

A Method for Accelerating CFG-Parsing by Using Dependency Information

Hideo Watanabe

IBM Research, Tokyo Research Laboratory
1623-14 Shimotsuruma, Yamato, Kanagawa 242-8502, Japan
watanabe@trl.ibm.co.jp

Abstract

This paper describes an algorithm for accelerating the CFG-parsing process by using dependency (or modifier-modifiee relationship) information given by, for instance, dependency estimation programs such as stochastic parsers, user's indication in an interactive application, and linguistic annotations added in a source text. This is a method for enhancing existing grammar-based CFG-parsing system by using dependency information.

1 Introduction

The parsing system is a key component for natural language applications such as machine translation, information retrieval, text summarization, and its performance (processing speed and accuracy) is very important to the success of these applications.

The usual CFG-parsing algorithms [3, 6] keep all intermediate possibilities which may or may not be used in the final parse results. Therefore, we usually reduce these intermediate possibilities which are unlikely to be used as final results in the middle of the process by using several pruning techniques. One good information source for pruning is dependency information between words. It has not been so easy to get such dependency information until a few years ago, but, the situation has recently changed.

Recent intensive studies on statistical approach [7, 1, 2] advanced statistical parsing systems, and we can get relatively correct dependency information using these systems. Further, if we suppose an interactive NLP system, then there are some types of user interactions which can be considered to determine the modifiee candidate. In addition, recent studies on the linguistic information annotation [10, 4, 12, 13] provide tools by which a user can easily annotate linguistic information (special XML markup tags) into source texts, and we can expect to see an increase of the number of texts with linguistic information. This linguistic information usually includes dependency information.

For instance, the following example shows an annotation example by Linguistic Annotation Language described in [12, 13], and the *id* and *mod* attributes inside *lal:w* elements specify word dependencies.

He $\langle \text{lal:w id="1"} \rangle$ saw $\langle /\text{lal:w} \rangle$ a man $\langle \text{lal:w mod="1"} \rangle$ with $\langle /\text{lal:w} \rangle$ a telescope.

In this example, the word "with" modifies the word "saw."

As shown in the above examples, we can now get dependency information more easily than a few years ago. This paper describes an algorithm for accelerating CFG-parsing systems by using such dependency (or modifier-modifiee relationship) information. The proposed algorithm does not assume all words are given dependency information, rather it works in case such that some of words are partially given dependency information.

2 Optimizing Algorithm Using Dependency Information

We use a normal CFG parsing system with one extension that for each rule there must be one right-hand side (or RHS) term¹ marked as a head, and the information of a head term is transferred to the left-hand side (or LHS) term. In this paper, a CFG rule is denoted as follows:

$$\{X \rightarrow Y_1 \dots Y_i^* \dots Y_n\} \quad (n > 0)$$

In the above notation, X is the left-hand side (or LHS) term, and Y_i are right-hand side (or RHS) terms, and a RHS term followed by an asterisk '*' is a head term. The typical usage of the head is that the LHS term shares many features of the head term in the RHS. For instance, a matching word of the the LHS term becomes the same as the one of the head term in the RHS.

For each rule, an arc is constructed over a word segment in an input sentence. An arc is denoted using terms of its base rule as follows:

$$[X \rightarrow Y_1 \dots Y_i \cdot Y_{i+1}^* \dots Y_n]$$

¹A term expresses a non-terminal symbol in LHS, and a non-terminal or a terminal symbol in RHS.

The LHS term of an arc means the LHS term of the base rule of the arc, and RHS terms of an arc means RHS terms of the base rule of the arc. In the above notation, a single dot indicates that RHS terms located to the left of a dot are inactive, that is, they already match the LHS term of some other arcs. Three dots are used to represent zero or any number of terms. An arc whose RHS terms are all inactive is called an inactive arc, otherwise it is called an active arc. An arc covers a segment of input words; the start point of an arc is the index of the first word in the covering segment, and the end point of an arc is 1 plus the index of the last word in the covering segment.

Basically, a standard CFG parsing algorithm such as [3, 6] consists of the following three operations.

