接続情報を加味した形態素辞書による形態素解析の高速化

1R－3

安藤一秋，柏木雄一郎，獅々堀正幹，青江順一
徳島大学工学部　知能情報工学科

1．はじめに

一般的な形態素解析の処理は，辞書検索と接続検定，最適解探索で構成される．日本語は，単語間に区切りを持たない膠着語であるため，辞書検索時に入力文の各位置から始まる全ての形態素を検索する必要がある．また，辞書検索により得られた全ての隣接する形態素に対し，接続表や接続確率，接続コスト等を用いた接続検定を行わなければならない．これらの処理は解析時間の大部分を占有しているため，これらの速度向上は形態素解析全体の高速化につながる．

標準的な辞書検索手法は，トライを用いる手法である．しかし，トライを用いた場合，入力文に含まれる全ての可能な形態素を検索するために，入力文の各位置から検索を繰り返す必要がある．従って，一度読みこまれた文字が再度読みこまれるため検索速度が遅くなる．この欠点を改良するために Aho-Corasick 法（AC 法）[1][2]や決定性オートマトン[2]などを用いる高速化手法[1]が提案されている．

一方，接続検定の計算コストは品詞体系に依存する．特に，かな漢字変換のように品詞が細分化される分野では，辞書検索によって得られる形態素候補の数が増加するため，接続検定のための計算コストが増加する．しかし，効率的に接続検定を行う手法に関しては，あまり議論されていない．

本稿では，形態素解析の高速化を実現するため，辞書検索と接続検定に着目した．辞書構造として AC 法を採用し，辞書構築時に隣接する形態素間で予備的な接続検定を行うことにより，形態素辞書と接続情報を融合させる手法を提案する．提案手法の導入により，高速な辞書検索と効率的な接続検定が同時に行える．

2．AC 法を用いた形態素辞書

ここでは，提案手法の核となる AC 法[1]を説明する．

K を形態素の記号列 $y_1, y_2, \cdots y_n$; $n \geq 1$ を要素とする有限集合とする．AC 法は K に属する形態素 y およびその形態素情報を任意の入力記号列の中から検出する．但し，ここで形態素記号列は互いに重複することも許す．AC 法は状態集合 S で構成され，各々の状態は整数値で表す．特に初期状態を 0 で表す．入力記号の集合を I とするとき，AC 法の動作は次の 3 つの関数によって制御される．

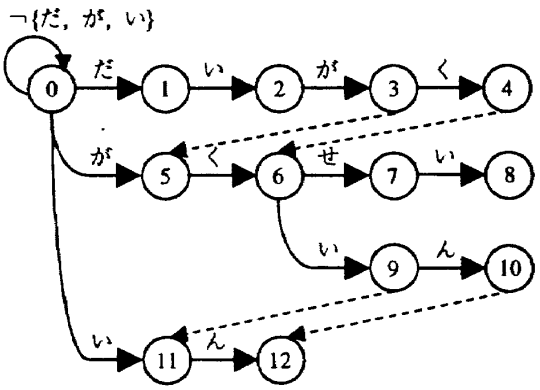
goto 関数　$g : S \times I \rightarrow S \cup \{fail\}$

failure 関数　$f : S \rightarrow S$

output 関数　$out : S \rightarrow$ 形態素情報

goto 関数は，状態と入力記号から状態への写像で，任意の状態からの任意の入力に対する遷移先を示す．*failure* 関数は，状態から状態への写像で *goto* 関数が失敗（g=*fail*）したときの遷移先を示す．*output* 関数は，状態から形態素情報への写像で，初期状態からの経路に含まれる部分文字列を表記としてもつ形態素情報を出力する．一般的には，文字長，品詞番号，頻度などが出力される．

図 1 に K＝{だい（大），だいがく（大学），がく（額），がくせい（学生），がくいん（学院），いん（印）}に対する AC 法の辞書構造（AC 辞書）を示す．但し，ここでの形態素情報は表記のみである．また，破線は *failure*



(a) *goto* 関数

out(2)={大}, *out* (4)={大学，額}, *out* (6)={額},
out (8)={学生}, *out* (10)={学院，印}, *out out*(12)={印}

(b) *output* 関数

図 1　K に対する AC 辞書

A High Speed Morphological Analysis Using the Morphological Dictionary Containing the Grammatical Connectivity
Kazuaki Ando, Yuichirou Kashiwagi, Masami Shishibori, Jun-Ichi Aoe
The University of Tokushima
2-1 Minami-Josanjima-Cho, Tokushima-Shi 770-8506, Japan

