転置インデクスに適用可能な高速文字列あいまい照合アルゴリズム

5L－4

下村　秀樹
NECヒューマンメディア研究所

1. はじめに

テキストから任意の文字列を検索する技術は，全文検索などに広く使われている．また最近では，OCR後の誤りを含んだテキストを全文検索する文書画像ファイリングシステムなども商用になっている．このような誤りを含むテキストにまで検索対象が広がるにつれて，検索の高速性はもとより，多少の文字の不一致を許容して検索する「あいまい照合」の技術が求められ始めている．

大量のテキストを検索する場合の高速化技術の一つとして，予め作成した文字の出現位置リスト（転置インデクス）を使った方法がある[1]．この方法では，検索文字列に含まれる文字に対応する転置インデクスだけを処理すればよいので，照合処理に必要なデータの転送時間，および照合処理自身の時間が短くなるという利点がある．しかし，転置インデクスを使った照合処理において，一部文字の不一致を許したあいまい照合を効率よく行える有効なアルゴリズムは提案されていない．

そこで，通常のテキストに対するあいまい照合の代表的アルゴリズムであるShift-ORアルゴリズム[2]を転置インデクスに対しても効率よく適用できるよう拡張した．提案するアルゴリズムは，検索文字列から得られる転置インデクスを整列して，ただ1回走査することにより，k回以下の挿入／削除／置換を許すあいまい照合を効率よく実施することができる．

2. Shift-ORアルゴリズム

今，長さnのテキストT中にある，長さmの検索文字列Pを，k回以内の文字の挿入／削除／置換を許容して検索することを考える．アルゴリズムの説明に必要な記号は次のとおりである．

・T=t[1]t[2]t[3]...t[n]：検索対象のテキスト．
・P=p[1]p[2]p[3]...p[m]：検索したい文字列．
・一致状態ビット列配列R[i,j] ($0 \leq i \leq n$, $0 \leq j \leq k$)：長さm（検索文字列と同じ長さ）のビット列の2次元配列．R[i,j]の先頭からxビット目が1ならば，テキストのi番目の文字で，検索文字列の先頭からx番目までの記号と，j回以内の不一致を許して照合が成功していることを示す．
・B(x)：先頭のxビットが1で残りが0である長さmのビット列．
・Sft(R)：mビットのビット列Rを右に1ビットシフトする演算（空いた先頭ビットには1を入れる）．
・Msk(c)：記号cが検索文字列に存在した位置が1で残りが0となる長さmのビット列．例えば，検索文字列'ababc'に対して，Msk('a')='10100'，Msk('b')='01010'，Msk('c')='00001'，それ以外は'00000'である．
・AND，OR：ビット列の論理積，論理和．

Shift-ORアルゴリズムは，次の漸化式に従ってテキストを1回走査しながらR[i,j]を計算し，そのmビット目が1になったことにより，検索成功を検出する[2]．

R[i,j] = B(j):(i=0のとき) -(1-1)
R[i,j] = Sft(R[i-1,0]) AND Msk(t[i])
:(i>0,j=0のとき) -(1-2)
R[i,j] = (Sft(R[i-1,j]) AND Msk(t[i]))
OR R[i-1,j-1] OR Sft(R[i-1,j-1])
OR Sft(R[i,j-1]):(i>0,j>0のとき) -(1-3)

3. 転置インデクスに向けたShift-ORの拡張

3.1 アイデア

転置インデクスを使った検索における文字列照合では，インデクス情報が元のテキストに対して飛び飛び，つまり不連続に得られる．この特徴が照合に不要なデータの転送をなくしているのであるが，この不連続性のため，テキストを逐次走査するShift-ORアルゴリズムをそのまま適用することができない．そこで，転置インデクスが元のテキストに対して不連続に得られても，飛ばされた位置に対する計算をスキップし，インデクスを1回走査することで照合処理を行うよう，Shift-ORアルゴリズムの拡張を考えた．次に示す式の展開により，それが可能であることがわかった．

