自 然 言 語 処 理 104－7
（1994．11．17）

複数キーワードによる検索の一高速化手法

有田 健† 津田和彦† 入口浩一† 青江順一†

†徳島大学工学部

近年の電子記憶媒体の大容量化に伴って，大容量の文書データから必要とする情報を高速に検索する技法の研究はますます重要になってきた．全文検索法は，特徴ベクトルにより走査対象の文書ブロックを絞り込む手法と，マッチングマシンを構成して源文書を全て探索する手法とに分けられ，後者はキーワードが１個の場合，複数の場合に対して種々の技法が考案されている．しかし，複数キーワードに対する前者の特徴ベクトル拡張手法の研究成果は見あたらない．本研究では，特徴ベクトルを用いた全文検索において，特徴ベクトルの特性に着目し，複数のキーワードを全文検索する際の高速化手法を提案する．本手法により、キーワード数が16から32個の場合で、複数キーワードのパターンマッチングＡＣ（Aho-Chorasic）法を使用した場合より約10から17倍高速になることが分かった．

The Fast Algorithm of Full Text Retrieval for Multiple Keywords

Takeshi ARITA† Kazuhiko TSUDA† Hirokazu IRIGUCHI†
and Jun-ichi AOE†

† Faculty of Engneering, The University of Tokushima

Text retrieval methods have attracted much interest recently. There are numerous applications involving storage and retrieval of textual data: Electronic office filing, Computerized libraries, Automated law and so on. A well-known and simple approach of searching texts is full text retrieval using signature files, but the method can not apply a finite number of keywords. This paper presents a fast retrieval algorithm for multiple keywords by using characteristic of multiple signatures. The algorithm decreases the number of comparisons between multiple signatures. From the simulation result, it is show that the algorithm presented is from 10 to 17 times faster than the traditional approach for from 16 to 32 multiple keywords.

1. はじめに

近年の電子記憶媒体の大容量化に伴って，CD-ROM出版などの電子出版が現実に行われるようになってきたため，大容量の文書情報から必要とする一部の情報を探し出す技法の研究はますます重要な課題となってきた[Cristos, 92], [Faloutsos, 85].

この検索の代表的な技法の一つは，キーワード情報を文書（一次情報）とは別の二次情報として記憶しておき，そのキーワードに対する索引表を構築して検索するキー検索技法である．検索速度は，既に考案されている多彩な検索アルゴリズムを適切に採用することで実用上問題はない[Aoe, 91], [park et al., 94]．この手法には，次のような問題点がある．

(1) 大量の文書データからのキーワード抽出に多大な労力と時間を必要とする．
(2) 検索対象はキーワードにより制約され，任意の語彙の検索が不可能である．

(1)については，自動的にキーワードを切り出す手法が提案されているが，最終的には人の手を煩わすことになる．特に，日々新しい語が増えている現状では，キーワード辞書の管理と保守が必要不可欠である．また，(2)については，この方式の構造的な問題であり，解消するためには膨大な索引表を必要とする．

この手法と対照的な手法に，検索指定キーワードで一次情報である全文書を検索する全文検索法がある[Aoe, 94], [小川他, 92]．また，全文検索法は，キーワードに対する検索マシンを構成して，全文書を走査する手法（全走査法と呼ぶ）と，分割された文書ブロック単位にそのブロックの特徴ベクトル（signature vectors, これを集めたものを特徴ファイルと呼ぶ）を事前に照合することで，一部の文書ブロックのみを走査する手法（部分走査法と呼ぶ）に分類することができる．両者の手法とも，上記に示した索引表による欠点を解消するが，検索時間は文書の大きさに比例して長くなる欠点をもつ．従って，文書検索システムでは，まずキーワードを索引表で高速検索を行い，検索できない場合は，全文検索にゆだねるのが妥当な構成となる．これに対して，ハードウエア技術を駆使した全文検索の高速化も多く研究[Cheng et al., 87], [Foster eta al., 80],[Halaas, 83],[Haskin, 81],[Haskin et al., 83],[Hollaar, 79],[Hollaar, 83],[Isenman et al., 90]されているが，応用面の広さを考えると，やはりソフトウェアによる全文検索に対する高速化は，一つの重要な研究課題である[菊池, 92].

