二つのトライを用いた自然言語辞書検索技法

5F−4

森本勝士　青江順一
徳島大学

1. はじめに

トライ[7]構造は自然言語辞書の検索を中心としてよく用いられているが，キーの数が多くなると状態数の増加に伴う記憶量の増加が問題となる．これに対し，トライの共通接尾辞を併合したＤＡＷＧ[3,9]が提案され，この問題を解決しているが，このＤＡＷＧではキーに対するレコード情報が一意に決定できない．そのため，その応用はレコード情報を必要としない分野（プログラミング言語処理系の指定語の検索，スペルチェックにおける一般辞書[9]など）に限られていた．またＤＡＷＧは静的なキー集合を対象としていたので，青江ら[2]はＤＡＷＧの動的な構成法を提案している．ＤＡＷＧ以外のトライの圧縮手法として，Kurt[8]の手法が存在するが，静的キー集合のみを対象としていた．

そこで，本稿ではレコード情報が一意に決定でき，しかも動的キー集合に適用可能なトライの圧縮法を実現するために，二つのトライを利用した新しい構造ダブルトライ(double-trie)を導入し，種々のキー集合に対する実験結果により，提案手法の有効性を実証する．

2. キー集合の記憶

2．1 トライ構造

以後，断りのない限り，トライ上で"*the*","*then*"のようなキーを区別するため，キーの最後尾に端記号[7]#を付する．図１にキー集合$K1$＝｛電気工学#，電子工学#，情報工学#，情報処理#｝に対するトライを示す．図中の二重丸で示される状態は，検索中のキーを他のキーと区別するのに十分な，最初の状態（セパレート状態と呼ぶ）を示す．トライでの検索時間はキーの登録数には無関係なので，大きなキー集合に対し，高速な検索手法となる．また，先頭から同じ綴（共通接頭辞）を含む語の間では，共通接尾辞はそれらの語全体で共有されている．しかし，一度分岐した遷移は再び併合されず，各単語間の最後尾から同じ綴（共通接尾辞）の部分は共有されないため，登録キーが増加するとトライの状態数が大きくなり，記憶量を圧迫する．

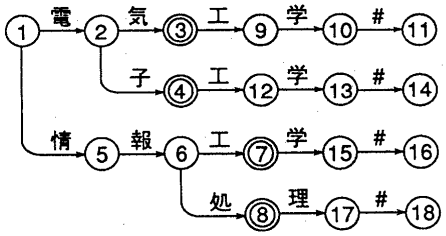

図１　$K1$に対するトライ

A Searching Strategy of Natural Language Dictionaries Using Two Tries
Katsushi MORIMOTO　Jun-ichi AOE
University of Tokushima

2．2 ＤＡＷＧ

$K1$に対するＤＡＷＧを図２に示す．$K1$のトライ（図１）とＤＡＷＧ（図２）により状態数は１８から９に減少していることが判る．ＤＡＷＧでは，↑を付してある状態６が全てのキーに対する最終（出力）状態となり，検索終了時の状態が複数のキーに対応するので，キーに対応するレコードを一意に決定できない．

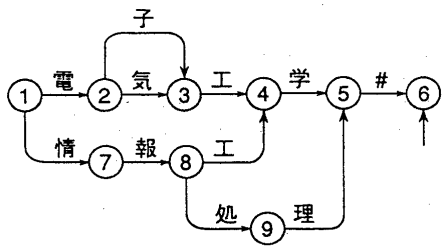

図２　$K1$に対するＤＡＷＧ

3. 二つのトライによる共通接尾辞の圧縮

本手法では，トライ上の共通接尾辞を併合し，状態数を削減する．そのため，図３に示すように，通常のトライのセパレート状態以降を取り除いたトライ（セパレート状態を含む：ＬＨトライ）と，セパレート状態以後の未併合の遷移を対象に作られたトライ（ＲＨトライ）の二つのトライを用いキー集合を構成する．但しＲＨトライは，葉から根の方向に検索を進め，根が各キーに対する検索終了状態となるように構成される．二つのトライを連結するために，ＬＨトライのセパレート状態から，ＲＨトライにリンク（破線で示される）を引く．「情報工学#」の検索は，ＬＨトライ上で状態１，５，６，７と遷移し，"情報工"を検索後，状態７からリンクをたどりＲＨトライの状態３に至る．ＲＨトライ上で状態３，２，１と逆向きにたどり，残りの"学#"が検索され，初期状態で検索成功となる．

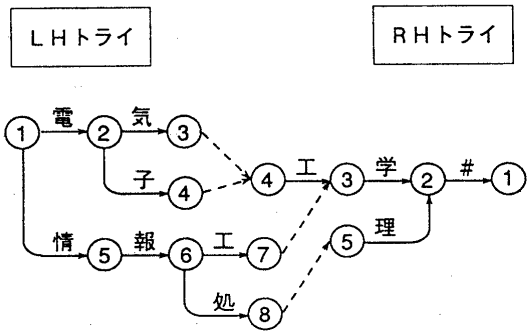

図３　$K1$に対する二つのトライ（ダブルトライ）

なお，ＬＨトライ上のセパレート状態は各キーと一対一に対応するので，各キーのレコード情報はこのＬＨトライのセパレート状態に対応させることにより，一意に取り扱うことができる．

