係り受け関係を用いたＣＦＧ構文解析の枝刈手法*

2N－3

渡辺 日出雄†
日本アイ・ビー・エム株式会社 東京基礎研究所‡

1 はじめに

対話的な言語処理システムを考えた場合、ユーザーから単語間の依存関係（係り受け関係）に相当する情報が得られるということを想定することができる。このような場合に得られる単語間の依存関係情報は、構文解析の規則適用における制約として使用できるはずであり、これにより効率の良い構文解析システムを構築することができる。また、依存関係情報の供給源としては、今一ら[1]の研究により示されたように、最近では統計的な係り受け解析システムを考えることもできる。本論文では、このような単語間の依存関係情報を制約として利用したＣＦＧ構文解析システムの枝刈手法について報告する。

2 依存構造を利用した枝刈処理

2.1 文脈自由文法への制約

本論文で使用する文脈自由文法は、以下の条件を満たすものであると仮定する。

- 全てのルールの右辺にはただ一つのヘッドとしてマークされた項がある。
- 全ての項はそれとマッチした単語のインデックスを持つ。このインデックスはマッチした項、および、右辺のヘッド項から左辺項へと継承される。

たとえば、以下のように表記する。右辺項の＊でマークされているのがヘッドである。

X → A B* C

2.2 枝刈手法

一般に通常のＣＦＧ構文解析は、以下の二つの処理Ａ及びＢにより行われる。

処理Ａ それぞれの不活性アークａについて、ある規則ｒの最左右辺項とａの左辺項[1]がマッチする場合、ｒから新たなアークを作成する。

処理Ｂ それぞれの不活性アークａについて、ａの始点を終点とするアークｂがあり、そのｂの最左活性項とａの左辺項がマッチする場合、ａとｂから新たなアークを作成する。

以下の説明では、規則とアークを以下のように表記することにする。アークの表記では、右辺中にあるドットでその右隣の項が最左活性項であることを示している。

規則: {X → A B* C}
アーク: [X → A . B C]

さて、ここで幾つかの単語間の依存関係が与えられた場合、以下の追加条件をチェックする。

条件Ａ１：不活性アーク Arc_I([A → ...]) と、二つ以上の右辺項を持ち最左右辺項がヘッド項であるような規則 ({X → A* B ...}) が与えられたとき、依存関係情報 $W_a \Leftarrow W_b$ [2]がある場合にだけ処理Ａを行う。ここで、W_a は Arc_I の左辺項 A とマッチした単語、W_b は Arc_I の終点より後方にある単語である。

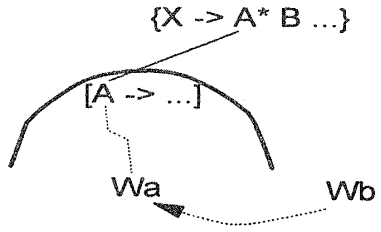

図 1: Condition A1

条件Ａ２：不活性アーク Arc_I([A → ...]) と、二つ以上の右辺項を持ち最左右辺項がヘッド項でないような規則 ({X → A ... D* ...}) が与えられたとき、依存関係情報 $W_a \Rightarrow W_b$ がある場合にだけ処理Ａを行う。ここで、W_a は Arc_I の左辺項 A とマッチする単語、W_b は Arc_I の終点より後方にある単語である。

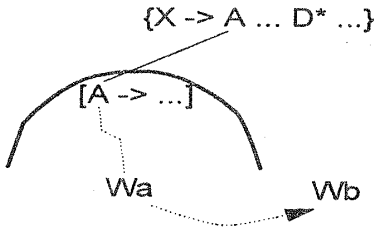

図 2: Condition A2

条件Ｂ１：不活性アーク Arc_I([B → ...]) と、その終点と同じ始点を持つ活性アーク Arc_A([X → A_0 ... A_n . B* ...]) が与えられたとき、依存関係情報 $W_{ai} \Rightarrow W_b$ がある場合にだけ処理Ｂを行う。ここで、W_{ai} は Arc_A の右辺項 a_i とマッチした単語、W_b は Arc_I の左辺項 B とマッチした単語である。

条件Ｂ２：不活性アーク Arc_I([B → ...]) と、その終点と同じ始点を持つ活性アーク Arc_A([X → ... A* ... . B ...]) が与えられたとき、依存関係情報 $W_a \Leftarrow W_b$ がある場合にだけ処理Ｂを行う。ここで、W_a は Arc_A の右辺項 A とマッチした単語、W_b は Arc_I の左辺項 B とマッチした単語である。

---

*A Pruning Method of CFG-Parsing Using Dependency Information

†Hideo Watanabe (watanabe@trl.ibm.co.jp)
‡IBM Research, Tokyo Research Laboratory

[1]アークの左辺項とはそのアークを構成する元になる規則の左辺項を意味する。

[2]$W_a \Leftarrow W_b$ は W_b が W_a に係ることを表し、$W_a \Rightarrow W_b$ は W_a が W_b に係ることを示す。

