全文検索システム RetrievalExpress の開発と評価

福島 俊一　　赤峯 享
NEC　情報メディア研究所
E-mail: {fuku, akamine}@hum.cl.nec.co.jp

1　はじめに

昨今、情報洪水と言われるほど大量にあふれるテキスト情報を管理・活用するために、高速な全文検索技術が不可欠なものになっている[1]。

全文検索は、分類・選別された二次情報ではなく、一次情報であるテキスト全文を検索対象とする。検索条件語をテキスト全文と文字列照合(全文走査)することで機能は実現できるが[2]、大規模なテキストを対象に高速な検索レスポンスを達成するために、インバーテッドファイル(インデックス)を用いた方式の改良が進められている[1]。インバーテッドファイルの形式は、テキストからキーとして(A)単語、(B)文字列のいずれを取り出すか、キーに対応付ける位置情報として(ア)テキストIDのみ、(イ)テキストID＋オフセット(テキスト内位置)のいずれを格納するかの組み合わせによって、2×2＝4通りに大別できる。

しかしながら、それらタイプの違いによって、必ずしも振る舞いが同じでないことに注意を要する。ユーザにとって理解しやすい全文走査による検索結果を基準にすると、(A-ア)や(A-イ)のタイプ[3]は、形態素解析の失敗や曖昧性に起因する検索洩れが避けられない。また、(B-ア)のタイプ[4][5]は、キー文字列の位置関係を保存しないため、検索条件語を複数の部分文字列に分割して検索するケースにおいて、検索ノイズ(過剰ヒット)が発生する。検索洩れやノイズを生まず全文走査と等価な結果が得られるのは、(B-イ)のタイプ[6][7][8]である。(B-ア)に後処理として全文走査を組み合わせて検索ノイズを除去するアプローチ[4]もあるが、全文走査を用いる限り、安定して高速な検索レスポンスを得ることは難しい。

本論文で述べる全文検索システム RetrievalExpress では、フレキシブル文字列インバージョン法[8]を開発して、実装した。これは(B-イ)のタイプのインバーテッドファイルをベースとしており、全文走査と等価な(洩れやノイズのない)検索結果を保証する。さらに、位置情報データの読み出し量を削減する新しいパラメータ(字種別キー文字列長や縮退文脈など)を導入し、それを検索対象テキストの文字列統計に適合させることで高速な検索レスポンスを達成する。

本論文では、フレキシブル文字列インバージョン法をベースとした近接演算機能と検索結果ランキング機能についても紹介する。一般に、近接演算は全文走査と相性がよく、ランキングには単語をキーとしたインバーテッドファイルを用いることが多い[9]。それに対して、Retrieval Express では、(B-イ)のタイプのインバーテッドファイルを用いた近接演算とランキング処理を実現している。

2　フレキシブル文字列インバージョン法

(B-イ)タイプの最もナイーブな実装は、テキスト中の全文字(キー文字列長:1)に対して、その出現位置を記録するものである。例えば、図1のようなテキスト集合に対して、「情」の出現位置は#1:1、「報」の出現位置は#1:2 と#3:4 などを記録する(#n:m はテキスト#n の m 文字目を意味する)。検索時には、検索条件語を構成する各文字の位置情報をインバーテッドファイルから読み出し、その位置関係をチェックすることで検索条件語の出現を判定できる。このような実装では、インバーテッドファイルのキー文字列照合に要する時間はごくわずかで、位置情報データの読み出し時間の方が検索レスポンスを高速化する際のボトルネックになる。

キー文字列長を長くすることで、位置情報データの平均的な読み出し量は削減できるが、キーの種類はキー文字列長の累乗オーダーで増加するので、検索レスポンスとファイル容量との間に極端なトレードオフが生じる。

