日本語係り受け解析の高速化手法

中山拓也　松本裕治
奈良先端科学技術大学院大学 情報科学研究科
{takuya-n,matsu}@is.aist-nara.ac.jp

1 はじめに

日本語係り受け解析は，粗い構文情報を抽出するための解析処理として，さまざまな利用方法が研究されている．その利用分野によっては(より高度な自然言語処理の為の前処理や情報検索など)，形態素/文節解析で生じる曖昧性は，より高いレベルでの解析によって解消可能であったり，必ずしも解消する必要がない場合がある．しかし，これまで提案された係り受け解析システムでは，形態素解析で生じる曖昧性を保持したまま解析できるものは少なく，常に低いレベルでの曖昧性解消処理に頼らざるを得ないため，問題となる場合がある．一般的に，低いレベルでの曖昧性を保持することは，解析速度を遅くさせる要因となる．しかし，その問題をクリアできれば非常に有用な解析器となると思われる．

以上のような状況を考慮し，形態素解析の段階で生じる曖昧性をいくらか保持しつつ，かつ十分高速に係り受け解析するための解析器を作製した．

2 係り受け解析

本稿で述べる係り受け解析とは，以下の処理を包括する．

1. 形態素解析
2. 文節生成
3. 係り受け可能性チェック

これらを一貫して処理することにより，一般的な形態素解析器の結果を用いる場合よりも高速に係り受け解析をすることが目的である．また，各段階において曖昧性を許容する．

出力は，図2のようなマトリックスとする．一般に係り受け解析の出力は，係り受けの曖昧性をヒューリスティクスなどによって解消したものであるが，ここでは，係り受けの曖昧性解消までは扱わない．より高度な言語処理の為の前処理として用いる場合など，係り受け解析器の応用によっては，必ずしも一意的な結果を必要としないのが，その理由である．

3 高速化のための手法

3.1 連接チェック

一般的な形態素解析では，品詞の連接可能性をチェックしながら解析をする．ここで，連接可能な場合には次の2種類ある．

1. **強い連接**
   連接する品詞Aと品詞Bが文節を構成する場合．(例) A:名詞 + B:格助詞
2. **弱い連接**
   連接する品詞Aと品詞Bは別々の文節に属する．(例) A:副詞 + B:名詞

ここで，強い連接は文節生成に利用される情報と等価であり，弱い連接は特に意味がなく使われるか，連接コストの設定によって文節数最小法などのヒューリスティックスをエミュレートするものである．つまり，前者は文節生成過程において同等の処理を行い，後者は，最適文節区切りの検索段階で同等の処理を行うことが出来る．形態素解析器の出力を利用して文節生成/係り受け解析をする場合，これらの処理が重複することになる．

そこで，形態素解析段階では連接チェックをせずに，文節生成の段階でこれらの処理を行うことによって，処理時間が短縮できる．

3.2 連接チェックでの文節情報の利用

形態素の連接チェックを文節生成の段階で行なうことを前節で述べたが，それにはもう一つの利点がある．それは，連接チェックに文節情報を利用できる点である．一般的な形態素解析器の連接チェックでは，隣接する形態素情報のみから連接の可能性を判断する (すなわち bi-gram 情報しか使えない)．しかし，文節情報が利用できれば，bi-gram の範囲を超える情報を利用して解析できることになる．

例えば,「研究所にのみ (蚤) を売る」という文を茶筌 で解析すると次のようになる．

研究所	普通名詞
に	格助詞
のみ	副助詞*
を	格助詞
売る	動詞-子音動詞ラ行-基本形

これは，bi-gram モデルでは 「のみ-を」の連接チェックの段階で,「に-のみ」という連接が存在するという情報を利用できないために,「のみ (副助詞)」という可能性を排除できないからである．この例のような場合，形態素解析結果の最適候補のみを利用して解析する係り受け解析器では，対処不可能な問題となってしまう．しかし，ここで文節情報が利用できれば，次のように処理することが出来る．

文字列	終端形態素	文節
研究所	名詞	名詞句
研究所-に	格助詞	格助付名詞句
研究所-に-のみ	副助詞	格助付名詞句
のみ	名詞	名詞句
のみ-を	格助詞	格助付名詞句
研究所-に-のみ-を	(接続不能)	
	...	

