構文解析における未定義語の抽出

3K-7

高橋　清一　奥村　昌司　伊藤　篤　小川　束
㈱ＣＳＫ総合研究所

1．はじめに

構文解析には，辞書・文法（構文規則）が必要である．しかし，入力文中の全ての単語がその辞書に登録されているとは限らない．そのため，未登録の単語（未定義語）が含まれた文を解析しようとすると失敗してしまう．未定義語を切り出し，構文解析が成功すれば，構文解析の後段階（意味解析）で外部記憶を参照するなどして未定義語の意味を補うことが可能である．

当研究所では，構文解析のアルゴリズムとして単語の切り分けをする段階から横型に探索する構文解析方法を用いている．この方法では単純に未定義語の抽出を行うと指数関数的に処理量が増大してしまう．そこでこの処理量の増大を防ぐ方法を考案したので報告する．

2．構文解析アルゴリズム

使用している構文解析アルゴリズムは，入力された文を左から１文字ずつ読み込んで処理を行う．解析は基本的にはボトム・アップに行われるが，現在必要とされている文法カテゴリーを目指して解析を行うことにより無駄な解析木の成長を抑えている．探索には横型探索が採用されている．

予め単語単位に解析を進めていないために分かち書きされていない文に対して単語の切り分け段階から横型に探索が行われる．

１文字毎に読み込んで解析を行うため，内部的には辞書と文法の区別がない．従って辞書びきという操作がなくなっているのが特徴である．

文法は分脈自由文法を採用している．辞書項目も文脈自由文法であるがその右辺のカテゴリーの単位は１文字となっている．つまり，「わたし」という名詞は，

名詞　→　わ　＋　た　＋　し

という文脈自由文法で表現される．（漢字を用いる場合は，漢字１文字が一つのカテゴリーとなる．）

3．未定義語の抽出

（１）前提

未定義語の処理を行うための前提として，次の２点を仮定する．

Ⅰ．　特別な機能を持つ単語は全て登録されているものとする．

Ⅱ．　構文解析の段階では，ある文字列が未定義語であることとそれがどの文法カテゴリーに属すかが分かれば良い．

Ⅰの仮定は，日本語の場合は助詞・助動詞や活用語尾などがこれに相当するが，これらは数が限られており，それらの全てを予め登録しておくことは実現上無理がない．Ⅱの仮定の意味することは，未定義語として抽出された単語がどの様な意味を持つかは意味解析のときに必要な情報であり，その時点で外部記憶にアクセスするなどすれば良いということである．（構文解析の時から意味情報を取り入れるべきだという意見もあるが，現在ではまだ構文解析と意味解析を分けて行うのが一般的であること，横型探索を行っているため意味情報を取り入れてもいずれ複数の解析結果が現れてしまうということを考慮して頂きたい．）

日本語の場合，機能語（助詞・助動詞など）から機能語までを１単語とみれる場合が多いのでⅠの仮定より単語の切り出しが出来る．更に節頭・節尾辞などを機能語として扱えばより正確に単語の切り出しが行える．

（２）基本アルゴリズム

未定義語を抽出するには，基本的には任意の文字列が全て未定義語である可能性があるとして解析を進めれば良い．具体的には，まず，次のような文法規則を用意する．

名詞　→　UNKNOWN　UNKNOWN　･･･　UNKNOWN　(3-1)

（UNKNOWNは１個からｎ個まで；つまり，１個以上のUNKNOWNの連続は名詞である．）

新たに入力された文字ａに対して，

UNKNOWN　→　ａ　(3-2)

という文法規則を作り出すと(3-1)(3-2)から入力さ

On Processing of Undefined Words in Parsing
Seiichi TAKAHASHI, Masashi OKUMURA, Atsushi ITOH, Tsukane OGAWA
CSK Research Institute

れた文字列は未定義の名詞の可能性もあるとして解析が進められる．

（３）改良点

このアルゴリズムでは，既に登録されている単語があるにも関わらず，未定義語の処理も同時に行ってしまう．つまり，単語が登録されているという利点が生かされていない．図１にこの様子を示す．

