April 23, 1999
RT0316
Computer Science 9 pages

Research Report

A Method for Accelerating CFG-Parsing Using Dependency Information

Hideo Watanabe

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

A Method for Accelerating CFG-Parsing Using Dependency Information

Hideo Watanabe
IBM Research, Tokyo Research Laboratory
1623-14 Shimotsuruma, Yamato, Kanagawa 242-8502, Japan
hiwat@jp.ibm.com

Abstract

This paper describes an algorithm for accelerating the CFG-parsing process by using dependency (or modifier-modifiee relationship) information given by, for instance, modifiee estimation program using stochastic approaches or user's indication in an interactive application. This is a method for enhancing existing grammar-based parser by using dependency information.

1 Introduction

The parsing system is becoming to be widely used in many applications such as machine translation, information retrieval, text summarization. In these environments, it is very important for parsing systems to have high performance as well as high accuracy.

Recent intensive studies on statistical parsing system (Magerman 1995; Collins 1996; Collins 1997) facilitate to construct a system predicting plausible modifiee candidates for each word. Further, if we suppose an interactive NLP system, then there are some types of user interactions which can be considered as the modifiee candidate indication for a word. On the other hand, since the usual CFG-parsing algorithms (Earley 1969; Kay 1980) keep all intermediate possibilities which may or may not be used in the final parse results, there are rooms to be optimized given some kinds of information outside the parsing system. Therefore, the dependency information should be effectively used for reducing the intermediate data in the parsing system. This paper describes an algorithm for accelerating CFG-parsing systems by using dependency (or modifier-modifiee relationship) 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) term[1] 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:

$$\{X \rightarrow Y_1 \ldots Y_i{*} \ldots Y_n\} \ (n > 0)$$

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

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 is 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:

$[X \rightarrow Y_1 \ldots Y_i.Y_j{*} \ldots Y_n]$

The LHS term of an arc means the LHS term of the base rule of an arc, and RHS terms of an arc means RHS terms of the base rule of an arc. In the above notation, a single dot indicates that RHS terms located in the left side of a dot are inactive, that is, they already match the LHS term of any other arcs. Further, 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 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 usual parsing CFG algorithm (Earley 1969; Kay 1980) 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 the scope of B is the same as the start point of the scope of A.

We assume that 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 one 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 as $[A \rightarrow ...]$ and a rule which has more than two RHS terms and the leftmost RHS term is a head denoted as $\{X \rightarrow A * B\ ...\}$, Operation A is executed only if there is a 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 at anywhere to the right of the end point of Arc_I.

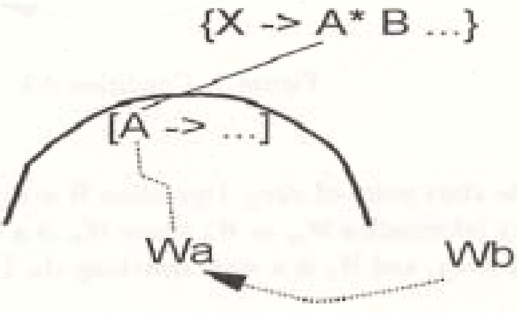


Figure 1: Condition A1

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 on the right side of the scope 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 needed 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 as $[A \rightarrow ...]$ and a rule which has more than two RHS terms and the leftmost RHS term is not a head denoted as $\{X \rightarrow A\ ...\ D*\ ...\}$, 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 at anywhere to the right of the end point of Arc_I.

Figure 2 shows the above condition. In this case, this type of rule implies that a word mathing the LHS term of the arc to be matched with the leftmost term of the rule must modify any word which is located on the right side of the scope 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 as $[X \rightarrow A_0 ... A_n . B*\ ...]$ and an inactive arc Arc_I denoted as $[B \rightarrow ...]$ such that the end point of Arc_A is the

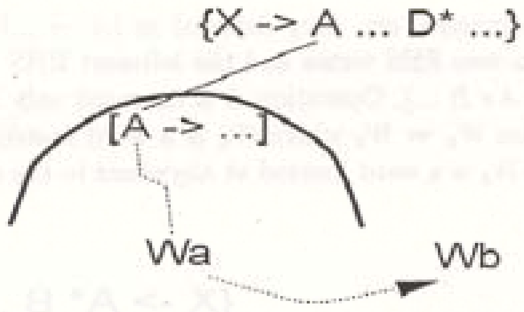


Figure 2: Condition A2

same as the start point of Arc_I, Operation B is executed only if there is a dependency information $W_{ai} \Rightarrow W_b$ where W_{ai} is a word matching the RHS term A_i of Arc_A, and W_b is a word matching the LHS term B of Arc_I.