Initialization: For each word, arcs are generated from rules such that the leftmost RHS term matches it.

Operation A: For each inactive arc A , an arc is generated from A and a rule R such that the leftmost RHS term of R matches the LHS term of A .

Operation B: For each inactive arc A , an arc is generated from A and another active arc B such that the leftmost active RHS term of B matches the LHS term of A and the end point of B is the same as the start point of A .

We assume that some dependency information between words are given, and such dependency information is denoted as follows:

$$W_x \Leftarrow W_y$$

$$W_x \Rightarrow W_y$$

The first of the above examples represents that a word W_y modifies another word W_x and W_x precedes W_y , while the second one represents that a word W_x modifies another word W_y and W_x precedes W_y .

Given this kind of dependency information, the following conditions are imposed on Operation A and Operation B.

Conditions for Operation A:

Condition A1 (when the leftmost RHS term of a rule is a head term):

Given an inactive arc Arc_I denoted by $[A \rightarrow \dots]$ and a rule which has two or more RHS terms and the leftmost RHS term is a head denoted by $\{X \rightarrow A * B \dots\}$, Operation A is executed only if there is dependency information $W_a \Leftarrow W_b$ where W_a is a word matching the LHS term A of Arc_I and W_b is a word

located anywhere to the right of the end point of Arc_I .

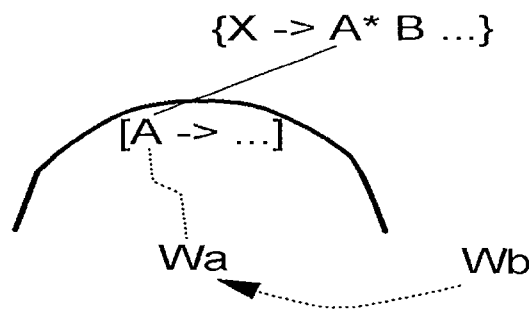


Figure 1: Condition A1

Figure 1 shows the above condition. In this figure, a thick arc represents an inactive arc, a line represents a matching to be tried in this operation, a dotted line represents a matching between a term in an arc and a word, and a dotted arrow represents dependency information. In this case, this type of rule implies that a word matching the LHS term of the arc to be matched with the leftmost term of the rule must be modified by any word which is located after the end point of the arc, since the head term is the leftmost term of the rule. Therefore, if the A1 condition does not hold, Operation A is not required to be executed.

Condition A2 (when the leftmost RHS term of a rule is not a head term):

Given an inactive arc Arc_I denoted by $[A \rightarrow \dots]$ and a rule which has two or more RHS terms and the leftmost RHS term is not a head denoted by $\{X \rightarrow A \dots D * \dots\}$, Operation A is executed only if there is a dependency information $W_a \Rightarrow W_b$ where W_a is a word matching the LHS term A of Arc_I and W_b is a word located anywhere after the end point of Arc_I .

Figure 2 shows the above condition. In this case, this type of rule implies that a word matching the LHS term of the arc to be matched with the leftmost term of the rule must modify any word which is located after the end point of the arc, since the head term is not the leftmost term of the rule.

Conditions for Operation B:

Condition B1 (when the leftmost active RHS term of an active arc is the head term):

Given an active arc Arc_A denoted by $[X \rightarrow A_0 \dots A_n . B * \dots]$ and an inactive arc Arc_I denoted by $[B \rightarrow \dots]$

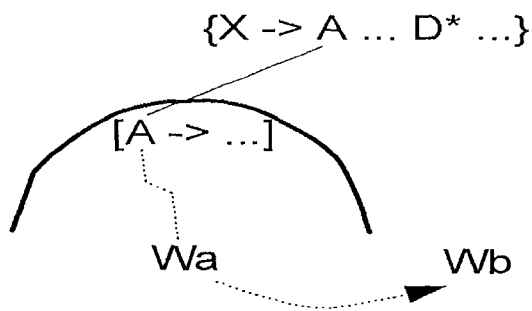


Figure 2: Condition A2

such that the end point of Arc_A is the same as the start point of Arc_I , Operation B is executed only if, for each W_{a_i} ($0 \leq i \leq n$) which is a word matching the RHS term A_i of Arc_A , there is dependency information $W_{a_i} \Rightarrow W_b$, where W_b is a word matching the LHS term B of Arc_I .

