TRIE構造とグラフスタックを持ちいた日本語形態素解析

1F-4

中嶋章子　　杉村領一
松下電器産業(株)情報システム研究所

1 はじめに

自然言語処理に用いる辞書構造の一つとしてTRIE構造が有用であることがロジックプログラミングの分野などで知られている[3][4]. また，かな漢字変換や，並列処理用の解探索手法の研究成果の一つとして，局所的な曖昧さをパックする手法が知られている[2][5].

これらの手法を実用レベル規模の辞書を持つ形態素解析に適用した場合にはどのような工夫が必要になり，どの程度の性能を持つのだろうか. 手続き型言語による実装および評価を行ったので報告する.

2 TRIE構造の実装

TRIE構造辞書は，例えば図1のような辞書エントリの文字列を，図2のような文字列の左端を共有した構造へ変換して得られる.

ーーエントリーー	品詞（参考）
あらかた	副詞
ありがた	形容詞語幹
ありがたみ	名詞
ありがためいわく	名詞
ありがとう	間投詞

図1: 辞書エントリ例

```
あ＋ら－か－た
  ＋り－が＋た＋み
          |  ＋め－い－わ－く
          ＋と－う
```

図2: TRIE構造例

[1]の指摘にもあるが，TRIE構造をそのまま文字列検索用の辞書構造として用いると，例えば図2の下線を施した文字列のように，構造的には枝分かれしないにも関わらず，枝分かれを想定した構造ができる. これらの部分については，TRIE構造を1文字毎に用意するため，辞書のサイズが冗長になり，検索時間にも若干の無駄が生じる.

そこで，TRIE構造化の不要な文字列部分を圧縮した図3のような構造へ辞書を変換し，TRIE構造辞書のサイズ，および，これを用いた検索の効率化を図った.

```
あ＋［らかた］
  ＋［りが］＋た＋み
            |  ＋［めいわく］
            ＋［とう］
```

図3: TRIE構造圧縮例

また，大容量の辞書の場合，この全てのTRIE構造を主記憶に持つことは不可能であり，二次記憶装置への効率の良い実装[6]が不可欠となる. そこで，第一文字を共有する図3のような辞書エントリをまとめてTRIE構造化して二次記憶装置へ格納し，このまとまりへのポインタだけを主記憶に常駐させた.

以上の方式で4万語の辞書を実装した場合，改良以前に比べて辞書容量が約1割減少した. 検索時間については，これが二次記憶からのTRIE構造読みだし時間の誤差範囲になっているため，圧縮する以前に比べて差は殆ど出なかった.

3 グラフスタックを用いた形態素解析

今回実装したアルゴリズム[5]は，接続行列を用いた3型(正規)文法に基づいて，横型の全解探索を決定的に行う. 検索結果の表現にパッキングを実現する接続ベクトル[2]を用いているが，文節の拡張は行わず，文節数最少法などのヒューリスティクスを用いることを可能にしている.

3.1 グラフスタックの形式

TRIE構造を用いて辞書検索を行うと，例えば文字列「計算機」を検索する過程で，動詞語幹「計」とサ変名詞「計算」を検索できるように，ある文字に始まる全ての形態素を自然に検索できる.

解析は，この検索機能を用いて入力文字列の左端より形態素を検索することから始める. 検索結果は，順次表1のようなテーブルに登録する. テーブルの各エントリは，図4のような意味を持っている.

Japanese morphological analyser with TRIE structure dictionary and graph stack for local ambiguity packing
Akiko Nakajima and Ryoichi Sugimura
Matsushita Electric Industrial Co.,LTD.

