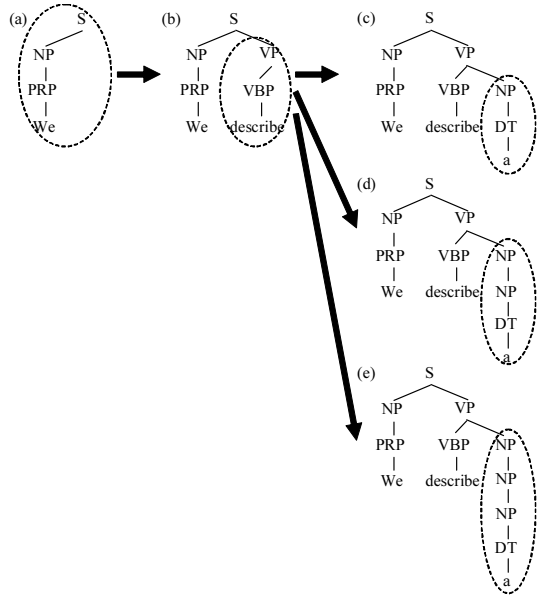

Figure 1: A process in incremental parsing

2.1 Infinite Local Ambiguity

Incremental parsing suffers from the problem of infinite local ambiguity. The ambiguity is caused by left-recursion. An infinite number of partial parse trees are produced, because we cannot predict the depth of left-recursive nesting.

Let us consider the fragment "We describe a." For this fragment, there exist several candidates of partial parse trees. Figure 1 shows candidates of partial parse trees. The partial parse tree (c) represents that the noun phrase which starts with "a" has no adjunct. The tree (d) represents that the noun phrase has an adjunct or is a conjunct of a coordinated noun phrase. The tree (e) represents that the noun phrase has an adjunct and the noun phrase with an adjunct is a conjunct of a coordinated noun phrase. The partial parse trees (d) and (e) are the instances of partial parse trees which have left-recursive structures. The major problem is that there is no information to determine the depth of left-recursive nesting at this point.

3 Incremental Parsing Method Based on Adjoining Operation

In order to avoid the problem of infinite local ambiguity, the previous works have adopted the following approaches: (1) a beam search strategy (Collins and Roark, 2004; Roark, 2001; Roark, 2004), (2) limiting the allowable chains to those actually observed in the treebank (Collins and Roark, 2004), and (3) transforming the parse trees with a selective left-corner transformation (Johnson and Roark, 2000) before inducing the allowable chains and allowable triples (Collins and Roark, 2004). The first and second approaches can prevent the parser from infinitely producing partial parse trees, but the parser has to produce partial parse trees as shown in Figure 1. The local ambiguity still remains. In the third approach, no left recursive structure exists in the transformed grammar, but the parse trees defined by the grammar are different from those defined by the original grammar. It is not clear if partial parse trees defined by the transformed grammar represent syntactic relations correctly.

As an approach to solve these problems, we introduce an adjoining operation to incremental parsing. Lombardo and Sturt (1997) and Kato et al. (2004) have already adopted this approach. However, their methods have another problem that their adjoining operations cannot preserve lexical dependencies of partial parse trees. To solve this problem, this section proposes a restricted version of the adjoining operation.

3.1 Adjoining Operation

An adjoining operation is used in Tree-Adjoining Grammar (Joshi, 1985). The operation inserts a tree into another tree. The inserted tree is called an *auxiliary tree*. Each auxiliary tree has a leaf called a *foot* which has the same nonterminal symbol as its root. An adjoining operation is defined as follows:

adjoining An adjoining operation splits a parse tree σ at a nonterminal node η and inserts an auxiliary tree β having the same nonterminal symbol as η, i.e., combines the upper tree of σ with the root of β and the lower tree of σ with the foot of β.

We write $a_{\eta,\beta}(\sigma)$ for the partial parse tree obtained by adjoining β to σ at η.

We use simplest auxiliary trees, which consist of a root and a foot.

