拡張LR構文解析アルゴリズムによる未定義語を含む日本語文の構文解析

2E-1

堀内 靖雄　伊藤 克亘　田中 穂積
(東京工業大学 工学部)

1 はじめに

自然言語を構文解析する際に、すべての単語が辞書に登録されていると一般には仮定できない。構文解析システムは、入力文が未定義語を含む場合にも失敗しないで、ある文字列が未定義語と仮定して構文的に妥当であるならば、そういった可能性も出力すべきである。

日本語は分かち書きされていないので、未定義語の抽出時に多くの可能性が生じるが、その中で構文的に受容可能な未定義語はすべて抽出できることが望ましく、それらの候補の絞り込みは意味解析などのより上位の処理に任せればよいと思われる。[大場 89]は多段階に構文解析することにより受容可能なすべての未定義語を抽出しているが、一部の文字列を何回も解析するため効率が悪い。本研究では拡張LR構文解析アルゴリズムを用いることにより、入力文を左から右へ読み進みながら決定的に解析し、一度の構文解析で受容可能な未定義語をすべて抽出する方法を提案する。

しかし、どんな文字列でも未定義語の可能性があるとしてしまうと、指数関数的に処理量が増えてしまう。本研究では、形態素解析と構文解析を同時に行ない、構文解析時の予測によって受容不可能な候補を刈り込み、解析の候補を絞り込んでいる。また複数のプロセスで同じ処理をしないようにすることによって処理量の増大を抑制している。

本研究では処理量を減らすために、未定義語の品詞は名詞であると仮定する。

また日本語の入力形式には、漢字、平仮名、ローマ字等様々なものがある。本システムでは、将来音声認識システムと結合することも考慮に入れ、ローマ字列を入力とする。ただし構文解析前に入力ローマ字列は、「かっぱ」[ka,Q,pa]のように仮名(音節)に対応したシンボル列に変換される。

2 解析方法

2.1 構文解析の概要

本研究では、拡張LR構文解析アルゴリズム(Generalized LR Parser)[Tomita 85]を用いて構文解析を行なっている。拡張LR構文解析アルゴリズムは、LR構文解析アルゴリズム[Knuth 65]を文脈自由文法を扱えるように拡張したものであり、構文解析前に構文解析中の動作を決定するLRテーブルを文法から作成しておく。構文解析時には、入力文を左から右へと一方向に読みながら、LRテーブルに従って次の状態を決定的に決めて構文解析を進めて行く。解析途中で曖昧性が生じるとそれ以後の処理を2つに分割し、それらを並列に実行して解析を進めてゆく。この時、未定義語により生じる曖昧性も同様に並列に処理する。本アルゴリズムは並列論理型言語GHC[Ueda 85]により記述している。

しかしこのアルゴリズムのナイーブな実現法では、文法的曖昧性あるいは未定義語による曖昧性が生じる度にプロセスが分割され、処理量が指数関数的に増えてしまう。例えば、文頭の方で曖昧性が生じたとすると、それ以降の解析が同じであっても2つのプロセスで同じ処理をしてしまう。そこで本研究では、[沼崎 89]の木構造化スタックを用いている。このアルゴリズムでは構文解析中、同じ状態にあるとき、今までの構文解析経過を持っている2つのスタックを結合して、以後一つの処理にまとめてしまうことにより、処理量が増大するのを抑制している。

また、辞書引き処理も拡張LRパーザをもとにして行なうことにより、効率を上げている。辞書引き処理は、入力文字列を一文字ずつ読み込みLRテーブルにしたがって遷移を行ない、遷移が行なえなくなるまで辞書を引き、辞書引きできたすべての単語のリストを返す。辞書は品詞ごとにブロック化されている。次節で詳述するように、拡張LRパーザでは、構文解析時に次に来る単語の品詞を予測しながら解析を進めるので、その特性を利用して、その品詞しか辞書引きを行なわないようにするためであり、そうすることにより効率良く辞書引きを行なうことができる。また辞書の保守や分割コンパイルなども容易になる。辞書を分割するとLRテーブルの数は増えるが、辞書を文法に組み込み一つのテーブルにまとめたLRテーブルと、品詞ごとにブロック化したLRテーブルとは、実際にはほとんど等価である。またここでは、同じ辞書引きを繰り返さないようにして効率化を図っている。

このように次に来る品詞を予測できると、未定義語の可能性のある品詞の時だけ未定義語の処理を行なえば良いので、構文解析システムに容易に未定義語の処理を組み込める。

また本研究では、形態素解析を構文解析と完全に融合して行なっている。そのため接続条件などは文法に直接記述している。このように形態素情報を文法に組み込むことにより文法は大きくなるが、形態素解析を構文解析と独立に行なう場合に比較して無駄な形態素解析の候補を生成しないので、それだけ処理効率は上がる。また、LRテーブルは一度作ってしまえば、文法に変更がない限りそのまま使用でき、サイズが大きくても高速に構文解析が可能である。

A Parsing System Using Generalized LR Argorithm
for Japanese Sentence Containing Undefined Words
Yasuo HORIUCHI　Katsunobu ITOU　Hozumi TANAKA
Department of Computer Science, Tokyo Institute of Technology

2.2 解析方法

実際の解析方法について例を挙げて説明する。解析例として、

「自由が丘へ行く。」

(jiyuugaokaeyuku.)