#ID	テキスト全文
#1	情報検索論を読む
#2	論文を検索する
#3	検索の報告文

図１　テキスト集合の例

キー文字列	縮退文脈	位置情報(#ID:offset)
す	(0,0)	#2:6
の	(0,0)	#3:3
む	(0,0)	#1:8
る	(0,0)	#2:7
を	(0,0)	#1:6, #2:3
検	(0,1)	#2:4, #3:1
検	(1,1)	#1:3
検索	(0,0)	#3:1
検索	(0,1)	#2:4
検索	(1,0)	#1:3
告	(0,0)	#3:5
告文	(0,0)	#3:5
索	(1,0)	#1:4, #3:2
索	(1,1)	#2:5
索論	(0,0)	#1:4
情	(0,0)	#1:1
情報	(0,0)	#1:1
読	(0,0)	#1:7
文	(0,0)	#2:2, #3:6
報	(0,0)	#1:2, #3:4
報検	(0,0)	#1:2
報告	(0,0)	#3:4
論	(0,0)	#1:5, #2:1
論文	(0,0)	#2:1

図２　主インバーテッドファイルの例

キー文字列	位置情報(#ID)
検	#1, #2, #3
索	#1, #2, #3
検索	#1, #2, #3

図３　副インバーテッドファイルの例

字種別キー文字列長		縮退文脈幅	
漢字	1, 2	検	2
片仮名	1, 2, 3	索	2
平仮名	1	検索	2
英字	1	(その他)	1
その他	1		

図４　パラメータテーブルの例

ナイーブな実装のとき:

情	#1:1	分割数: 5
報	#1:2, #3:4	データ数: 11
検	#1:3, #2:4, #3:1	
索	#1:4, #2:5, #3:2	
論	#1:5, #2:1	

キー文字列長の字種別可変化を行なったとき:

情報	#1:1	分割数: 3
検索	#1:3, #2:4, #3:1	データ数: 5
索論	#1:4	

さらに縮退文脈も利用したとき:

情報 (*,0)	#1:1	分割数: 3
検索 (1,0)	#1:3	データ数: 3
索論 (0,*)	#1:4	

図５　検索条件語「情報検索論」に対する位置情報データ読み出し量の比較

フレキシブル文字列インバージョン法では、このトレードオフを改善するために、以下のような5つの手法を導入した[8]。

(1) キー文字列長の字種別可変化
(2) 縮退文脈の付与
(3) 高頻度キー文字列用の副インバーテッドファイル
(4) 位置情報データの圧縮
(5) 文字列統計に基づくパラメータ適合

図２は、フレキシブル文字列インバージョン法で用いる(B-イ)タイプの主インバーテッドファイルの例である[8]。図２は、手法(1)によって漢字２文字組もキー文字列として取り出されている点と、手法(2)の縮退文脈(x,y)が付与されている点が、前述のナイーブな実装と異なる。縮退文脈とは、テキストあるいは検索条件語におけるキー文字列の前後の各１文字の文字コードを、ある値(=縮退文脈幅)未満の値にハッシュ変換したものである(xは前方、yは後方の縮退文脈)。

これら字種別キー文字列長と、キー文字列ごとに定める縮退文脈幅は、図４に示すように、フレキシブル文字列インバージョン法の実装時のパラメータである。手法(5)の考え方により、検索条件語に使われやすい字種のキー文字列長を長くしたり、高頻度キー文字列について縮退文脈幅を大き

くするような調整により、検索時の位置情報データの読み出し量を効率よく削減できる。図5にその読み出し量削減効果の具体例を示す。

図3は、手法(3)の副インバーテッドファイルの例である[8]。フレキシブル文字列インバージョン法で用いる2種類のインバーテッドファイルのうち、主の方は(B-イ)タイプであるが、副は(B-ア)タイプである。検索条件語が部分文字列に分割しなくともそのままキー文字列に一致する場合は、(B-ア)タイプでも検索ノイズは発生せず、(B-イ)タイプよりも高速な検索が可能である。しかし、完全な2通りのインバーテッドファイルをもつのは効率が悪い。そこで、フレキシブル文字列インバージョン法では、主(B-イ)の苦手を補うように、高頻度キー文字列に限定して小規模な副(B-ア)を使用している。