As we have seen in Figure 1, Collins and Roark's parser produces partial parse trees such as (c), (d) and (e). On the other hand, by using the adjoining operation, our parser produces only the partial parse tree (c). When a left-recursive structure is required to parse the sentence, our parser adjoins it. In the example above, the parser adjoins the auxiliary tree $\langle \text{NP} \rightarrow \text{NP} \rangle$ to the partial parse tree (c) when the word "for" is read. This enables

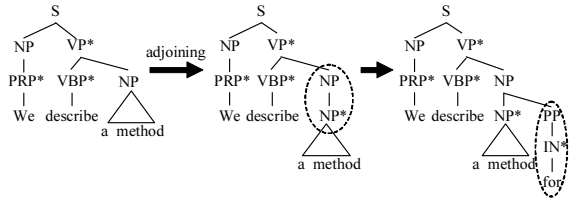

Figure 2: Adjoining operation

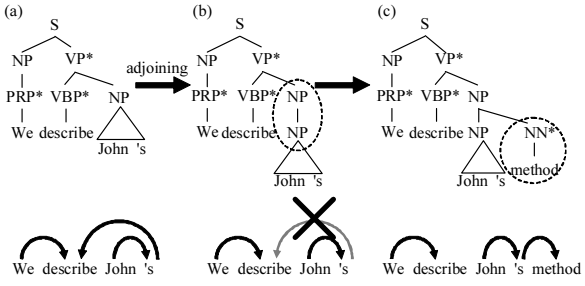

Figure 3: Non-monotonic adjoining operation

the parser to attach the allowable chain ⟨PP → IN → for⟩. The parsing process is shown in Figure 2.

3.2 Adjoining Operation and Monotonicity

By using the adjoining operation, we avoid the problem of infinite local ambiguity. However, the adjoining operation cannot preserve lexical dependencies of partial parse trees. Lexical dependency is a kind of relation between words, which represents head-modifier relation. We can map parse trees to sets of lexical dependencies by identifying the head-child of each constituent in the parse tree (Collins, 1999).

Let us consider the parsing process as shown in Figure 3. The partial parse tree (a) is a candidate for the initial fragment "We describe John 's". We mark each head-child with a special symbol $*$. We obtain three lexical dependencies ⟨We → describe⟩, ⟨John → 's⟩ and ⟨'s → describe⟩ from (a). When the parser reads the next word "method", it produces the partial parse tree (b) by adjoining the auxiliary tree ⟨NP → NP⟩. The partial parse tree (b) does not have ⟨'s → describe⟩. The dependency ⟨'s → describe⟩ is removed when the parser adjoins the auxiliary tree ⟨NP → NP⟩ to (a). This example demonstrates that the adjoining operation cannot preserve lexical dependencies of partial parse trees.

Now, we define the monotonicity of the adjoining operation. We say that adjoining an auxiliary tree β to a partial parse tree σ at a node η is *monotonic* when $dep(\sigma) \subseteq dep(a_{\eta,\beta}(\sigma))$ where dep is the mapping from a parse tree to a set of dependencies. An auxiliary tree β is monotonic if adjoining β to any partial parse tree is monotonic.

We want to exclude any non-monotonic auxiliary tree from the grammar. For this purpose, we restrict the form of auxiliary trees. In our framework, all auxiliary trees satisfy the following constraint:

- The foot of each auxiliary tree must be the head-child of its parent.

The auxiliary tree ⟨NP → NP*⟩ satisfies the constraint, while ⟨NP → NP⟩ does not.

3.3 Our Incremental Parser

Our incremental parser is based on a probabilistic parsing model which assigns a probability to each operation. The probability of a partial parse tree is defined by the product of the probabilities of the operations used in its construction. The probability of attaching an allowable chain c to a partial parse tree σ is approximated as follows:

$$\begin{aligned} P(c \mid \sigma) \quad = \quad & P_{root}(R \mid \mathbf{P}, \mathbf{L}, H, t_H, w_H, \mathbf{D}) \\ & \times P_{template}(c' \mid R, \mathbf{P}, \mathbf{L}, H) \\ & \times P_{word}(w \mid c', t_h, w_h) \end{aligned}$$

where R is the root label of c, c' is the sequence which is obtained by omitting the last element from c and w is the last element of c. The probability is conditioned on a limited context of σ. $\mathbf{P}$ is a set of the ancestor labels of R. $\mathbf{L}$ is a set of the left-sibling labels of R. H is the head label in $\mathbf{L}$. w_H and t_H are the head word and head tag of H, respectively. $\mathbf{D}$ is a set of distance features. w_h and t_h are the word and POS tag modified by w, respectively. The adjoining probability is approximated as follows:

$$P(\beta \mid \sigma) = P_{adjoining}(\beta \mid \mathbf{P}, \mathbf{L}, H, \mathbf{D})$$

where β is an auxiliary tree or a special symbol nil, the nil means that no auxiliary tree is adjoined. The limited contexts used in this model are similar to the previous methods (Collins and Roark, 2004; Roark, 2001; Roark, 2004).