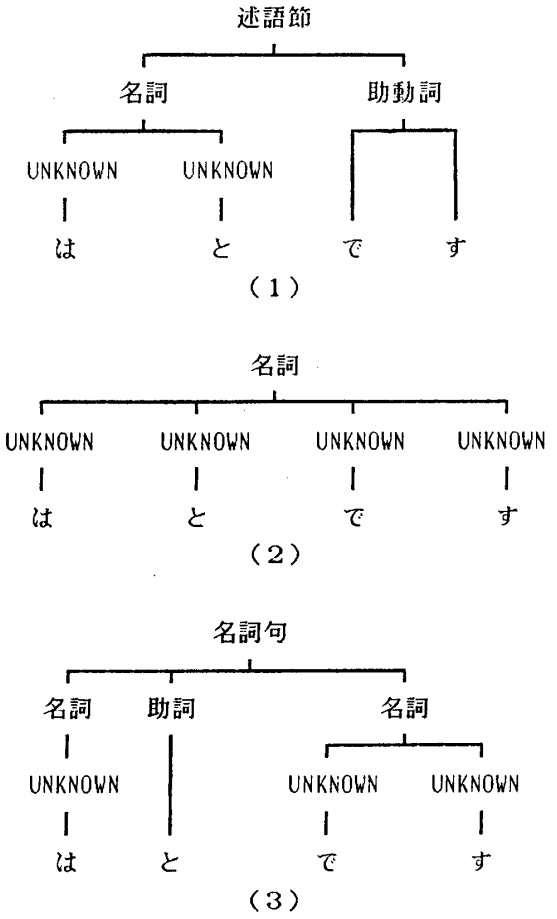

図１：「はとです」の解析例

「はとです」という文を解析すると，「はと」という単語が登録されているか否かに関わらず図１に示すような解析結果が得られてしまう．また，助詞・助動詞といったものも生かされていない．

これは，(3-2)の文法規則を常に作っているためである．そこで，ある時点で未定義語の文字列を作り出す作業を中止すれば良いことになる．ここでは，次の点で制御することにした．

Ⅰ．既知の単語（１文字の単語を除く）の一部として処理されている間は(3-2)の文法規則を作り出さない．

これにより，かなりの場合，無駄な未定義語の文字列を作り出さないで済むことになった．例えば，図１の例では（１）の結果のみが得られるようになった．しかし，この制限を加えたため未定義語が含まれているときに解析結果が正しく得られないことがある．これに付いて次節で述べる．

４．問題点

ある文字列が本来未定義語として処理されなければならないものが，その文字列が一度ある単語として決定されてしまうと，その単語を優先して解析が進む．そのため，次のような場合に正しい解析結果が得られなくなってしまう．

Ⅰ．未定義語となるべき文字列の一部が登録済みの単語であるとき．

例えば，「しろい」という文字列が含まれているとき，「しろ」（動詞の命令形，名詞）が登録されていると「白い」（形容詞）という未定義語とは解釈されない．

Ⅱ．未定義語として確定した文字列が含まれるとき．

例えば，「かわいい」という単語が未定義である場合には，先に「かわい」が形容詞として決定されてしまい「可愛い」という形容詞とは解釈されない．

但し，ここに挙げた例は文法規則の作り方によって変わる場合がある．

５．おわりに

今回説明したアルゴリズムにより辞書に登録されている単語が十分に生かされ，処理量が減少している．処理量は構文規則に非常に依存するため一概には言えないが，当研究所では実用上問題ない程度になっている．

今回は処理量の増大を防ぐことを最優先したため前節で述べたような問題点が生じた．この問題点は，利用する構文規則によるところが大きい．従って今後は構文規則と絡めて未定義語の処理を考えていくつもりである．

参考文献

[1]田中・佐藤・元吉：自然言語処理のためのプログラミング・システム　－－拡張ＬＩＮＧＯＬについて－－，電子通信学会論文誌，60D-12，1977．

[2]井佐原・元吉・田中：Ｎ進木拡張ＬＩＮＧＯＬのユーティリティ関数について，電子技術総合研究所い報46-12，1982．

[3]元吉・井佐原・石崎：日本語用完全横型探索構文解析法，情報処理学会第３２回全国大会4S-3，1986．