さらに手法(4)として、位置情報データ(#n:m)の配列に対して、直前の要素値との差分をとった上で、NULL バイトを削除して可変長化し、要素の区切りにフラグビットを付与する方法でデータ圧縮も加えている。

3 近接演算

近接演算は、2 個の検索条件語の間の距離に関して条件を与えることをいう。例えば「音声 and 認識」という検索条件は、長いテキストにおいて「音声」と「認識」が無関係に出現してもマッチするのに対して、「音声 near(0-10)認識」のように 10 文字以内などの条件を加えれば「音声」と「認識」に強い関係を期待できるテキストに絞り込める。

このような近接演算は元来、全文走査と相性がよい。テキスト本文をプレインテキスト形式で保存しておき、文字列照合によって検出した検索条件語の間の位置関係を確認することで、近接演算の距離条件が判定できる。

これに対して RetrievalExpress では、全文走査は行なわず、(B-イ)タイプのインバーテッドファイルを用いた近接演算を実現している。(B-イ)タイプでは、キー文字列のオフセット情報をもっているため、検索条件語の出現位置が計算できる。したがって、2 つの検索条件語の位置関係に基づき、近接演算の距離条件の判定が可能である。

ただし、オフセット情報をもたない副インバーテッドファイルでは上記の計算ができない。よって、近接演算の際は、すべての検索条件語に対して主インバーテッドファイルを検索する。その結果、近接演算を使うと、距離条件判定に要するオーバヘッド発生に加えて、前述の手法(3)の効果も消え、検索レスポンスがやや低下する。

4 ランキング

検索結果ランキング(順位付け/スコア付け)の方式[9]にはいくつかのアプローチがあるが、よく知られているのはベクトル空間モデルに基づくものである。これは、各テキストと検索条件とにキーワードベクトルを割り当て、その類似度(cos 値)でソートする方式である。テキストのキーワードベクトルは、各キーワード(ベクトルの各次元に対応)の tf＊idf を計算して求める。ここで、tf は1テキスト中にそのキーワードが頻出するほど大きくなり、idf は少ないテキストに偏って出現するキーワードほど大きくなる。単語をキーとしたインバーテッドファイルでは通常、このようなウェイト値も併せて格納しておき、ランキング計算に使う。

しかし、文字列をキーとしたインバーテッドファイルでは、検索条件語そのものがキー文字列に登録されているとは限らないので、上述のような事前計算のアプローチがとれない。検索条件が与えられた時点で、各テキストにおける検索条件語のウェイト値を計算することになる。このような検索時のウェイト計算を、全文走査によって実現した例[10]があるが、RetrievalExpress では(B-イ)タイプのインバーテッドファイルから得られる情報を用いて実現する。

RetrievalExpress では、検索条件にマッチしたテキストのみを対象にランキング処理を実行する。ランキングを行なわない場合は、1 テキストにつき検索条件語の存在が1個だけ確認できれば十分だが、ランキングを行なう場合は、tf 値を得るために、1 テキストに現われた検索条件語の全位置を求める。したがって、その分の検索レスポンス低下は発生する。一方、各検索条件語が出現したテキスト件数もわかっているから idf 値も得られ、tf＊idf によるスコア計算が可能になる。

表1　ランキングの有無と検索レスポンス

検索条件	該当件数	ランキング無 COLD	ランキング無 HOT	ランキング有 COLD	ランキング有 HOT
液晶	14493	0.6 秒	0.2 秒	2.0 秒	0.6 秒
検索	8027	0.5 秒	0.1 秒	2.0 秒	0.3 秒
半導体	43985	2.2 秒	1.2 秒	4.9 秒	3.0 秒
音声認識	779	1.1 秒	0.3 秒	3.1 秒	0.5 秒
モデム	1160	0.4 秒	0.0 秒	2.4 秒	0.1 秒
CMOS	1137	0.6 秒	0.1 秒	2.2 秒	0.2 秒
インタフェース	3495	1.2 秒	0.3 秒	2.8 秒	0.5 秒
電子メール	310	1.2 秒	0.3 秒	2.7 秒	0.4 秒
複合条件 1	14993	2.4 秒	0.7 秒	3.9 秒	1.3 秒
複合条件 2	1029	1.7 秒	0.2 秒	3.5 秒	0.3 秒