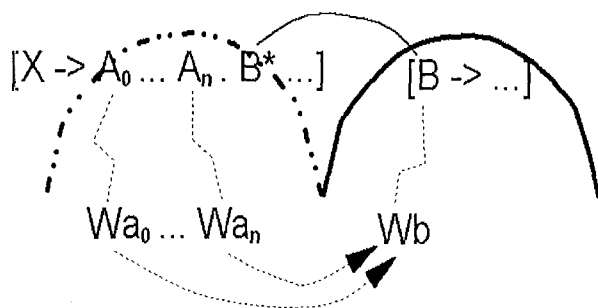


Figure 3: Condition B1

Figure 3 shows the above condition. In this figure, a dotted thick arc represents an active arc. In this case, this type of active arc implies that words matching inactive terms before the head term of the active arc must modify a word matching the LHS term of the inactive arc.

Condition B2 (when the head term is on the left side of the leftmost active RHS term of an active arc):

Given an active arc Arc_A denoted by $[X \rightarrow \dots A^* \dots B \dots]$ and an inactive arc Arc_I denoted by $[B \rightarrow \dots]$ such that the end point of Arc_A is the same as the start point of Arc_I , Operation B is executed only if there is dependency information $W_a \Leftarrow W_b$ where W_a is a word matching the RHS term

A of Arc_A , and W_b is a word matching the LHS term B of Arc_I .

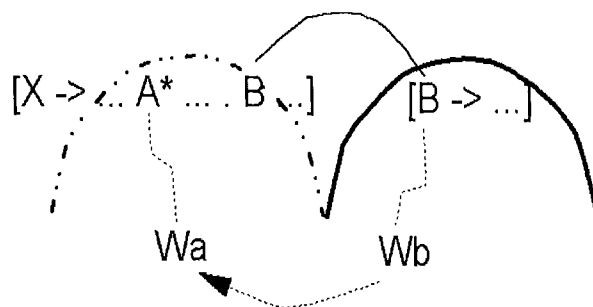


Figure 4: Condition B2

Figure 4 shows the above condition. In this case, this type of active arc implies that a word matching the LHS term of the inactive arc must modify a word matching the head term of the active arc.

Condition B3 (when the head term is on the right side of the leftmost active RHS term of an active arc):

Given an active arc Arc_A denoted by $[X \rightarrow A . B \dots C^* \dots]$ and an inactive arc Arc_I denoted by $[B \rightarrow \dots]$ such that the end point of Arc_A is the same as the start point of Arc_I , Operation B is executed only if there is dependency information $W_b \Rightarrow W_c$ where W_b is a word matching the LHS term B of Arc_I , and W_c is a word on the right side of the end point of Arc_I .

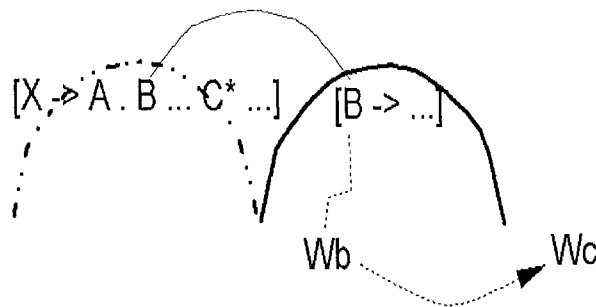


Figure 5: Condition B3

Figure 5 shows the above condition. In this case, this type of active arc implies that a word matching the LHS term of the inactive arc must modify a word after the end point of the inactive arc.

The dependency information is not necessarily given to all words. If there is any source word except for the root word of a sentence such that there

is no dependency information originating from it, then a set of such dependency information is called *partial*, otherwise, it is called *total*. If the given dependency information is partial, the A1 condition can not be used, since, even if there is no dependency information targeting W_a , we cannot know if such dependency information does not really exist, or if such dependency information is not supplied. For other conditions, we check them only when all source words for dependency checking have dependency information. On the other hand, if the given dependency information is total, all conditions are checked.