ここでは，既に格助詞を構成要素としてもつ名詞句 (格助付名詞句) であった場合，副助詞の後に格助詞が付かないとしている．そのため,「研究所にのみ」の後の「を」が強い連接によって「研究所にのみを」という文節を構成することが防げる．これにより，形態素解析を分離した係り受け解析器では解決困難な問題に対処できるのみならず，曖昧性を減らせるので，解析速度の向上が期待できる．

3.3 解析単位

茶筌[7], JUMAN[8] などの形態素解析器では，解析の単位は形態素である．例えば,「(…し) なければならない」は,「(…し) / なければ/なら/ない」と解析されるが，ここで，形態素という単位をそのまま解析の単位として利用するのは得策でない．上の例の場合，辞書のエントリーとして,「なければならない」という見出しで次のように登録できれば,「なければ」「なら」「ない」の連接のチェックを省略することが出来る．

【なければならない】

なければ	形容詞性述語接尾辞
なら	動詞-未然
ない	形容詞性述語接尾辞

このように，この見出しを解析単位とすることで，形態素 (解析単位) 数最小法と組み合わせれば，解析時間の短縮が可能である．また，辞書のエントリーには「なければ/なら/ない」という形態素情報が含まれるため，上記の見出しで解析したとしても，従来と同様の出力が可能である．

なお，上記と同等の操作は JUMAN では連語規則[3]，茶筌 では variable-gram モデル[2]で実現されている．しかしながら，これらは形態素単位で解析される結果を補正するという方針を採っており，本節で述べた解析単位を形態素と独立に考える方法とは根本的に方針が異なる．

3.4 曖昧さのパッキング

曖昧性を許容した解析をする場合，各段階で解消できる可能性のあるもののみを展開し，その他の曖昧さはパッキングして処理することが，高速な処理をする上で重要である．

例えば，格助詞「と」と引用助詞「と」の違いは,「と」の前が終止形や命令形の動詞ならば，連接チェック (文節生成) の段階で格助詞でないと判断できる．しかし「名詞+と」で構成される文節の場合は，どちらも用言句を修飾する文節 (後置詞句) であるため，係り受け可能性をチェックする段階であっても曖昧さを解消できない．このような場合，連接チェックの段階では格助詞と引用助詞の両方を展開しておき，係り受けチェックの段

階でそれらの曖昧さをパッキングできれば，時間的に効率良く処理できる．

係り受け解析において曖昧さを解消できる場面は，連接チェックの段階，係り受け可能性のチェック段階，文節の最適区切りの推定段階が考えられる．この内，前者2つの段階において解消できる可能性のある文節/形態素属性はそれぞれ，次のようになる[1]．

連接チェック段階

連接特性の異なる形態素の曖昧性解消．

係り受け可能性チェック段階

文節の主辞的な属性 (用言句，体言句など)．特性の異なる係り属性 (用言句に係る，体言句に係る，など)

ここでは，連接チェック段階においても文節の主辞属性を用いるので，文節主辞属性に加え，各段階でそれぞれ形態素，係り属性のみを展開すればよいことになる．

3.5 字種情報の利用

連続する片仮名列や数字列，英字列を区切り，別々の文節として扱わなければならないことはまずない．そこで，それらの列の途中での辞書の検索を省略すれば，辞書引き処理を高速化できる．また，片仮名列の品詞の多くはサ変名詞，英字列や数字列は名詞や数字として扱えばよく，それらを辞書に登録せずに自動的に品詞情報を付加すれば，辞書のサイズを減らすことができる．