を考える。これを構文解析すると、「自由が丘」が辞書になければ、解釈としては

自由 (名詞) が 丘 (名詞) へ 行く。

自由が丘 (未定義名詞) へ 行く。

の2つが得られるべきである。しかし未定義語の処理を行なっているパーザでも、このように登録語だけで構文解析可能な解釈が存在すると、未定義語を含む解釈が行なわずに最初の候補しか出力しない [高橋 86][塚田 89]。このような手法では、未定義語を含むとして処理した解釈の方が意味的に妥当である場合にも、その解釈が不可能になる。本システムではこのような入力に対して、以上の2つの構文解析結果を出力することができる。

以下に解析過程を示し、解析方法の概要を示す。まず構文解析部は、LR テーブルから次に来る単語の品詞を予測し、来る可能性のあるすべての品詞についてプロセスを起動し、それぞれのプロセスが並列に辞書引きを行ない構文解析を進めてゆく。そのとき辞書引きに失敗したプロセスはそこで死ぬものとする。

予測される品詞は文法に依存するが、ここでは名詞と動詞の語幹が予測されたとする。ここで2つのプロセスが起動され、それぞれが名詞と動詞の語幹の辞書引きを行なう。入力文に対して動詞の語幹を辞書引きすると、この辞書引きは失敗し、このプロセスは死ぬ。名詞を予測したプロセスは、名詞の辞書を引くが、名詞は未定義語の可能性があるため、名詞の辞書引きでは、辞書に登録されていない文字列は未定義語として辞書引きに成功するため、この例では、以下の文字列が名詞として返される。

[ji]
[ji, yu]
[ji, yu, u]
[ji, yu, u, ga]
[ji, yu, u, ga, o]
[ji, yu, u, ga, o, ka]
[ji, yu, u, ga, o, ka, e]
[ji, yu, u, ga, o, ka, e, yu]
[ji, yu, u, ga, o, ka, e, yu, ku]

ただしこの中で辞書に登録されているものは [ji, yu, u] だけである。

ここで辞書引きできたすべての単語についてプロセスを起動し、それぞれが残りの文を構文解析する。次の段階では、予測される品詞は助詞である。そこで、この8つのプロセスは、それぞれ助詞の辞書引きを行なう。すると助詞の辞書引きに成功するのは以下の3つであり、それ以外のプロセスは死んでしまう。

[ji, yu, u], [ga]
[ji, yu, u, ga], [o]
[ji, yu, u, ga, o, ka], [e]

この例から明らかなように未定義語の候補がたくさん出ても、本手法では形態素解析と構文解析を同時に行ない、かつ、LR テーブルにより、次に来るべき単語の構文情報を予測として用いているため、構文的に受容不可能なプロセスは、その後の数文字の処理で死んでしまい、処理量がそれほど増大することはない。

その後この3つのプロセスが更に構文解析を進めてゆく。2番目のプロセスはその後、[ka] と [ka, e] が動詞の語幹として辞書引きに成功するが、[e, yu, ku, '.'] と [yu, ku, '.'] の時点で辞書引きに失敗して死んでしまう。そして、結果として最初に述べたような2つの結果が出力される。

3 おわりに

未定義語を含む日本語文の構文解析を並列拡張 LR 構文解析システムを用いて実現した。ここでは形態素解析と構文解析を同時に行ない、構文解析時の予測を用いることにより、すべての未定義語を効率良く抽出することが可能となった。この時使用する辞書が LR テーブルとしてコンパイルされている点に特徴がある。

本システムでは構文解析の結果が複数生じたとき、それぞれの候補の妥当性について順序付けを行なっていないが、未定義語の数、長さなどによって順序を決めることができる。今後は、解析結果をスコア付けして表現し、スコアの高い順に出力することを考えたい。

また、意味処理部とも結合し、意味情報を用いて複数の候補にさらにスコアを付けて絞り込み、より正しいと思われる候補から出力することも考えている。

更に音声認識部とも結合して、構文情報を音声認識に役立たせると共に、音韻以外の音声情報をも構文解析、意味解析に積極的に利用することも考えている。

また、この拡張 LR 構文解析アルゴリズムは決定的でバックトラックがないので、Pascal,C 等の手続き型言語上で実現可能であり、処理速度向上のため C 言語上で本システムを実現する予定である。

参考文献

[大場 89] 大場, 元吉, 井佐原, 横山, 石崎, 板橋:
未定義語を含む文の多段階解析法
情報処理学会第 38 回全国大会 5e-5, pp.372-372 (1989)

[高橋 86] 高橋, 奥村, 伊藤, 小川:
構文解析における未定義語の抽出
情報処理学会第 33 回全国大会 3k-7, pp.1811-1812 (1986)

[塚田 89] 塚田, 西野, 小柳:
未登録語を含む文の一解析法
自然言語処理 73-6, pp.43-50 (1989)

[沼崎 89] 沼崎, 田村, 田中:
並列論理型言語による一般化 LR 構文解析アルゴリズムの実現
自然言語処理 74-5, PP.33-40 (1989)

[Knuth 65] Knuth, D.E.:
On the translation of languages from left to right
Information and Control 8:6, pp.607-639 (1965)

[Tomita 85] Tomita, M.:
Efficient Parsing for Natural Language
Kluwer Academic Publishers (1985)

[Ueda 85] Ueda, K.:
Guarded Horn Clauses
Proc. The Logic Programming Conference, Lecture Notes in Computer Science, 221 (1985)