日本語文書検索のための頻度情報を用いた効率的部分文字列索引の提案

小川泰嗣†

文書における部分文字列の出現を記録する部分文字列索引は，単語切り出しのための言語処理が不要という点で日本語文書検索向きである．しかし，出現位置情報を捨象しているため誤検索が発生する，検索語の長さに応じて検索時間が増大する，という問題があり，こうした点を改善するためには索引サイズを大きくせざるをえない．すなわち，誤検索率・検索時間・索引サイズのすべてに優れた部分文字列索引を実現することは困難であった．本論文では，ユーザの使用する可能性の高い検索語の処理を優先的に高速化し，誤検索率・索引サイズを悪化させることなく平均検索時間を短縮する効率的な部分文字列索引を提案する．このために，文字および部分文字列の2レベルの頻度を用いる．文字レベルの出現頻度は二文字組から索引エントリを決定するためのハッシングに利用され，検索の高速化および誤検索の低減に作用する．一方，部分文字列レベルの出現頻度は長い部分文字列を独立した索引エントリとして選択するために利用され，出現頻度が高く長い検索語の処理の高速化に作用する．特許要約文10万件（14 MB）を用いて検索時間・検索精度・索引サイズの評価実験を行い，本手法の有効性を確認できた．

An Efficient Substring Indexing Method for Japanese Document Retrieval Using Frequency Data

YASUSHI OGAWA†

Substring indexing method is suitable for Japanese document retrieval systems, because it requires no natural language processing to identify words. A substring index does sometimes create false drops and retrieval time is proportional to the query length, however, an index must be large to reduce false drops or shorten retrieval response. In other words, it's difficult to get good performance measures for false drops, retrieval time as well as index size. We propose a new substring indexing method that shortens the average response time by selectively fastening retrievals of frequently used words, and yet does not either increase false drops or enlarge indexes. We use two kinds of frequency data: (1) Character frequency is used to generate a hash table for character-pairs, both increasing retrieval speed and reducing false drops, (2) Substring frequency is used to select special substring index entries, resulting in quick retrieval for long but highly frequent words. We evaluated our method using 100,000 patent abstracts. Measurement results, including response time, index size and false drop rate, confirmed the effectiveness of our method.

1. はじめに

近年のコンピュータ・通信技術の発達にともない電子化文書が氾濫するようになっており，大規模文書データベースから所望の文書を特定するための文書検索システムへの要求が高まっている[4),7),26)]．大規模文書データベースを対象とする場合，検索高速化のために索引が不可欠である☆．

英語など単語が明示的に示される言語を対象とした検索システムでは，単語を索引の基本単位（以下「エントリ」と呼ぶ）とする「単語索引」を採用しているものがほとんどである[2),23)]．これに対し，日本語ではスペースなどにより単語が明示的に示されないため，単語索引を採用するには，単語を切り出すための言語処理が不可欠である．しかし，言語処理を用いた場合，(1) 解析誤りを無くすることはほとんど不可能であり，解析誤りに基づく検索洩れや誤検索が避けられ

† 株式会社リコー情報通信研究所
Information and Communication R&D Center, RICOH Co., Ltd.

☆ 本論文では，文書の内容理解などに基づく意味的検索ではなく，検索条件に指定された文字列を含む文書を特定する表層的検索を対象とする．また，「索引（index）」を「文書検索を高速に実現するためのデータ構造」の意味で用いる．

ない，(2) 言語処理系には一般に辞書類が必要であるため，その作成・管理にコストがかかるとともにシステムが大規模にならざるをえない，という問題が発生する[16),24)]．一方，文書中の部分文字列をエントリとする「部分文字列索引」では，索引を構成するために単語を識別する必要がない[18),19)]．その結果，検索システムに言語処理系を組み込む必要がなくなるので，部分文字列索引はシステムの小型化や維持管理の簡易化の点で好ましい☆．

部分文字列索引の構成法は，登録文書における部分文字列の出現位置を記録する方式[15)]と記録しない方式[10)]に大別できる[22)]．索引サイズの点で，後者は比較的小型であるのに対し，前者は出現情報を保持するため索引サイズは大きい[11)]．そのため，実際の文書検索システムでは後者が採用されることが多い[18),19)]．本論文も出現位置を記録しない形式の部分文字列索引を対象とする．

しかし，この方式には，(1) 部分文字列の出現位置を捨象しているため，検索文字列を実際には含まない文書が誤まって検索される，(2) 検索語の長さに応じて検索時間も増大する，という2つの問題がある．索引のエントリ数を増加することでこれら問題の影響を小さくすることができるが，エントリ数の増加は索引の大型化につながる．すなわち，誤検索率・検索時間・索引サイズをすべて満足する部分文字列索引を構成することは難しい[19),21)]．