To achieve efficient parsing, we use a beam search strategy like the previous methods (Collins and Roark, 2004; Roark, 2001; Roark, 2004). For each word position i, our parser has a priority queue H_i. Each queue H_i stores the only N-best

Table 1: Parsing results

	LR(%)	LP(%)	F(%)
Roark (2004)	86.4	86.8	86.6
Collins and Roark (2004)	86.5	86.8	86.7
No adjoining	86.3	86.8	86.6
Non-monotonic adjoining	86.1	87.1	86.6
Monotonic adjoining	87.2	87.7	87.4

partial parse trees. In addition, the parser discards the partial parse tree σ whose probability $P(\sigma)$ is less than the $P^*\gamma$ where P^* is the highest probability on the queue H_i and γ is a beam factor.

4 Experimental Evaluation

To evaluate the performance of our incremental parser, we conducted a parsing experiment. We implemented the following three types of incremental parsers to assess the influence of the adjoining operation and its monotonicity: (1) without adjoining operation, (2) with non-monotonic adjoining operation, and (3) with monotonic adjoining operation. The grammars were extracted from the parse trees in sections 02-21 of the Wall Street Journal in Penn Treebank. We identified the head-child in each constituent by using the head rule of Collins (Collins, 1999). The probabilistic models were built by using the maximum entropy method. We set the beam-width N to 300 and the beam factor γ to 10^{-11}.

We evaluated the parsing accuracy by using section 23. We measured labeled recall and labeled precision. Table 1 shows the results[2]. Our incremental parser is competitive with the previous ones. The incremental parser with the monotonic adjoining operation outperforms the others. The result means that our proposed constraint of auxiliary trees improves parsing accuracy.

5 Conclusion

This paper has proposed an incremental parser based on an adjoining operation to solve the problem of infinite local ambiguity. The adjoining operation causes another problem that the parser cannot preserve lexical dependencies of partial parse trees. To tackle this problem, we defined the monotonicity of adjoining operation and restricted the form of auxiliary trees to satisfy the constraint of the monotonicity. Our experimental result showed that the restriction improved the accuracy of our incremental parser.

In future work, we will investigate the incremental parser for head-final language such as Japanese. Head-final language includes many indirect left-recursive structures. In this paper, we dealt with direct left-recursive structures only. To process indirect left-recursive structures, we need to extend our method.

[2]The best results of Collins and Roark (2004) (LR=88.4%, LP=89.1% and F=88.8%) are achieved when the parser utilizes the information about the final punctuation and the look-ahead. However, the parsing process is not on a word-by-word basis. The results shown in Table 1 are achieved when the parser does not utilize such informations.

References

James Allen, George Ferguson, and Amanda Stent. 2001. An architecture for more realistic conversational systems. In *Proceedings of International Conference of Intelligent User Interfaces*, pages 1–8.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, July.

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

Mark Johnson and Brian Roark. 2000. Compact non-left-recursive grammars using the selective left-corner transform and factoring. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 355–361, July.

Aravind K. Joshi. 1985. Tree adjoining grammars: How much context sensitivity is required to provide a reasonable structural description? In David R. Dowty, Lauri Karttunen, and Arnold M. Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press.

Yoshihide Kato, Shigeki Matsubara, and Yasuyoshi Inagaki. 2004. Stochastically evaluating the validity of partial parse trees in incremental parsing. In *Proceedings of the ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together*, pages 9–15, July.

Vincenzo Lombardo and Patrick Sturt. 1997. Incremental processing and infinite local ambiguity. In *Proceedings of the 19th Annual Conference of the Cognitive Science Society*, pages 448–453.

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

Brian Roark. 2004. Robust garden path parsing. *Natural language engineering*, 10(1):1–24.