3 Experiment

We have implemented the proposed algorithm into an existing English CFG-parser we have developed for a machine translation product [8, 9, 11]², and conducted an experiment to know the effectiveness of this algorithm.

We selected 280 test sentences randomly from a sentence set created by JEIDA³ for evaluating translation system, and made the correct dependency relation data for these selected test sentences. We collected the number of inactive arcs, the number of active arcs, and the processing time for cases such that C modify candidates (one of which is the correct modifyee) are given to a word.⁴ If C=1 then it corresponds to the best case for a parser such that only one correct modifyee is given for each word, while if C is 3 or 4 then it corresponds to the approximation of using a statistical modifyee estimation program for getting candidate modifyees.

The graphs in Figure 6 indicate the reduction ratios of active arcs, inactive arcs, and processing time for using conditions for total dependency information and conditions for partial dependency information. The denominators for calculating these ratios are the numbers of arcs and the processing time (seconds) in case of the parser without this algorithm. In these graphs, C=X indicates that X is the maximum number of modifyee candidates given to a word.

From these graphs, we can see that the more words in a sentence, the better the performance. In a real domain, most sentences consist of more than ten words. Therefore, looking at values for around 10 in the X axis, we can see that inactive arcs are reduced by about 40% and 25%, active arcs

are reduced by about 65% and 35%, and processing time is reduced by about 45% and 15%, for the ideal case (C=1) and more practical cases (C=3 or 4), respectively, in the case of total dependency information. Please note that, since the parser in which this algorithm is implemented has already several pruning mechanisms, we can expect more reduction (or performance gain) for generic CFG parsers.

4 Discussion

As a study for accelerating the parsing process using dependency information, Imaichi[5] reported an algorithm for Japanese language. The conditions introduced by Imaichi are described by using the notation in this paper as follows:

Condition M1:

Given an active arc Arc_A denoted by $[X \rightarrow A . B^*]$ and an inactive arc Arc_I denoted by $[B \rightarrow \dots]$ such that the end point of Arc_A is the same as the start point of Arc_I , Operation B is executed only if there is dependency information $W_a \Rightarrow W_b$ where W_a is a word matching the RHS term A of Arc_A , and W_b is a word matching the LHS term B of Arc_I .

Condition M2:

Given an inactive arc Arc_I denoted by $[A \rightarrow \dots]$ and a rule denoted by $\{X \rightarrow A \dots\}$, Operation A is executed only if there is no dependency information $W_k \Rightarrow W_a$ where W_a is a word matching the LHS term A of Arc_I and W_k is a word located before the start point of Arc_I .

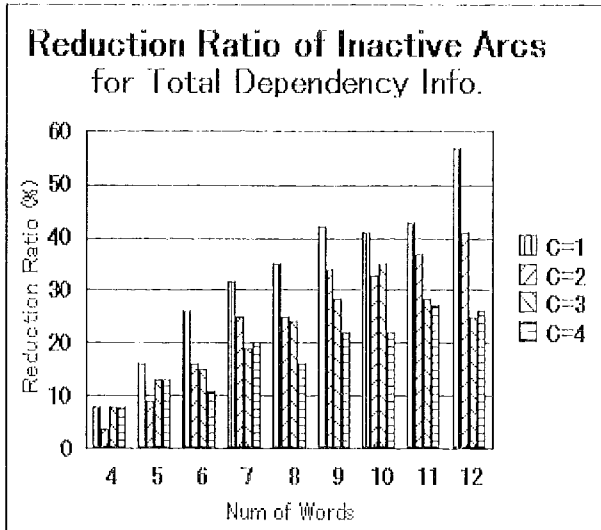
The condition M1 corresponds to B1. Since Imaichi's algorithm considers only Japanese in which all words other than the last word modifies one of the succeeding words, it does not deal with cases usually seen in European languages where a word modifies one of the preceding words. Therefore, it is not applicable to any language other than Japanese in general. Further, since a CFG rule is restricted to be in Chomsky normal form, Imaichi's algorithm is limited in terms of applicability.