本論文では，誤検索率・検索時間・索引サイズのすべてに優れた部分文字列索引の構成法を提案する．本手法では，二文字組から索引エントリを決定する際に文字レベルの頻度情報を，長い部分文字列を独立した索引エントリとして選択する際に部分文字列レベルの頻度情報を利用する[20),21)]．その結果，従来方式と比較して，索引サイズをほとんど増大させることなく誤検索率の低下および検索時間の短縮を実現できる．

以下，2章で，既存の部分文字列索引について述べ，その問題点を明らかにする．3章で，今回提案する部分文字列索引の設計方針および詳細な構成を説明する．4章では，本手法の有効性に関する評価実験の結果およびその考察を示す．

2. 部分文字列索引とその問題点

部分文字列索引は，部分文字列をエントリとし，互いが部分的に重複するものも含めて，文書中の部分文字列の出現を記録するものである[22)]．文書中のすべての部分文字列を記録することはきわめて困難なので，実際には適当な部分文字列のみを選択する．部分文字列索引の選択法に応じて，等しい長さの部分文字列を選択する「固定長部分文字列索引」☆☆と，長さとは無関係に高出現頻度のものを選択する「可変長部分文字列索引」がある[1)]．

2.1 固定長部分文字列索引

2.1.1 処理方式

固定長部分文字列索引は，等しい長さの部分文字列をエントリとする．索引に用いる部分文字列の選択に長さという単純な基準を用いているので，辞書類はまったく必要なく，システム構成を簡単にできる．

文書登録では，登録文書中にある長さ n の部分文字列のすべての出現を索引に記録する．文書検索処理は，検索語の長さ m に応じて異なる．$m \geq n$ の場合，検索語に含まれる部分文字列は $m-n+1$ 個存在しているので，それらすべてが同時に存在している文書を検索結果とする．$m < n$ の場合，文字の異なり数（以下「字彙」と呼ぶ）を c と書くと，検索語を含む部分文字列が c^{n-m} 個存在しているので，それらの少なくとも1つが存在している文書を検索結果とする．$m > n$ の場合，検索語から抽出された部分文字列がすべて存在する文書でも，それらが連続位置に出現しているとは限らないので，実際には検索語を含まない文書が誤検索されることがある．

2.1.2 特性と問題点

この索引の特性は以下のようにまとめられる．

- 索引サイズ
  部分文字列の異なり数は c^n となり，索引サイズはこれにほぼ比例する．
- 検索時間
  検索語から抽出される部分文字列の個数（$m \geq n$ の場合 $m-n+1$，$m < n$ の場合 c^{n-m}）にほぼ比例する．日本語では $c \gg 1$ なので，n 文字未満の検索語の検索には非常に時間がかかる．
- 誤検索率
  部分文字列を長くすれば，検索語から抽出される部分文字列間の重なりが大きくなる．すなわち，ばらばらに出現したものが偶然に検索条件を満たす確率は低下し，誤検索は低減される．

ここに示したように，部分文字列の長さ n は索引サイズ・検索時間・誤検索率に相反する影響を与え，

☆ 意味的検索の検索精度の点からは単語索引に好ましい面も多いが，部分文字列索引も同等の検索精度を達成できるという報告もある[5)]．

☆☆ この索引を n-gram 索引，特に $n = 1, 2$ のものを文字成分表と呼ぶこともある．しかし，本論文では名称を固定長部分文字列索引に統一する．

すべてを満足させることはきわめて難しい．特に対象が日本語である場合，$n=1$ の「単一文字索引」では十分な検索精度が得られないが，$n=2$ の「二文字組索引」では索引サイズが大きくなる．すなわち，単純な固定長部分文字列索引では検索精度か索引サイズのいずれかが極端に悪化するという問題がある．

2.1.3 ハッシュによる索引エントリの圧縮

上述の問題を回避するため，複数の部分文字列をハッシュを用いてグループ化し，1つのエントリで複数の部分文字列の出現を記録する方式が提案されている．この方式では，ハッシュ法の変更により索引サイズと誤検索率のトレードオフを調整できる．これまでに，以下のハッシュ法が提案されている．

(1) 文字コード・ハッシュ
文字コードにハッシュ関数を作用させ，その値を索引エントリとする[9),25)]．

(2) 文字種分割型文字コード・ハッシュ
日本語には漢字・平仮名・片仮名など複数の文字種があり，固有の役割，異なった出現特性を持っている．しかし，文字コード・ハッシュ法は異なる役割・特性の文字種を一律に扱うので，索引サイズの増大あるいは誤検索率の悪化につながる．これに対し，文字種ごとにその特性に合わせたハッシングを行えば，上述の問題を回避できる[6),13)]．この手法を文字種分割型文字コード・ハッシュと呼ぶ．

(3) 出現頻度ハッシュ
部分文字列の出現頻度のばらつきは大きい[27)]．しかし，文字コード・ハッシュ方式は出現頻度とは無関係の文字コードに基づいてエントリを決定するので，エントリの出現頻度（エントリに割り当てられる部分文字列の頻度の総和）のばらつきも大きくなり，誤検索率が高くなる．これに対し，エントリ頻度を平均化するようにハッシュ先を決定すれば，頻度の高い部分文字列が特定のエントリに集中することを防ぎ，誤検索率を低くできる[8),14)]．この手法を出現頻度ハッシュと呼ぶ．