エントリ : N[C ; Li,...Ln ;S,M]

N : 形態素番号（検索された順に番号が打たれる）
C : 形態素表層
Li,...,Ln: 左方に接続する形態素番号
＊のついた形態素が最少文節列を構成
S : 左方総文節数の最少値
M : 形態素の含まれている文節内の形態素総数

図 4: 解析テーブルのエントリ

文頭からの検索が終了し例のように1,2,3,4の形態素を登録すると，つぎに文字位置2からの検索を行う．そして，例えば形態素5を見つけると左方の形態素 1,2 との接続判定を行う．接続可能な場合には，その番号を左方に接続する形態素として登録する．これにより，接続ベクトルが実現され曖昧さがバックされる．

一般に漢字かな混じり列を形態素解析した場合でも，辞書の規模が膨大になるほど，曖昧さが出る．そこで曖昧さを縮退するヒューリスティクスである文節数最少法と形態素数最少法を用いるために左方総文節数の最少値 S と処理中の形態素の含まれる文節内の形態素総数 M を形態素毎に計算しておく．計算の際に，接続可能な形態素が複数ある場合は，最少文節を構成する形態素に印 (*) をつける．

文字位置	表層	解析結果
0		0[文頭]
1	か	1[書;0;0,1],2[欠;0;0,1]
2	い	5[い;1*,2*;0,2],3[会;0;0,1]
3	て	4[買い手;0;0,1],6[て;5;0,3]
4	い	7[居;4,6;1,1],8[い;6;0,4]
5	る	9[る;7,8*;0,5]
6	。	10[。;9;1,1]

表 1: 解析テーブル例

以上の手法により「かいている」という文字列を解析すると，表1のような構造ができる．これは図5のようなグラフスタックを表現している．図中 + は形態素の切れ目を示し，/ は文節の切れ目を示す．

入力文字：か い て い る 。

```
文頭＋書＋い＋て＋い＋る＋。＋
    ＋欠＋          |
    ＋－会－＋      |
    ＋－買い手－／居＋
```

図 5: グラフスタック

3.2 最尤解選択処理

図1の結果から文節数最少法と形態素数最少法による解析結果を得るには，解析結果を文字列の後方から前方へ辿り，例えば，形態素9のように前方のパスが二つ以上ある場合には＊の付いた形態素を辿れば良い．本処理は入力文字列長さにたいして比例した時間 (線形) で終了する．例の場合，最尤解は「(欠 or 書) いている」である．

4 実験結果

以上の手法に基づいた形態素解析を実装した実験結果を示す．解析時間は，長さ1から10までの形態素が入力文字の全ての部分で見付かり，かつ，全ての形態素が接続する場合をワーストケースとして試験的に出した．

計算機	SUN－3
OS	UNIX
実装言語	C
解析時間	100ms／50文字 以下
辞書エントリ数	40,000 語

5 おわりに

TRIE 構造辞書およびグラフスタックを用いた形態素解析手法が若干の改良を加えることにより，実用レベルで十分な性能を持つことを確認した．今後の課題として解析時間の大半を占める辞書読み出しを，キャッシュを用いて減少させる方法について検討したい．また，辞書サイズや検索時間の理論的な整理も行いたい．

参考文献

[1] 青江順一，自然言語辞書の検索（ダブル配列による高速デジタル検索アルゴリズム）*bit, Vol.21, No.6* pp776–784, 1989

[2] 大河内正明，分かち書き方式仮名漢字変換のためのバックトラックを必要としない文法解析，情報処理学会論文誌, *Vol.24, No.4* pp389–396,1983

[3] 上脇正, 田中穂積，辞書の TRIE 構造化と熟語処理，*Proceedings of the Logic Programming Conference '85*, pp329–340,1985

[4] 杉村領一, 赤坂宏二, 久保幸弘, 松本裕治, 佐野洋，論理型形態素解析 LAX，*Proceedings of the Logic Programming Conference '88*, pp213-222,1988

[5] R.Sugimura, Y.Kubo, Y.Matsumoto Logic based lexical analyser LAX, *Lecture Notes on Computer Science (to be appeared)*, 1989

[6] 日高達, 稲永紘之, 吉田将，拡張 B-tree による日本語単語辞書の作成，情報処理学会自然言語処理研究会資料 *33*,1982