４．　goto関数の構築

４．１．　ダブル配列

本手法では，コンパクト性と検索の高速性に優れたgoto関数のデータ構造として，青江[1]の提案するダブル配列(double-array)を用いる．ダブル配列は未使用要素を容易に解放できないため，頻繁に削除が行われるようなキー集合の検索には向かない．しかし，一般的な自然言語辞書のように，あらかじめ基本的なキー集合を構築しておき，後に使用者がキーを適宜追加して拡充させるようなキー集合に対する検索（準静的検索）には適している．

４．２．　データ構造

goto関数gは特定の状態上での特定の記号の検索結果を表す関数とする．検索木上で状態番号rにおける検索記号aが存在し，その先の状態番号がtならば，
$g(r,a)=t$と書き，rにおいてaが検索できないとき，
$g(r,a)$=failと書く．

以下，これを用いて遷移を$g(r,a)=t$の形で表現する．図１の例では$g(1,'電')=2$，$g(2,'情')$=failとなる．

ダブル配列法は二つの一次元配列BASEとCHECKの対で検索木を表現し，遷移$g(r,a)=t$を次の関係で実現する．

t←BASE[r]+a;
if(CHECK[t]=r) then 次の状態番号はt.
else 次の状態は定義されていない．

ここで，aは記号aに対する内部表現値(numerical value)を意味する．また，$g'(t,a)=r$で表される逆向きの遷移は次の関係で実現される．

r←CHECK[t];
if(BASE[r]+a=t) then 前の状態番号はr.
else 前の状態は定義されていない．

このようなデータ構造により，一つの遷移を検索する最大時間計算量(wrost-case time complexity)は前向き走査，後ろ向き走査共に常に$O(1)$となり非常に高速となる．

５．実験結果

本手法の構成システムは約1,500行のＣ言語で記述されており，Sparc Station2上で稼働している．

表１　実験結果

各値	K_3	K_4	K_M	K_E
キー総数	50,000	50,000	85,408	80,126
接尾辞の種類	6,289	12,025	9,546	799
シングル状態数				
通常のトライ	114,563	180,105	190,815	108,480
ダブルトライ	6,764	16,583	36,071	2,640
圧縮率（%）	94.1	90.8	81.1	97.6
総状態数				
通常のトライ	184,369	253,614	321,567	274,840
ダブルトライ	76,570	90,092	166,823	169,000
圧縮率（%）	58.7	64.5	48.1	38.5
時間（ミリ秒）				
検索	0.08	0.10	0.07	0.09
追加	73	136	50	113

表１に示す実験結果の各キー集合の要素は，K_3，K_4：各々３，４漢字からなるキー，K_M:形態素辞書用キー，K_E：英単語のキー，となっている．なお，実験はキー表記（#を含む）の記憶のみを対象とし，レコード情報の記憶は行っていない．

検索・追加時間は，各キー集合に対して，登録済みの全ての語の検索と，未登録の300語の追加を行った場合の１語あたりに要する時間である．

接尾辞（ＲＨトライ上に記憶される文字列）の記憶に必要な状態数（表１では"シングル状態数"で示される）は，通常のトライに比べ，約３～２０％と大幅に減少しており，共通接尾辞の圧縮が極めて効率的に行われている．またキー集合の構成に必要な総状態数は，通常のトライに比べ，約３５～６０％に減少しており，本手法の有効性が判る．さらに，検索は非常に高速であることが判る．また，追加時間も充分実用的な値と思われる．

６．　おわりに

本手法は，以下のような特徴を持つ．

(1)　各キーの共通接尾辞を共有することにより，トライの総状態数の増加が抑えられる．

(2)　ダブルトライは，各々のキーに対応するレコードを一意に取り扱うことができる．

(1)は大きなキー集合に対して，トライの総状態数の増加の問題を解決する．(2)はＤＡＷＧでできなかった各々のキーに対応したレコード情報の取扱いを可能にした．このように，ダブルトライは共通接尾辞の圧縮により，辞書をコンパクトに構築でき，自然言語処理の幅広い分野に利用できる．

今後の課題は，キーの追加時間短縮のためのgoto関数のデータ構造考案，ダブルトライ更新時における不要遷移の効率的な選出・削除法の考案等がある．

参考文献

[1] 青江順一：ダブル配列による高速ディジタル検索アルゴリズム，信学論(D),Vol.J71-D,No.9, pp.1592-1600(1988)

[2] 青江順一，森本勝士，長谷美紀：トライ構造における共通接尾辞の圧縮アルゴリズム，信学論(D-II), Vol.J75-D-II,No.4,pp.770-779(1992)

[3] Appel A.W. and Jacobson G.J.: *The world's Fastest Scrabble program, Comm.ACM*,Vol.31, No. 5, pp.572-578(1988).

[4] John A. and Dundas III: *Implementing Dynamic Minimal-prefix Tries, Softw.Pract.Exper*, vol.21(10), pp.1027-1040 (1991).

[5] 森本勝士，青江順一：トライ構造による複合語解析，情処秋期全大，2G-7(1991)

[6] 森本勝士，青江順一：トライ構造による形態素辞書と共起辞書の統合法，情処研報，91-NL-85-3(1991)

[7] Knuth D.E.: *The Art of Computer Programming*,3, *Sorting and Searching*, Ch.6 (1973).

[8] Kurt M.: *Compressed Tries, Commum. ACM*, 19,7, pp.409-415 (1976).

[9] Peterson J.L.: *Computer Programas for Spelling Correction, Lecture Notes in Comput. Sci., Springer-Verlag*, N.Y.pp.57-67(1980).

[10] 田中穂積：自然言語解析の基礎，産業図書(1989)