3.6 曖昧性を許容した係り受け可能性チェック

曖昧性を許容した文節のラティスから，無駄なく係り受け可能性のチェックをするために，ここでは以下のような手順で処理している．

1. 可能な文節パスを探索する際，文末に接続する文節から深さ優先でパスを探索し，番号 (ID) を順に付与しておく．(例:図 1)
2. 同様の順序で，係り受け可能性を調べる．この際，通過した文節の番号をスタックに保持しておく．文節は自身より文末寄りの文節にしか係りえないという仮定をおけば，その文節の位置までにスタックに積まれた文節番号の文節が係り先の候補となる．ただし，例えば 1–7–3 と調べるとき，3 → 1 への係り受け可能性は，1–2–3 のパスを調べる際に既に調べているはずであるので無駄な手続きとなる．そのような無駄なチェックを避けるために，以下の項目に従って解析する．
   - その文節位置での最初の係り先チェック処理である場合は，スタック中の全ての文節を対象に係り受け可能性をチェックする．
   - 2回目以降の係り先チェック処理である場合は，スタック中で自身の文節番号よりも値の大きい番号の文節のみをチェック対象とする．
   - 2回目以降の係り先チェック処理である場合は，スタックに自身を積まない．

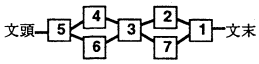

図 1: 文節番号の付与順序

4 実装

前述のアイディアを盛り込み，日本語係り受け解析器 *fdx* として実装した[2]．*fdx* では，形態素，文節 (種類および生成規則)，係り受け特性のすべてをユーザが自由に設定する．辞書未登録語に対する文節生成 (例: 名詞句 → 未登録, 格助詞) も可能であるため，漢字を多く含む文章では付属語だけの登録でも，ある程度の解析ができる．そこで，付属語と少量の自立語の辞書と文節生成規則を用いたところ，表1のように，ある程度の高速処理性能を持つことが確認できた[4]．辞書や出力が異なるので単純な比較はできないが，参考として QJP[1] と茶筌での解析速度も併記しておく．

解析速度や精度は辞書や規則の定義に依るところが大きい．今回は語長の短かい付属語を主に登録しているが，より長い語長のエントリーを登録すれば，さらに高速化が期待できる[3]．また，精度に関しては，図2のように漢字を適度に含む文章ならば自立語の登録は少なくても良いが，平仮名

[1] 最適区切りの推定の段階については，文節数最小法を用いて解析するため，曖昧性の表現方法には特に関係しない

[2] 実装には C++ を用いた．

[3] 曖昧性が高くなるような登録があれば解析速度は落ちるが，全体的に見れば速度向上が見込まれる．

の多い文では誤り易い.

表 1: 解析の速さ [4] (bytes/msec.)

	fdx	*fdx*(表示なし)	QJP	茶筌
A	32.7	68.1	13.1	20.4
B	25.6	50.0	14.8	22.1

```
1 1 1 0 0 1 1 1 0 1 1 1 (体言句-連用|連体)-収益性の
1 1 1 0 0 1 1 1 0 1 1 (用言句-基本|連体)-低い
1 1 1 0 0 1 1 1 0 1 (体言句-連用|連体)-事業の
1 0 1 0 0 1 0 0 0 (用言句-連用)-見直し
0 0 0 0 0 0 0 0 0 (句読点)-、
1 1 1 0 0 1 1 (体言句-連用|連体)-不良資産の
1 0 1 0 0 1 (体言句-連用)-償却を
1 0 1 0 0 (用言句-連用)-一層進め
0 0 0 0 (句読点)-、
1 0 0 (判定詞句-連用)-高収益体質に
1 0 (用言句-連用)-高収益体質に
1 (体言句-連用)-高収益体質に
(用言句-連体)-変えなければならない
```