全走査法において，１個のキーワードに対するアルゴリズムはKnuth-Morris-Pratt法[Knutk et al., 77]や，Boyer-Moore法[Boyer et al., 77]が，複数のキーワードに対するアルゴリズムは，Aho-Corasick法[Aho et al., 75](AC法と呼ぶ)がよく知られている．また，部分走査法では，特徴ベクトルの構成法に変化はあるが，基本的には特徴ベクトルなる圧縮された文書情報を用いて，キーワードの照合範囲を絞りこむ方法である．この手法では，文書をまずブロックに分け，それぞれの文書ブロックに特徴ベクトル（文書ベクトルと呼ぶ）を構築する．この特徴ファイルは，源文書の大きさの約10%前後の大きさとなる．検索時には，キーワードの特徴ベクトル（キーワードベクトルと呼ぶ）と，文書ベクトルを照合し，文書ブロックにそのキーワードが含まれている可能性がある場合にのみ，文書ブロッを部分走査する．しかし，特徴ベクトルによる手法は，一つのキーワードに限定したものであるので，本稿では複数キーワードに対する特徴ベクトルの技法を提案する．

キーワードが複数になると，対応するキーワードベクトルと文書ベクトルを繰り返して照合するので，キーワード数に比例して照合時間が長くなる．本研究では，特徴ベクトルの特性に着目し，複数のキーワードに対する高速化手法を提案する．また，実験によるシミュレーションにより，複数キーワードの全走査法であるＡＣ法と比較し，提案手法の有効性を確かめる．

2. キーワード検索

本章では，索引検索について簡単に触れた後，本研究の対象である全文検索の技術について述べる．

2.1 索引検索

索引検索の概念図を図2-1に示す．

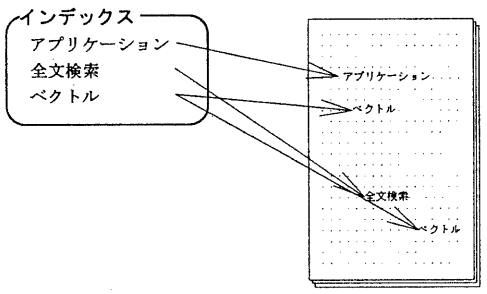


図2-1　索引検索の概念図

キーワードの検索時には，索引中からキーワードを検索し，キーワードの位置情報を得る．この検索方法は，非常に多く提案されているキー検索技法（ハッシュ法，２分探索法，Ｂ＋木法[Aoe, 91]など）が適用できるので，高速な検索が実現できる．しかし，図2-1のようにキーワードの多くの位置情報を格納する必要があるので，索引部の記憶量は多くなる．

2.2　全文検索

本章では，文字列照合の技術である，単一のキーワードに対するKnuth-Morris-Pratt法，Boyer-Moore法については，本研究と直接関係がないので，省略し，複数キーワードの検索技法であるAC法について述べ，次に特徴ベクトルを用いた全文検索法について述べる．

2.2.1　Aho-Corasick(AC)法

キーワードが複数になると一般的には単一キーワードの照合を繰り返さなくてはならない．特に，複数キーワードでは，一方のキーワードに対しては照合が失敗したが，他方のキーワードに対しては照合が成功しているなど，キーワード間の重なりや絡み合いが生じ，処理は複雑化する．

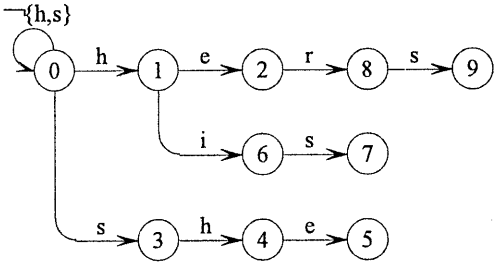
Ahoらは，これらの問題点を解決するため検索するキーワード集合より状態遷移図を作成し，この状態遷移図上でマッチングを実行し，マッチングが失敗した際に遷移する状態を表すfailure関数を定義することで効率的マッチングを実現している．以下に基本的なAho-Corasick法について説明する．