複合条件 1=集積回路 or LSI or IC

複合条件 2=(テキスト or 文書)and(検索 or サーチ)

表2　近接演算と検索レスポンス

検索条件語	演算	該当件数	COLD	HOT
音声，認識	and	1141	0.9 秒	0.2 秒
	near(0-10)	878	1.1 秒	0.4 秒
日，電気	and	35017	1.9 秒	0.9 秒
	near(0-1)	26134	3.7 秒	2.2 秒

5　評価

第2章で述べた各手法の効果については既報告[8]なので、ここでは近接演算やランキングをからめた評価結果を示す(表1・表2)。

テキストは特許公開公報の抄録文 100 万件(約760MB)を用い、UNIX ワークステーション NEC EWS4800/460 (CPU: R10000、200MHz)で実行した。字種別キー文字列長は、漢字:1・2、片仮名:1・2・3、平仮名:1・2、英字:1・2、その他:1 とし、インバーテッドファイルのサイズは約 1650MB (テキストの約 2.2 倍)となった。

表1の10種類の検索条件に対して、ランキングなしのとき COLD(キャッシュ効果なし)で 0.4～2.4 秒、HOT(キャッシュ効果あり)で 0.0～1.2 秒の検索レスポンスが得られた。ランキングありでは COLD で 1.4 秒～2.7 秒の低下、HOT で 0.1 秒～1.8 秒の低下となった。表2の近接演算でも同様の検索レスポンスが見られるが、特に副インバーテッドファイルに登録されるような高頻度キー文字列が検索条件に含まれるケースでの低下が大きい。

6　おわりに

全文検索システム RetrievalExpress で用いているフレキシブル文字列インバージョン法の概要と近接演算・ランキングの実現方法について述べた。特許公開公報の抄録文 760MB を用いた評価により、高速な検索性能が確認できた。ただし、近接演算/ランキング時の検索レスポンス低下に対しては、なお改善の余地があると考える。また、ランキングについては検索精度面でも評価し、改良を加えていくことが重要な課題である。

なお、RetrievalExpress は既にいくつかの実用システムで稼働しており、並列検索、曖昧検索、類似検索などの機能拡張を進めている。それらに関しても、機会を改めて報告したい。

参考文献

[1] 道本ほか、特集:高速全文検索の威力、日経バイト、1996 年 10 月号。

[2] 小川ほか、フルテキストデータベースの技術動向、情報処理、Vol.33、No.4、1992 年。

[3] 石川ほか、特集:日本語テキストを対象とした自動索引システムの課題、情報の科学と技術、Vol. 42、No. 11、1992 年。

[4] 畠山ほか、ソフトウェアによるテキストサーチマシンの実現、情処研報、92-FI-25-4、1992 年。

[5] R. E. Kimbrell, Searching for Text? Send an N-Gram!, BYTE, May 1988.

[6] 菊池、日本語文書用高速全文検索の一手法、信学論、Vol.J75-D-I、No.9、1992 年。

[7] G. H. Gonnet, et al., New Indices for Text: PAT trees and PAT arrays, Information Retrieval: Data Structures and Algorithms, Prentice Hall, 1992.

[8] 赤峯ほか、高速全文検索のためのフレキシブル文字列インバージョン法、情処学会 ADBS シンポジウム、1996 年。

[9] D. Harmann, Ranking Algorithms, Information Retrieval: Data Structures and Algorithms, Prentice Hall, 1992.

[10] 芥子、ベクトル空間モデルに基づくフルテキストサーチシステム、第 6 回人工知能学会全国大会、1992 年。