図 2: 係り受けマトリックスの出力例

5 関連研究

広く知られている日本語係り受け解析器には KNP[5] や QJP がある. KNP は JUMAN の出力結果を利用するもので, 速度的に JUMAN での形態素解析にかかる時間を下回ることはない. QJP は, 高速性に加えて軽量化 (使用メモリを出来る限り押える) を目指した解析器であり, それらのトレードオフによって速度を犠牲にしている面があるようである. どちらも, 形態素の曖昧性は形態素解析段階で解消してしまっているので, 一つの閉じたシステムとして見れば問題ないが, より上位の処理 (意味解析など) との組み合わせを考えたとき, やや不満が生じる.

形態素 (文節属性) の曖昧性を保持して解析するものとしては, 秋山ら [6] の研究がある. 形態素解析結果を入力とする点などに不満を残すものの, 文節の曖昧さを残しての係り受けの曖昧性の解消方法は参考になる. 形態素解析で生じる曖昧性をより高い段階の処理情報を用いて解消する方法は, 伴光ら [4] の研究などで有効性が示されている.

[4]表の数値は, SUN Ultra-1 上での計測結果. テストに使用したデータは, それぞれ A: 講談社和英辞書の例文 37,584 文 (約 1.4M bytes, 一文平均 37.8 bytes), B: 日経新聞記事 55,352 文 (約 5.5M bytes, 一文平均 99.5 bytes).

6 まとめ

本研究は, 形態素解析/文節区切りの曖昧性を保持しつつ, かつ高速に係り受け解析をする処理系を開発し, 自然言語の高速処理が必要となる応用分野へ適用することを目的とする. そのため, 処理系の出力は係り受けの曖昧性を保持し, その解消は対象としていない. 本稿では, 高速化のために利用できると思われる幾つかの手法について述べ, それらを実装した解析器 *fdx* において十分な高速性が得られることを確認した.

今後の課題としては, 辞書や文節生成/係り受け規則の充実が挙げられる. 現段階では主に機能語のみの辞書構成であるが, 茶筌等の辞書を利用することで辞書記述を充実させる予定である.

また, 形態素/文節レベルの曖昧性を保持すると, 特に長文において, 文節パスの曖昧性の爆発的増加を引き起こす可能性が高い. 現段階の *fdx* では文節パスの上限数を設けることで対処しているが, さらに, 文節に対するコストを与えることで優先順位を設定できるようにする予定である.

参考文献

[1] Masayuki Kameda. A Portable & Quick Japanese Parser: QJP. In *COLING-96*, Vol. 2, pp. 616–621, Aug 1996.

[2] 北内啓, 山下達雄, 松本裕治. 日本語形態素解析システムへの可変長連接規則の実装. 言語処理学会第 3 回年次大会発表論文集, pp. 437–440, 1997.

[3] 山地治, 黒橋禎夫, 長尾眞. 連語登録による形態素解析システム juman の精度向上. 言語処理学会第 2 回年次大会発表論文集, pp. 73–76, 1996.

[4] 伴光昇, 福田譲, 白井清昭, 田中穂積. 圧縮統語森上での形態素解析候補の絞り込み –品詞列統 計 情報の利用–. 人工知能学会全国大会 (第 8 回) 論文集, pp. 527–530, 1994.

[5] 黒橋禎夫. 日本語構文解析システム KNP version 2.0 b3 使用説明書, 1997.

[6] 秋山典丈, 藤井敦, 徳永健伸, 田中穂積. 形態素で残る曖昧性を考慮した日本語文の係り受け解析. 言語処理学会第 1 回年次大会, pp. 129–132, 1995.

[7] 松本裕治, 北内啓, 山下達雄, 平野善隆, 今一修, 今村友明. 日本語形態素解析システム『茶筌』 version 1.5 使用説 明書, 1997.

[8] 松本裕治, 黒橋禎夫, 山地治, 妙木裕, 長尾真. 日本語形態素解析システム JUMAN version 3.2, 1997.