ここでKはキーワードと呼ばれる記号列y1,y2･･･.yh;h≧１を要素とする有限集合を表す．マシンACはKに属するキーワードyおよびその位置を任意のテキスト記号列xの中から検出する．但し，ここで部分記号列は互いに重複することも許す．マシンACは状態集合Sより構成され，各々の状態は整数値で表される．特に初期状態を番号0で表す．入力記号をIとするとき，マシンACの動作は次の３つの関数で制御される．

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

failure関数　$f: S \rightarrow S$,

output関数　$output: S \rightarrow K$の部分集合.



(a) Goto function

s	f(s)
1	0
2	0
3	0
4	1
5	2
6	0
7	3
8	0
9	3

(b) Failure function

s	output(s)
2	{he}
5	{she,he}
7	{his}
9	{hers}

(c) Output function

図2-2　パターンマッチングマシンACの関数

図2-4にキー集合K={he,she,his,hers}に対するマシンACの関数を示す．但し，図2-2(a)の$\neg\{h,s\}$は，h,s以外の記号を表す．

図2-2(a)の状態遷移図はgoto関数を示しており，例えば0から1のhとラベル付けされた矢はg(0,h)=1と表し，状態$s \in S$と入力$i \in I$について，sからiとラベル付けされた矢が出ていない場合には，g(s,i)=fail と表す．また，初期状態0は$g(0, \neg\{h,s\}) = 0$が定義されているため，いかなる入力記号に対してもgoto関数はfailとならない．

図2-2(b)はfailure関数であり，goto関数による状態

入力記号	u	s	h	e	r	s
goto関数による状態遷移	0	0	3	4	5	8	9
failure関数による状態遷移				2			

図2-3　マシンACの状態遷移

遷移に失敗した際，すなわちgoto関数による状態遷移でfailが返ってきたときの状態遷移先を示す関数である．例えば，状態4でgoto関数による状態遷移に失敗した際は状態1に遷移し，goto関数による状態遷移を行えばよいといったことを示している．

図2-2(c)はoutput関数であり，マシンACが状態5に到達すると，キーワード{she,he}が検出されたこととなる．

図2-2に示すマシンACの，入力文字列"ushers"に対するマッチング動作を図2-3を用いて簡単に説明する．まず，初期状態0より文字'u'に対するgoto関数が定義されているかどうかを検索する．$u = \neg\{h, s\}$ よりg(0,u)=0，以下同様にg(0,s)=3, g(3,h)=4，g(4,e)=5と状態5まで遷移する．状態5では，output(5)≠emptyよりキーワード{she,he}を検出する．g(5,r)=failよりfailure関数による状態遷移f(5)=2により状態2へ遷移し，g(2,r)=8，g(8,s)=9へと遷移し，状態9においてキーワード{hers}を検出しマッチングを終了する．