3.2 計算処理のスキップ

今，テキストの文字位置x+1でMsk(t[x+1])=B(0)であるとして，R[x,j]からR[x+1,j]を求めることを考える．このとき，(1-2)式，(1-3)式は次のようになる．

R[x+1,0] = B(0) -(2)
R[x+1,j] = R[x,j-1] OR Sft(R[x,j-1])
OR Sft(R[x+1,j-1]) -(3)

(2)式でR[x+1,0]は求まった．次に，(3)式でj=1としR[x+1,1]を考えると，次のようになる．

R[x+1,1] = R[x,0] OR Sft(R[x,0]) OR B(1) -(4)

このとき，Sft(R)は演算の性質上必ず先頭ビットが1なので，(4)式は次のように簡単になる．

R[x+1,1] = R[x,0] OR Sft(R[x,0]) -(5)

またここでF(R)=(R OR Sft(R))という演算を導入すると，(5)式は次のように書ける．

R[x+1,1] = F(R[x,0]) -(6)

次に，R[x+1,z]=F(R[x,z-1])が成立すると仮定し，(3)式よりR[x+1,z+1]を求める．演算FとSftは適用順番の交換が可能かつ結合則が成立する．また(1-3)式でR[i,j]がSft(R[i,j-1])を論理和で含んでいるので(R[x,z] OR Sft(R[x,z-1]))=R[x,z]がいえる．したがって，次式となる．

R[x+1,z+1] = F(R[x,z]) OR Sft(R[x+1,z])]
= F(R[x,z]) OR Sft(F(R[x,z-1]))
= F(R[x,z]) OR F(Sft(R[x,z-1]))
= F(R[x,z] OR Sft(R[x,z-1]))

A high-speed approximate string matching algorithm applicable to the inverted-index
Hideki SHIMOMURA
Human Media Research Labs., NEC corp.

$= F(R[x,z])$ -(7)

以上により，数学的帰納法から，(8)式が導かれる．

$R[x+1,j]=F(R[x,j-1]):(j>0)$ -(8)

このように Msk(t[x+1])=B(0)なら，各 R[x+1,j]は R[x,j-1]だけに対する演算から計算できる．もし x+1 ～ x+d で Msk が B(0)であれば，(8)式を漸化的に適用して，R[x+d,j]までを計算することができる．$F_y(R)$を F を R に y 回適用する演算と定義すれば，次のようになる．

$R[x+d,j] = F(R[x+d-1,j-1])$

$= F(F(R[x+d-2,j-2])) = F_2(R[x+d-2,j-2])$

$= \cdots$

$= F_y(R[x+d-y,j-y])$ -(9)

すなわち，R[x+d-y,j-y]の値が既知であれば，その値に演算 F を y 回繰り返すことで，R[x+d,j]が計算できる．具体的には，位置 x における R が既知ならば，j ≧ d の範囲では y=d のときに該当する R[x,j-d]に対して d 回，また j<d の範囲では，y=j に該当する R[x+d-j,0]=B(0)に対して j 回，F を繰り返し適用すればよい．なお，$F_y(B(0))$は，先頭の j ビットが 1，すなわち B(j)という値になる．以上を整理すると次のようになる．

$R[x+d,j] = F_y(R[x+d-y,j-y]):(j \geq d$ の場合) -(10-1)

$R[x+d,j] = B(j):(j<d$ の場合) -(10-2)

この(10)式で，R[x,j]が既知でかつ，x+1 ～ x+d の区間で Msk(c)が B(0)の場合に，R[x+d,j]の値を，x+1 ～ x+d-1 までの区間の R の計算をスキップして，直接的に求める方法が示された．

3.3 転置インデクスへの適用

検索文字列に対応する転置インデクスがもとのテキストの位置 p_1 で得られ，次に p_2 で得られたとすると，p_1+1 ～ p_2-1 までの区間は，Msk が B(0)に相当する．そこで，$d=p_2-p_1-1$ とおけば，(10)式により $R[p_2-1,j]$に該当する値を $R[p_1,j]$から計算することができる．この $R[p_2-1,j]$により，(1)式で $R[p_2,j]$を計算すればよい．この結果，照合時間は検索文字列に対して得られた転置インデクスの数に比例する．