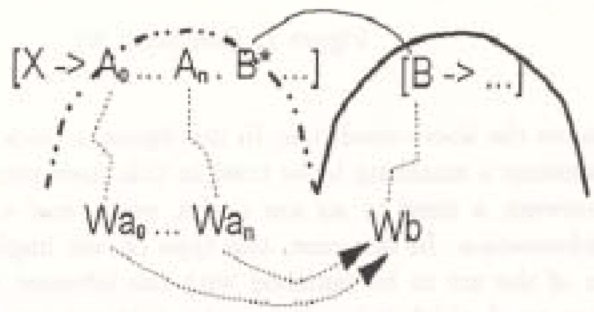


Figure 3: Condition B1

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 on the left side of 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 as $[X \rightarrow ... A* ... .B ...]$ and an inactive arc Arc_I denoted as $[B \rightarrow ...]$ 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 a 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 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):

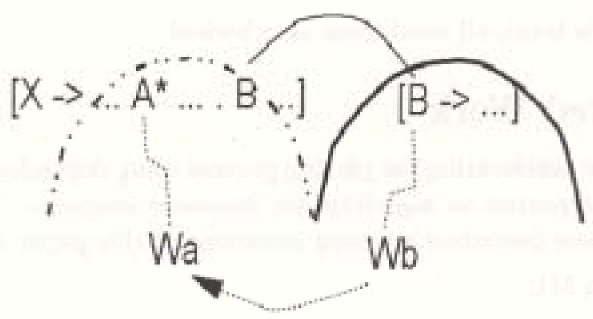


Figure 4: Condition B2

Given an active arc Arc_A denoted as $[X \rightarrow A.B \ldots C* \ldots]$ and an inactive arc Arc_I denoted as $[B \rightarrow \ldots]$ 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 a 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.

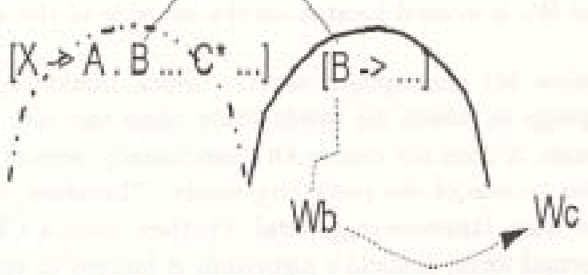


Figure 5: Condition B3

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 on the right side of the scope 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 originated 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 and B2 conditions can not be used, since we cannot know if dependency information ending at a word (W_a in both conditions) in question for these conditions is not given. On the other hand, for other conditions, if there is no dependency information originating from a word in question, then it is apparent that any dependency information originating the word does not exist. Therefore, in this case, these conditions do not have to be checked and are regarded as to be satisfied, that is, the operation is executed. In other words, for conditions A2, B1 and B3, they are checked only if there is a dependency information whose source is a word in question. If the given dependency information is total, all conditions are checked.

3 Related Work

As a study for accelerating the parsing process using dependency information, Imaichi(Imaichi et al. 1998) reported an algorithm for Japanese language. The conditions introduced by Imaichi were described by using notations 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 $[B \rightarrow ...]$ 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 a 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 as $[A \rightarrow ...]$ and a rule denoted as $\{X \rightarrow A ...\}$, Operation A is executed only if there is not a 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 on the left side of the start point of Arc_I.

The condition M1 corresponds to B1. Since Imaichi's algorithm considers only Japanese language 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 that a word modifies to 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 a 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.

4 Experiment

We have implemented the proposed algorithm into an existing English CFG-parser (Takeda 1996a; Takeda 1996b; Watanabe and Takeda 1998) , and conducted an experiment to know the effectiveness of this algorithm.