AC法のマシン構成時間は，総キーワードの長さに比例した時間となる．また，このマシンを動的に構成する手法は，[津田他, 94]を参照されたい．

2.2.2　特徴ベクトルを用いた検索

特徴ベクトルによる検索の特徴は，文書を部分走査することで，文書全体の操作時間と文書

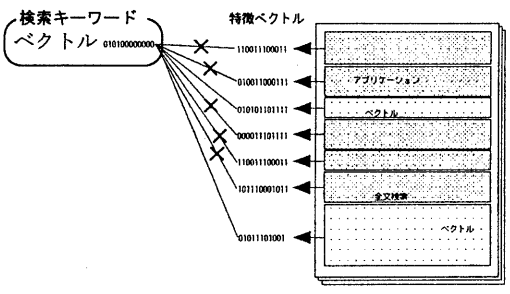


図2-4　特徴ベクトルを用いた検索の概念図

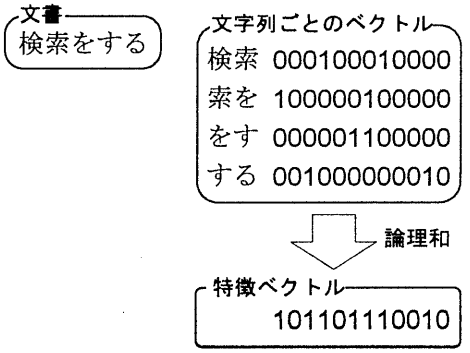


図2-5　特徴ベクトルの作成

ブロックの補助記憶から主記憶上への転送時間を軽減することである．図2-4に特徴ベクトルを用いた検索の概念図を示す．

文書はブロック分けされ，ブロックごとに特徴ベクトルである文書ベクトルをもつ．これは，文書ブロックの文字列情報を圧縮したものであり，英語のように単語が分かち書きされている場合は，各単語に対する特徴ベクトルを重ね合わす（論理和）手法[Faloutsos, 85]がとられるが，日本語の場合は隣接文字列から特徴ベクトルを構成する方法が多い．隣接の部分文字列[Harrison, 71]に対する特徴ベクトルは，文字コードの和をある定数で割るなどして作成した値を用いる．その文字列に固有な値ならどのようなものでも使用可能だが，ばらつきが大きい方が望ましい．この手法であれば，文書ブロックの任意の文字列をキーワードに指定することが可能となる．

キーワード検索時には，キーワードベクトルを作成し，文書ベクトルとの照合を行う．この照合で，キーワードが存在しないことが確認されれば，文書ブロックは検索を行わない．文書をブロック分けし，そのブロックごとに特徴ベクトルを作成した説明図を図2-5に示し，検索の例を図2-6に示す．

図2-6の例では，検索キーワードとして「検索」と「特徴」と「文書」の3つが与えられている．キーワードベクトルと，文書ベクトルの論理積が検索キーワードの特徴ベクトルと一致しなければ，そのキーワードはその文書ブロックに存在しないことが判明する．この例では，「特徴」に対するキーワードベクトルと文書ベ

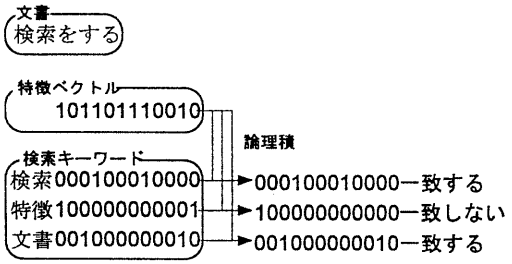


図2-6　特徴ベクトルの照合

クトルの照合で，この文書ブロックは検索する必要がないということが分かる．しかし，「文書」に対するキーワードベクトルと文書ベクトルの照合は成功するが，実際には文書ブロックの中には存在しない．このようにベクトル照合が成功しても，文書ブロックの中には対応するキーワードが存在しないことをフォルスドロップ(False Drop)と呼び，このフォルスドロップの確率が少ない程，全文検索の速度は向上する．

3. 複数キーワードによる全文検索の高速化手法

3.1 特徴ベクトル照合の高速化

特徴ベクトルを用いた全文検索の場合，文書ベクトルとキーワードベクトルの論理積をとり，それがキーワードベクトルと一致しないかでそのキーワードの存在を判別している．即ち，この論理積が一致しないことは，

「文書ブロックの特徴ベクトルのビットが0の位置に対応するキーワードの特徴ベクトルのビットが1の場合」

なる論理積の特性を意味する．従って，本手法ではこの論理積の特性を利用した検索手法を提案する．提案手法の概念図を図3-1に示す．

文書ブロックベクトルが0の位置に対応するキーワードベクトルのビットがひとつでも1になっている場合，そのキーワードは存在しないので，文書ベクトルのビットが0の位置に対応するキーワードベクトルのビットのそれぞれの論理和をビットスライスで[Grandi et al., 92]結果ベクトル上に計算する．この手順を繰り返すことで，結果ベクトルのビットが1であるキーワードは対応する文書ブロックに存在しないことが