Since the algorithm proposed in this paper does not have any restrictions on the dependency direction and the CFG rule format, it can be applicable to any CFG-parsers in any languages.

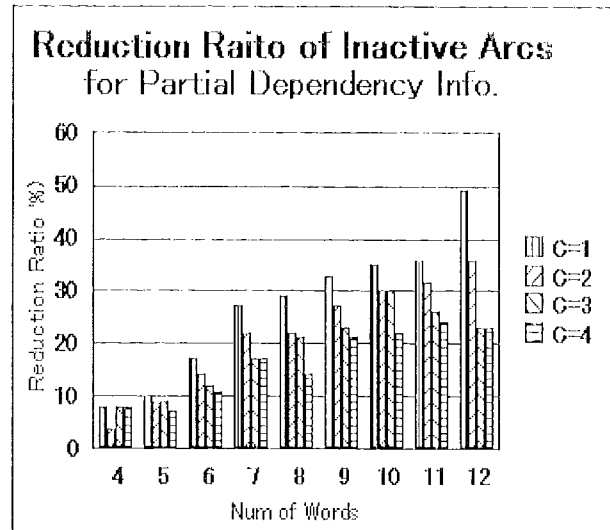
²This parser is used in a Web page translation software called "Internet King of Translation" released from IBM Japan.

³Japan Electronic Industry Development Association

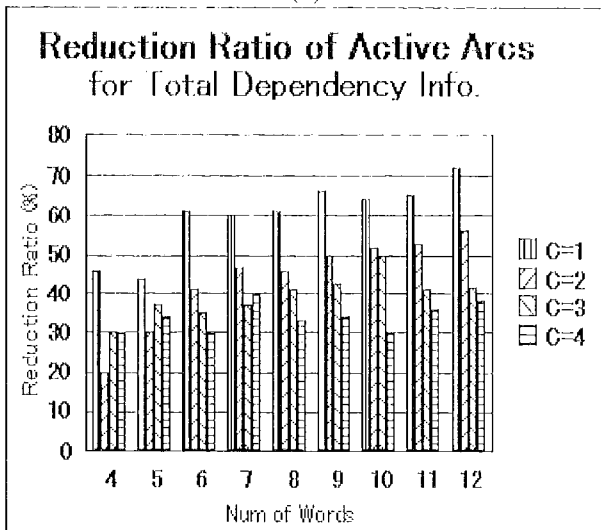
⁴Modifyee candidates are selected randomly except for the correct one.



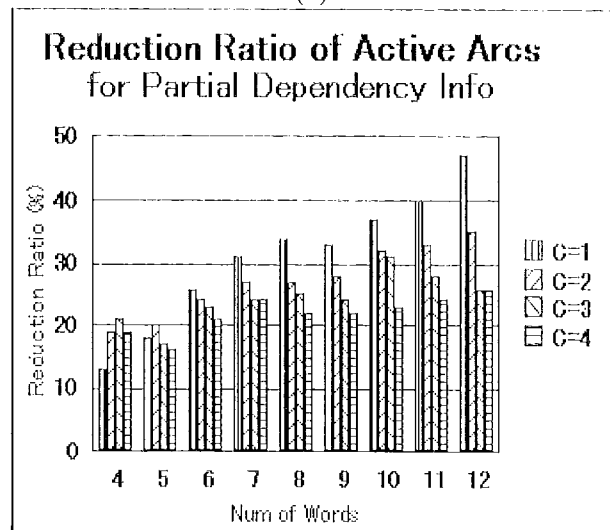
(a)



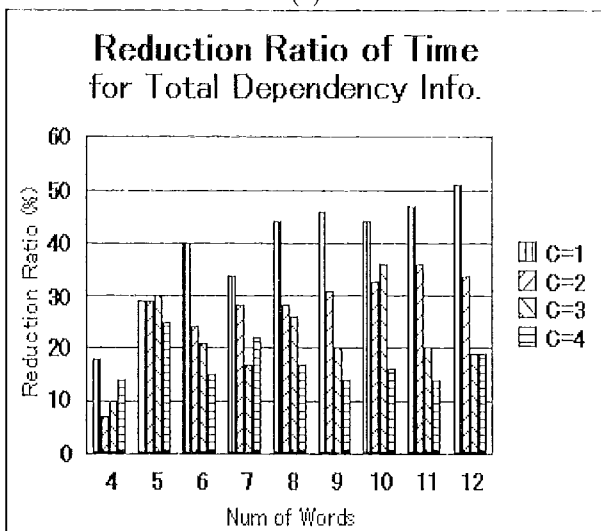
(b)