We selected 280 test sentences randomly from a sentence set created by JEIDA[2] 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 modifiee candidates (one of which is the correct modifiee) are given to a word. [3] If C=1 then it corresponds to the best case for a parser such that only one correct modifiee is given for each word, while

[2]Japan Electronic Industry Development Association

[3]Modifiee candidates are selected randomly except for the correct one.

if C is around 3 or 4 then it corresponds to the approximation of using a statistical modifiee estimation program for getting candidate modifiees.

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

From these graphs, we can see that the more the number of words in a sentence is, the better the performance is. 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 about 40% and 25%, active arcs are reduced about 65% and 35%, and processing time is reduced 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.

5 Conclusion

We developed an algorithm for accelerating the performance of 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 information given by stochastic approaches. Further, suppose that parsing system is used for an interactive application, some of interactions can be used for accelerating the parsing process by using this algorithm.

References

M. Collins. A new statistical parser based on bigram lexical dependencies. In *Proc. of 34th ACL*, pages 184–191, 1996.

M. Collins. Three generative, lexicalized models for statistical parsing. In *Proc. of 35th ACL*, pages 16–23, 1997.

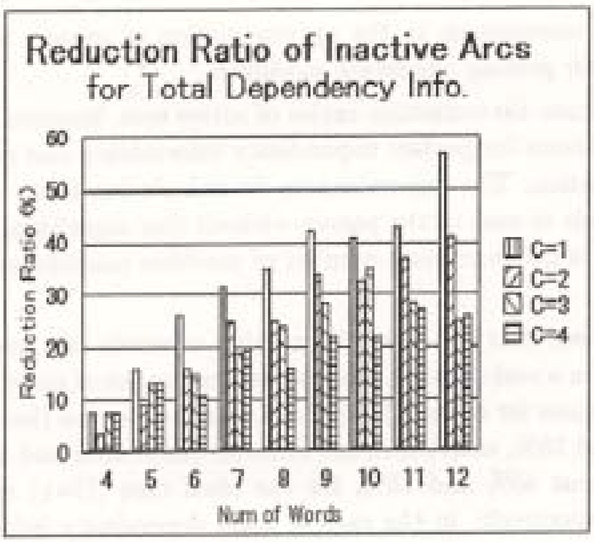
J. Earley. An efficient context-free parsing algorithm. In *Readings in Natural Language Processing*. Morgan Kaufman, 1969.

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.

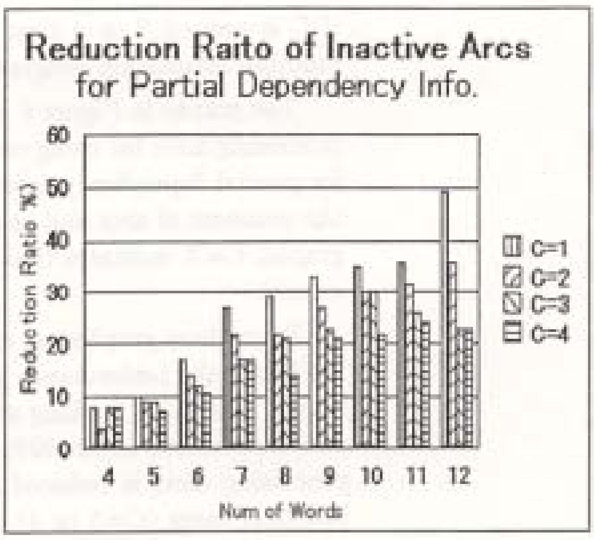
M. Kay. Algorithm schemata and data structure in syntactic processing. Technical Report CSL-80-12, Xerox PARC, 1980.