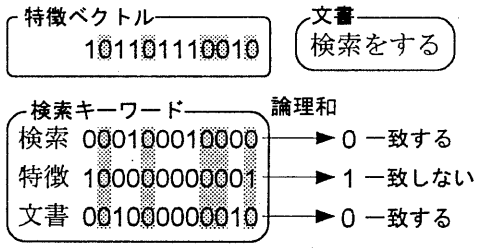


図3-1　特徴ベクトル照合の高速化手法

判明する．

この方法では，文書ベクトルのビット0の数だけ結果ベクトルへの論理和演算を繰り返すので，この回数がキーワード数より多い場合は，単純に個々のキーワードベクトルと論理演算を繰り返す場合より，計算コストは多く成るように思えるが，結果ベクトルのビットのビットが全て1に成ったとき，繰り返し演算は途中で終了できるので，早期に文書ブロックの走査判定を行える特徴がある．また，キーワードが多くなれば，結果ベクトルの計算は個々のキーワードベクトルを繰り返すコストより，当然少なくできる．また，キーワードが多くなれば，必然的に部分走査する文書ブロックの個数は増大し，AC法の照合時間に近づいてくる．これらは，次章で議論する．

3.2 複数特徴ベクトルの照合アルゴリズム

提案手法の照合処理は，次のようにまとめられる．

(a) 複数キーワードに対して，マシンACを構築する．
(b) 複数キーワードに対するキーワードベクトルを作成する．
(c) 文書ベクトルと(b)で構成したキーワードベクトルで照合を行う．
(d) (c)で文書ブロックの照合が不要となれば，次の文書ブロックに対して，(c)を実行する．
(e) (c)で文書ブロックの照合が必要になれば，文書ブロックを転送して，(a)で構成したACマシンで文書走査を行い，最終照合を行う．

以下に(c)の手順に対する照合アルゴリズムを示す．

配列BVectorは，文書ベクトルビット配列であり，最初の添字が文書ブロック番号，2番目の添字がビット番号を示す．配列KVectorはキーワードベクトルであり，最初の添字がキーワード番号，2番目の添え字はビット番号を示す．Matchは該当キーワードが含まれていないかどうか判定するフラグを格納する結果ベクトルでああり，添字はキーワード番号を示す．キーワードの数をKEYWORD_SIZEとし，特徴ベクトルのビット数をBIT_SIZEとした時，文書ブロック番号blockの特徴ベクトル照合は以下のアルゴリズムで行う．

```
Method
 begin
  for keyword←1 until KEYWORD_SIZE do
   begin
    Match[keyword] = 0;
   end

  for bit=0 until BIT_SIZE do
   begin
    if BVector[block][bit] = 0 then
     begin
      for i←0 until KEYWORD_SIZE then
       begin
        Match[i]←Match[i] | KVector[i][bit]
       end
     end
   end
end
```

最初のfor文でMatchをクリアする．つまり，全てのキーワードが存在の可能性があるように初期化する．次のfor文で文書ベクトルをビットごとに処理するためにビット長でループ処理を行う．次のif文で，文書ベクトルにビット１が立っていないかどうか調べ，文書ベクトルがビット0である場合にのみ，Matchを次のfor文で更新する．即ち，文書ベクトルのビット１の位置に対応するKVectorのビット列（縦方向）をMatchと論理和をとる．

２番目のfor文の動作を図3-1の例で説明すると，文書ブロックのベクトル，BVectorの最初のビットつまりbit=0の位置のビットは0であるので，Matchの更新は行われない（if内が実行されない）．次のビット（bit=1）は0なのでMatchの更新が行われ，検索キーワードの該当ビット，0,0,0,が順次論理和をとって格納される．このように実行していき，最終的に1になったキーワードが該当文書ブロックに存在しないことが判明する．

4. 評価と考察

4.1 実験による評価

複数キーワードの代表的な検索法であるＡＣ法との検索時間比較により，提案手法の評価を行う．また，単一キーワードベクトルを繰り返して文書ベクトルと単純に比較する方法（単一ベクトル法と呼ぶ）場合の検索時間も提案手法のベクトル照合の有効性をために，結果を行った．実験結果を図4-1に示す．なお，実験にはNEC社製PC-9821Apを用いた．評価用プログラムはC言語にてインプリメントを行った．また，表中の検索時間の単位は秒である．文書ブロックのサイズは，ビット長64の特徴ベクトルビット１の数はビット長の約50%に近づくように調整し，128バイトとした．対象文書は，岩波書店の広辞苑テキストデータ（約28メガバイト）である．