関数を示し，¬{だ，が，い}は，「だ」，「が」，「い」以外の記号を表す．この AC 辞書に対して，「だいがくせい」が入力として与えられた場合，$g(0,だ)=1$，$g(1,い)=2$，$out(2)=\{大\}$，$g(2,が)=3$，$g(3,く)=4$，$out(4)=\{大学，額\}$，$g(4,せ)=fail$，$f(4)=6$，$g(6,せ)=7$，$g(7,い)=8$，$out(8)=\{学生\}$のように状態遷移と出力を行う．

以上示したように，AC 法は入力文を一回走査するだけで全ての可能な部分文字列を検索できるため，トライを用いた手法よりも高速である．また，構築に関してはキーワード長の総和に比例する．

3．接続情報の導入

任意の状態の *output* 関数に含まれる形態素の集合を X，その状態から最終状態†までの *output* 関数に含まれる X の後接形態素の集合を Y とする．

図 2 に示す「たべすぎ」の例を用いて接続検定を考える．X＝｜た（24 種）｜，Y＝｜べ（2 種）｜の場合，「た」×「べ」＝24×2＝48 回の接続検定が必要である．しかし，全ての「べ」が「た」と接続しないため‡，この接続検定は無駄となる．同様に，「たべ」×「す」＝33，「たべ」×「すぎ」＝13 に対しても接続が成立しないため，合計 94 回の無駄な接続検定が行われたことになる．従って，形態素解析の際に，このような検定を省略できれば形態素解析の計算コストを削減できる．

そこで，我々は AC 法の特徴に着目した．AC 法には *failure* 関数が存在するため，初期状態から任意の状態までの経路に部分文字列が含まれる．これは，初期状態から最終状態までの全ての隣接する形態素間であらかじめ接続検定が可能であることを意味する．つまり，AC 辞書構築時に予備的な接続検定を行い，接続が成立しない形態素をあらかじめ調べておくことにより，解析時の接続検定回数を減少させることができる．また，AC 法の各状態は 1 入力多出力であるため，経路により出力が一意に決定できる．従って，AC 法の構造は接続検定の結果の保持に適している．しかし，全ての接続検定結果を辞書に登録することは難しいため，本稿では *output* 関数に含まれる形態素情報を次に示す不接続候補とそれ以外に分類する．以後，この情報を接続情報とよぶ．

定義：不接続候補とは，全ての $x \in X$ に対して接続が成立しない $y \in Y$ である．

† 次状態遷移が定義されていない状態

‡ 我々が作成したかな漢字変換用辞書と接続表を利用

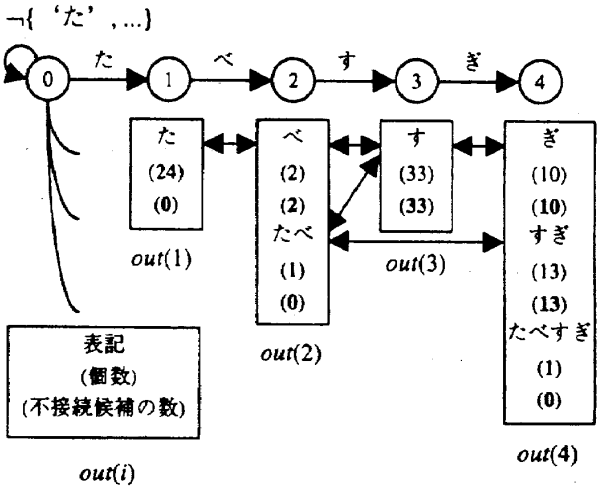


図 2　「たべすぎ」の *output* 関数と不接続候補

図 2 に「たべすぎ」の不接続候補を示す．「たべすぎ」に関して接続が成立する候補は，「た」と「たべ」「たべすぎ」のみである．これらの情報は，AC 法の *failure* 関数構築後に付加される．

4．予備実験

AC 辞書を形態素解析に導入するための予備実験として，不接続候補の割合を調べる簡単な実験を行った．使用した形態素辞書は，我々が作成したかな漢字変換用（形態素総数：100,053，品詞総数：486）である．また，AC 辞書の構築に要した時間は，約 3 分（PentiumII 300MHz）であった．実験の結果，AC 辞書全体の総 *output* 数は 5,637,007 で，不接続候補の数は，1,622,382 であった．これは，総 *output* 数の約 29%を占める．以上により，提案手法を形態素解析に導入することで，接続検定回数を減少できることが確認できた．

5．まとめ

本稿では，形態素解析の高速化を行うため，形態素辞書と接続情報を融合させ，辞書検索と接続検定の両面から高速化を行う手法を提案した．今後，AC 辞書を用いた形態素解析を実際に構築し，提案手法の有効性を確認する予定である．

参考文献

[1] A.V. Aho et al., "Efficient String Matching : An Aid to Bibliographic Search", Commu. of the ACM, Vol.18, No.6, pp.333-1996.

[2] 森，"DFA による形態素解析の高速辞書検索"，EDR シンポジウム，1997.