D. M. Magerman. Statistical decision-tree models for parsing. In *Proc. of 33rd ACL*, pages 276–283, 1995.

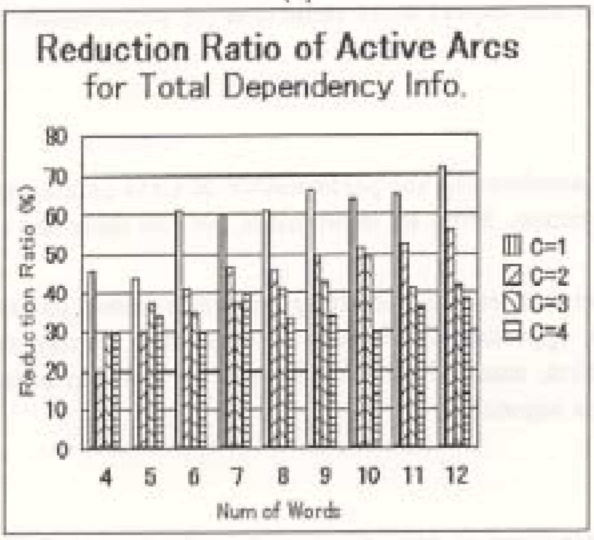
K. Takeda. Pattern-based context-free grammars for machine translation. In *Proc. of 34th ACL*, pages 144–151, 1996.



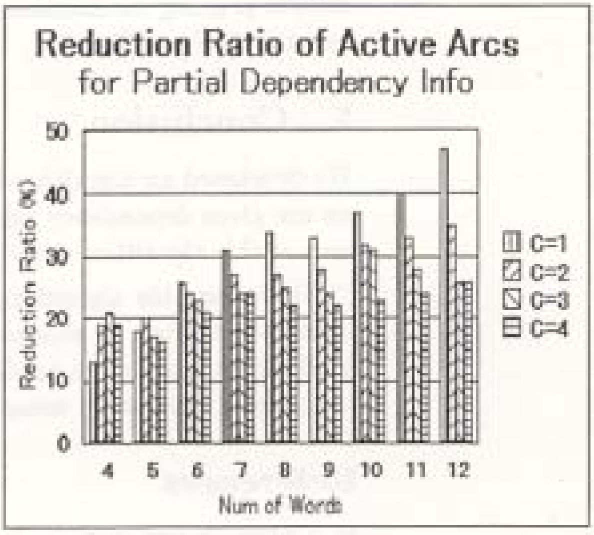
(a)



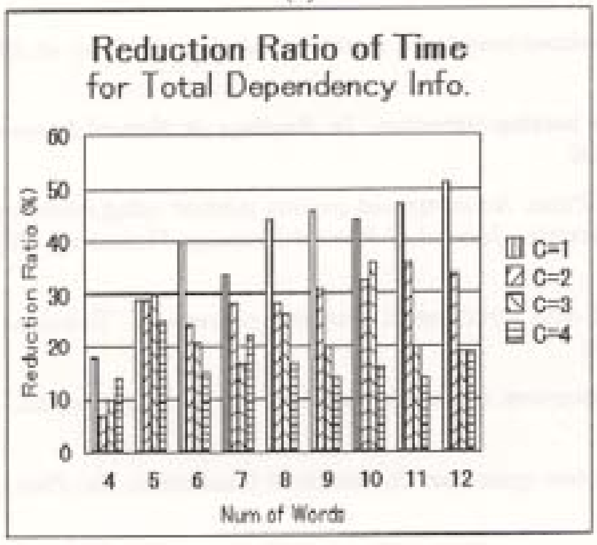
(b)



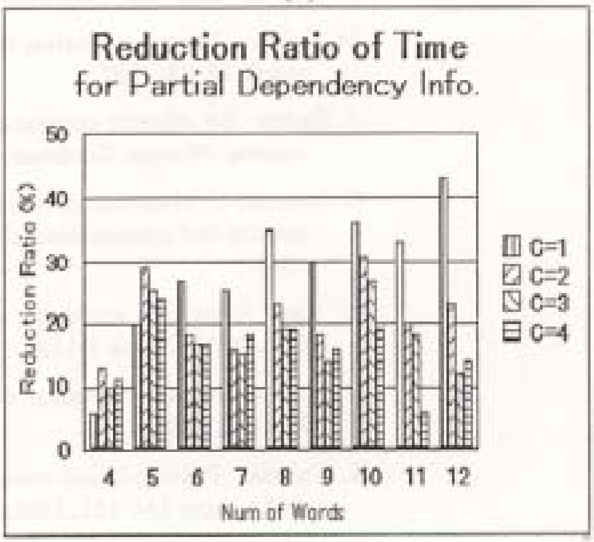
(c)



(d)



(e)



(f)

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

K. Takeda. Pattern-based machine translation. In *Proc. of 16th Coling*, volume 2, pages 1155–1158, 1996.

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.