この実験により，キーワード数が多いほど，本手法の方が単一ベクトル法より高速になることが分かった．これは，キーワード数が多いほど本手法の論理演算の回数が減少することに起因する．本質的に特徴ベクトルによる前処理で，

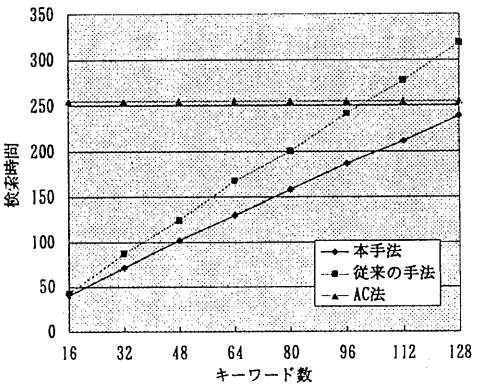


図4-1　実験結果

源文書を探索する回数は，本手法と単一特徴ベクトル法とは同じであるので，速度改善はこの特徴ベクトルの探索の効率化により実現されている．これに対して，AC法では，常に源文書を走査するので，キーワード数の多少に関わらず，源文書の長さに比例して検索時間が遅くなる．ただし，キーワード数が多くなると，ACマシンを構築する時間も若干多くなるが，この時間はキーワード長の合計に比例するので，源文書が長い場合は，無視できる．結局，提案手法がAC法より高速となっているが，これは特徴ベクトルの前処理で源文書の探索がどの程度の割合で排除できるかに依存している．従って，キーワード数が多くなり，文書ブロックの部分走査数が多くなると，提案手法の検索速度は，AC法の検索時間に近づくが，本実験で示した128個以内のキーワード数は，十分実用的な範囲であるので，提案手法は実用的有効性を十分保持しているといえる．

なお，提案手法ではメモリ使用量がキーワードが存在するかどうか結果ベクトル分だけ増加するが，容量的問題とはならない．

4.2 考察と今後の課題

文書検索では，一つのキーワードを指定してもその同義語や類義語も含めて検索したい場合が多々あるので，提案手法はこの様な検索範囲の拡張にも容易に対処できる特徴をもつ．

また，議論している検索は複数キーワードの何れか（OR演算）が，含まれている文書ブロックを検索対象とするが，複数のキーワードを同時に含む（AND演算）場合は，それらのキーワードベクトルの論理積をとることでまとめることができ，事前にキーワードを絞り込むことができるので，速度改善の向上が期待できる．

提案手法は，文書ベクトルを順次照合するが，その照合結果は再利用されていない．もし，キーワードベクトルと文書ベクトルの照合で文書ブロックの転送が行われないと判定されたとき，その文書ベクトルは，有効判定ベクトルとして以後の照合に利用することができる．従って，この有効判定ベクトルを学習しておけば，キーワードベクトルと文書ベクトルの照合の前に，有効判定ベクトルと文書ベクトルの照合（有効判定ベクトルのビット0の全ての位置で文書ベクトルのビットが0であればよい）を行えば，文書ブロックの走査をより早い段階で判定できる．この場合，有効判定ベクトルは，ビット0の総数が少ない文書ベクトルを優先的に学習すれば，この前処理の効果はより高くなる．

以上のように，複数キーワードベクトルと文書ベクトルの効率的論理演算を思案中であるが，まだまとめるに至っていない．また，提案手法の理論的評価についても現在検討中である．

5. おわりに

複数キーワードにおける特徴ベクトルを用いた全文検索の高速化手法を提案し，従来の検索手法ＡＣ法との比較実験により有効性を評価した．本手法により，全文検索の応用範囲が広がることを期待している．

４章でも触れたように，提案手法の理論的評価，キーワードのAND,OR演算への検討，有効判定文書ベクトルの学習法などが今後の課題として残されている．また，本実験では，１種類の文書データしか対象としなかったが，より多彩な文書データでの実験と評価を行う必要がある．

謝辞　文書データの実験に協力いただいた岩波書店辞書部の方々に深謝致します．

参考文献

[Aoe, 91]
J. Aoe : Computer Algorithms -Key Search Strategies- , IEEE Computer Society Press (1991).