図 1 に，提案アルゴリズムによる照合処理過程を示す．i は転置インデクスの番号，pos はテキストでの位置，chr はその位置の文字，tmp[j]は，該当位置の R を計算する際の作業変数で，さきに述べた $R[p_2-1,j]$に相当する．この例では，i=4 と i=8 のとき，不一致文字数 2 で，照合に成功している．なお，転置インデクスは元のテキストの出現位置によるソートが必要である．

4. 実験

提案した方式の効果を調べるために，べた書きテキストに対する Shift-OR[2]による検索（従来方式）と提案方式の処理速度を比較した．実験のテキストには，日本語特許（20M バイトのファイル）を用いた．検索文字列として「エンジン，正規分布，キーワード，特許明細書，ヒストグラム，音声認識処理」の 6 つ，許容する不一致文字数（距離）k は，0 ～ 2 の 3 種類を実験した．マシンは，NEC- EWS4800/460 を用いた．6 つの検索文字列の平均検索時間と提案方式に対する従来方式の時間比を表 1 に示す．時間の単位は ms である．

5. 考察

実験でわかったことを列挙する．

・検索にかかった合計時間を比べると，提案方式の方が大幅に高速である．28.1 倍（距離 0）～ 45.2 倍（距離 2）の高速化となった．

・項目別に見ると，検索処理のデータ転送時間は 19.7 倍高速化されている．転置インデクス利用によるデータ転送量削減の効果である．また照合時間は，76 倍以上も高速になった．検索文字列から得られた転置インデクスを 1 回だけ走査するので，字種の多い日本語に対してはこのような大きな効果となる．

・提案方式で，データの整列時間が全体に占める割合はかなり大きかった．距離が 0 の場合には，照合処理よりも多くの時間を要している．しかし，従来の照合時間と比べては桁違いに小さく，提案方式の有効性を失わせるものではなかった．なお，転置インデクスが文字別に出現位置順にソートされていれば，この整列はマージとなり，得られた転置インデクス数に比例する処理時間と推測できる．

表 1　検索処理速度の比較（20M バイトテキスト）

	Data 転送 (ms)	Data 整列 (ms)	距離別照合／合計(ms)					
			距離 0		距離 1		距離 2	
			照合	合計	照合	合計	照合	合計
従来	593.1	0.0	3223.8	3816.9	6054.3	6647.4	8637.2	9230.2
提案	30.1	63.5	42.0	135.6	79.3	172.9	110.5	204.1
比	19.7	0.0	76.8	28.1	76.3	38.4	78.2	45.2

T='adeabcddffabefcaefdd'
P='abaca'

i	pos	chr	Msk	tmp[0]	tmp[1]	tmp[2]	R[pos,0]	R[pos,1]	R[pos,2]
0	-	-	00000	-	-	-	00000	10000	11000
1	1	a	10101	00000	10000	11000	10000	11000	11100
2	4	a	10101	00000	10000	11100	10000	11000	11100
3	5	b	01000	10000	11000	11100	01000	11100	11110
4	6	c	00010	01000	11100	11110	00000	11110	11111*
5	11	a	10101	00000	10000	11000	10000	11000	11100
6	12	b	01000	10000	11000	11100	01000	11100	11110
7	15	c	00010	00000	10000	11010	00000	10000	11010
8	16	a	10101	00000	10000	11010	10000	11000	11101*

図 1　提案方式の照合処理例（*は検索成功）

6. おわりに

本稿では，文字列のあいまい検索を高速に行う Shift-OR アルゴリズムを，文字転置インデクスに効率よく適用する方式を提案し，その実験結果を報告した．今後は，検索対象テキストと検索要求文字列のバリエーションを広げて，さまざまな実験・検証を行いたい．

参考文献

[1]菊池他，日本語文書用高速全文検索の一手法，信学論，J75-D-I(9), 1992

[2]Wu et.al, Fast String Matching Allowing Errors, CACM, Vol.35, No.10, pp.83-91,1992