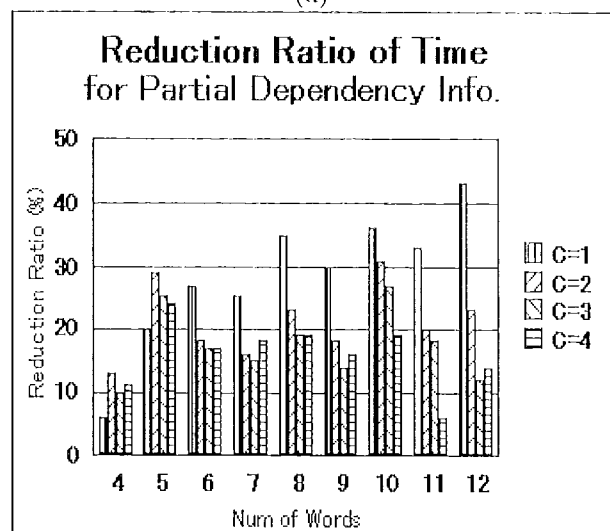
(c)



(d)



(e)



(f)

Figure 6: Reduction ratios of inactive arcs, active arcs, and processing time

5 Conclusion

We developed an algorithm for accelerating the performance of the CFG parsing process if we are given dependency information. From an experiment, we can show the effectiveness of this algorithm.

By using this algorithm, we can enhance existing grammar-based parsers using dependency information given by stochastic parsers, interactive systems, and texts created by linguistic annotation systems.

References

- [1] M. Collins. A new statistical parser based on bigram lexical dependencies. In *Proc. of 34th ACL*, pages 184–191, 1996.
- [2] M. Collins. Three generative, lexicalized models for statistical parsing. In *Proc. of 35th ACL*, pages 16–23, 1997.
- [3] J. Earley. An efficient context-free parsing algorithm. In *Readings in Natural Language Processing*. Morgan Kaufman, 1969.
- [4] K. Hashida, K. Nagao, et al., Progress and Prospect of Global Document Annotation. (in Japanese) In *Proc. of 4th Annual Meeting of the Association of Natural Language Processing*, pp. 618–621, 1998.
- [5] O. Imaichi, Y. Matsumoto, and M. Fujio. An integrated parsing method using stochastic information and grammatical constraints. *Journal of Natural Language Processing*, 5(3):67–83, 1998.
- [6] M. Kay. Algorithm schemata and data structure in syntactic processing. Technical Report CSL-80-12, Xerox PARC, 1980.
- [7] D. M. Magerman. Statistical decision-tree models for parsing. In *Proc. of 33rd ACL*, pages 276–283, 1995.
- [8] K. Takeda. Pattern-based context-free grammars for machine translation. In *Proc. of 34th ACL*, pages 144–151, 1996.
- [9] K. Takeda. Pattern-based machine translation. In *Proc. of 16th Coling*, volume 2, pages 1155–1158, 1996.
- [10] Text Encoding Initiative (<http://www.uic.edu:80/orgs/tei/>)
- [11] H. Watanabe and K. Takeda. A pattern-based machine translation system extended by example-based processing. In *Proc. of 17th Coling (Coling-ACL'98)*, volume 2, pages 1369–1373, 1998.
- [12] H. Watanabe, Linguistic Annotation Language - The Markup Language for Assisting NLP Programs -. IBM Research Report RT0334, 1999.
- [13] H. Watanabe, K. Nagao, et al., Linguistic Annotation System for Improving the Performance of Natural Language Processing Programs. In *Proc. of 6th Annual Meeting of The Association for NLP (in Japanese)*, pp. 171–174, 2000.