[Aoe, 94]
J. Aoe : Computer Algorithms -String Pattern Matching-, IEEE Computer Society Press (1994).

[津田他, 94]
津田和彦，入口浩一，青江順一：複数キーワードに対するパータンマッチングマシンの動的構成法，電子情報通信学会論文誌，第J77-D巻，第４号，pp.282-289 (1994)．

[菊池, 92]
菊池　忠一：日本語文書用高速全文検索の一手法，電子情報通信学会論文誌D-I, Vol.J75-D-I, No.9, pp.836-846 (1992).

[小川他, 92]
小川 隆一, 菊池 芳秀, 高橋 恒介 : フルテキスト・データベースの技術動向, 情報処理学会誌 Vol.33, No.4, pp.404-412 (Apr. 1992).

[Cristos, 92]
Cristos Faloutsos, Signature Files, in Frakes, W., and Baeza-Yates, R.A., eds., Information Retrieval: Algorithms and Data Structures, Prentice Hall, Englewood Cliffs, N.J., pp.44-65 (1992).

[Aho et al., 75]
Aho, A. and Corasick, M. : Efficient String Matching : An Aid to Bibliographic Search, Communications of the ACM, Vol.18, No.6, pp.333-340 (June 1975).

[Boyer et al., 77]
Boyer, R. and Moore, J.S. : A Fast String Searching Algorithm, Communications of ACM, Vol.20, No.10, pp.762-772 (Oct. 1977)

[Knuth et al., 77]
Knuth, D., Morris, H.J. and Pratt, V.R. : Fast Pattern Matching in Strings, Technical Report STAN-CS-74-440, Sci. Dep., Stanford University (Aug. 1974).

[Cheng et al., 87]
Cheng, H., and Fu, K., VLSI architectures for string matching and pattern matching, Pattern Recognition, 20, pp.125-141(1987).

[Faloutsos, 85]
Faloutsos, C., Access methods for text, ACM Comput. Surveys, 17, pp.49-74(1985).

[Foster et al., 80]
Foster, M.J., and Kung, H.T., The design of special purpose chips, IEEE Comput., 13, pp.26-40(1980).

[Grandi et al., 92]
Grandi, F., Tiberio, P., and Zezula, Frame-slice partitioned parallel signature files, ACM SIGIR'92, pp.286-297(1992).

[Halaas, 83]
Halaas, R.L., A systolic VLSI matrix for a family of fundamental search problem, Integration VLSI J., 1, pp.269-282(1983).

[Harrison, 71]
Harrison, M.C., Implementation of the substring test by hashing. C. ACM, 14, pp.777-779(1971).

[Haskin, 81]
Haskin, R.L., Special-purpose processors for text retrieval, Database Eng., 4, pp.16-29(1981).

[Haskin et al., 83]
Haskin, R.L., and Hollaar, L.A., Operational characteristics of a hardware-based faster matcher, ACM Trans. Database Syst., 8, 1(1983).

[Hollaar, 79]
Hollaar, L.A., Text retrieval computers, IEEE Comput. Mag., 12, 3, pp.40-50(1979).

[Hollaar, 83]
Hollaar, L.A., Architecture and operation of a large, full-text information-retrieval system, in Hsiao, D.K., ed., Advanced Database Machine Architecture, Prentice Hall, Englewood Cliffs, N.J., pp.256-299(1983).

[Isenman et al., 90]
Isenman, M.E., and Shasha, D.E., Performance and architectural issues for string matching, IEEE Trans. Comput., C-39, 2, pp.238-250(1990).

[Park et al., 94]
K-H. Park, J. Aoe and M. Fuketa:"A Method for Selecting Key Search Algorithms Using Expert Knowledge,," SIGIR, ACM Press, 28, 3, pp.13-25 (1994).