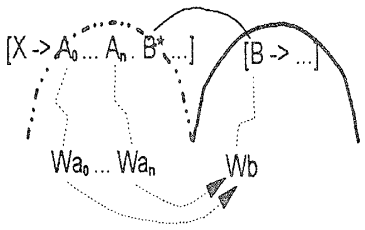

図 3: Condition B1

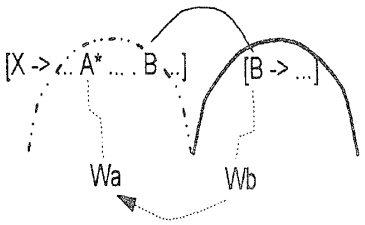

図 4: Condition B2

条件B3：不活性アーク Arc_I([B → ...]) と、その終点と同じ始点を持つ活性アーク Arc_A([X → A . B ... C* ...]) が与えられたとき、依存関係情報 $W_b \Rightarrow W_c$ がある場合にだけ処理Bを行う。ここで、W_c は Arc_I の終点より後方の任意の単語である。ここで、W_b は Arc_I の左辺項 B とマッチした単語、W_c は Arc_I の終点より後方にある単語である。

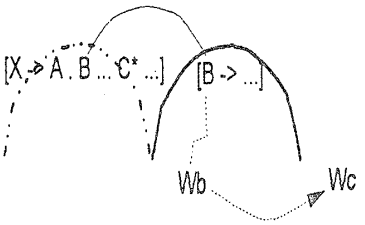

図 5: Condition B3

上記の追加条件の内、依存関係情報が全ての単語について与えられていない場合には、A1 および B2 は使用しない。また、A2,B1,B3 についても、依存関係情報の始点となる単語に関して何らかの依存関係情報がある場合だけチェックを行う。

3 実験

上記の枝刈手法を既存の英語パーサー [2] に適用してみた。JEIDA の英日機械翻訳用評価文の中からランダムに 280 文を取り出し、それに単語間の正しい係り受け関係を人手で付与した。本手法のパフォーマンスを調べるため、一つの単語に対して与えられる係先単語候補の数 C、文中の単語数、および、依存情報が全ての単語について与えられているかどうかをパラメータとし、C が 1 から 4 まで、単語数が 4 から 12 までで、構文解析の途中結果として生成される活性アークと不活性アークの削減率を計測した。図 6 に結果を示す。

全ての単語に依存関係情報が与えられた場合で単語数が 10 近辺を見てみると、C=1 の理想的なケースで不

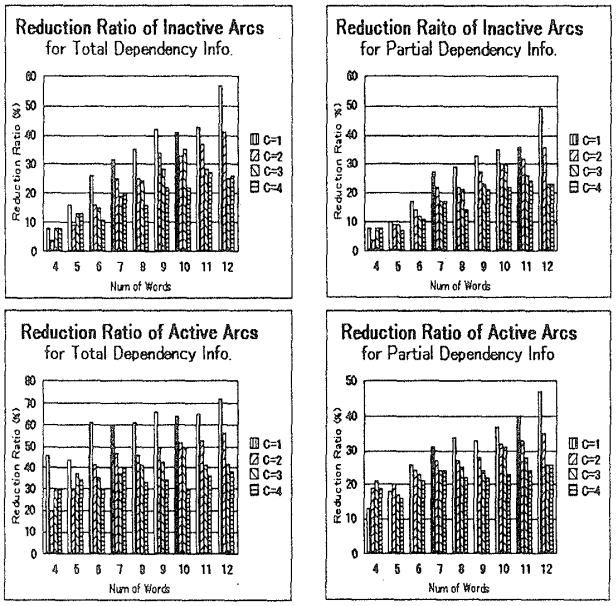

図 6: 不活性アークと活性アークの削減率

活性アーク数が約 40%、活性アーク数が約 65%の削減率であり、C=3 あるいは 4 のケース[3]では、それぞれ約 25%と約 35%の削減率となっている。ただし、この実験で用いた英語パーサーは既に幾つかの枝刈手法を導入済みのものであり、枝刈処理を施していないパーサーではこれ以上の削減が期待できる。

4 既存手法との比較

今一ら [1] による研究では、係り受けが後方に限定されるという日本語の特質を仮定した場合のアルゴリズムが紹介されている。しかし、これでは英語などの前方への係り受けがある言語に適用できないという問題があった。

本論文では、係り受けの方向に依存しない、任意の言語の解析に適用可能な依存構造を用いた枝刈手法のアルゴリズムを提案した。

5 おわりに

本論文では、単語間の依存関係（係り受け）情報を利用したＣＦＧ構文解析の枝刈手法について報告し、依存情報を用いることで既存の規則ベースのＣＦＧ構文解析システムにおいてかなりのパフォーマンスの向上を見込めることを示した。

参考文献

[1] 今一修、松本祐治、藤尾正和, 統計情報と文法制約を統合した統語解析手法, 自然言語処理,Vol. 5, No. 3, pp. 67–83, 1998.

[2] H. Watanabe and K. Takeda, A Pattern-based machine translation system extended by exmaple-based processing, Coling-ACL '98, Vol. 2, pp. 1369–1373, 1998.

[3] これは統計的な係り受けパーサーの出力を利用した場合と考えることもできる