さらに，ハッシュを適用した二文字組索引を単一文字索引と組み合わせることで，索引を小型化する方式が提案されている[12)~14),25)]．この方式では，索引サイズを小さくするために二文字組の索引エントリ数を減らすことによる検索精度の低下を，単一文字索引との組合せで補う．さらに，単一文字索引を用いることで，単一文字の検索語を高速検索できるという利点も得られる．ただし，この方式には，検索語の長さの検索時間に対する影響が大きいという問題点がある．m 文字の検索語の処理では，m 個の単一文字と $m-1$ 個の二文字組の計 $2m-1$ 個のエントリへのアクセスを必要とするからである[20)]．

このように，ハッシュや異なる n の部分文字列を組み合わせることで，検索時間・誤検索率・索引サイズの2つを満たすことはできても，すべてを同時に満たすことはできなかった．

2.2 可変長部分文字列索引

2.2.1 処理方式

可変長部分文字列索引では，部分文字列の頻度を調べ，適当数の頻度の高い部分文字列で索引エントリを構成する[1)]．この方式では，索引を構成する部分文字列を保持するための辞書が必要である．実用的には，数千～万程度の部分文字列を選択しておく必要があり，この辞書は二次記憶装置におかれる．

文書登録では，登録文書から辞書と照合するすべての部分文字列を抽出し，その出現を索引に記録する．辞書には通常他の部分文字列を包含する部分文字列も登録されており，他を包含している部分文字列を持つ文書には包含されているものは必ず存在する．そこで，文書検索では，検索語から抽出された部分文字列から他に包含されているものを除外した，残りの部分文字列のすべてを保有している文書を検索結果とする．

2.2.2 特性と問題点

この索引の特性は以下のようにまとめられる．

- 索引サイズ
辞書エントリ（登録された部分文字列）数に応じた大きさとなる．
- 検索時間
短い部分文字列ほど頻度は高い傾向にあるので，辞書エントリ数を多くすると検索語から抽出される部分文字列は長くなり，検索に使用される部分文字列数は減少する．すなわち，辞書エントリ数を多くすると検索時間は短縮される．
- 誤検索率
辞書エントリ数を多くすると長い部分文字列が多くなり，部分文字列間の重なりも大きくなる．したがって，辞書エントリ数を増やすことで誤検索は低減される．

ここで示したように，辞書エントリ数が索引サイズ・検索時間・誤検索率に相反する影響を持っており，固定長と同様に，これらすべてを満足させることは困難である．さらに，日本語に適用する場合，日本語は字集が大きく文字ごとの出現頻度の差が大きいので，辞書には非常に大量の部分文字列を登録しなければなら

ない．そのため，可変長部分文字列索引の日本語への適用事例は見当たらない．

3. 頻度情報を利用した部分文字列索引の提案

前章で示したように様々な部分文字列索引が提案されているが，誤検索率・検索時間・索引サイズのすべてを満足させることは難しい．本章では，部分文字列に関する頻度情報を利用した部分文字列索引を提案し，この索引により上記3点をすべて改善できることを示す．

3.1 設計方針と索引の全体構成

索引の設計では，「実使用時の検索時間を短縮する」という方針を採用した[21]．これは，ユーザの使用する可能性の高い検索語の処理を優先的に高速化することを意味する．この方針を採用したのは，低い誤検索率と小さい索引サイズという条件のもとでは，すべての検索語の処理を一律に高速化することは難しいが，頻度情報を用いることで一部の検索語の処理を高速化することは可能だからである．

本論文で提案する部分文字列索引は，前章で示した2つの異なる形式を組み合わせた構成をとる（後述の図2参照）．

- 基本エントリ：単一文字および二文字組を組み合わせた固定長部分文字列索引
- 拡張エントリ：長さ3以上の高出現頻度の部分文字列で構成する可変長部分文字列索引

この構成では，長い検索語のうち，頻度の高いものは拡張エントリの効果により検索時間が大幅に短縮される．一方，頻度の低い検索語は，基本エントリのみで処理されるが，基本エントリに新たに導入したハッシュ法の効果により検索時間はこれまでより短縮される．以下，各エントリの構成を説明する．

3.2 基本エントリの構成

基本エントリは，単一文字索引と文字種分割型出現頻度ハッシュ法という新たなハッシュ法を用いた二文字組索引の組合せである[21]．

3.2.1 文字種分割型出現頻度ハッシュ法

文字種分割型出現頻度ハッシュとは，文字種ごとのハッシュ法に出現頻度ハッシュを用いる手法である．その結果，索引サイズを増大させることなく高い検索精度を達成できる．

二文字組に対する出現頻度ハッシュでは，二文字組ごとにエントリを指定する参照テーブルが必要である．しかし，日本語では，字彙が約7000なので参照テーブルの大きさも7000*7000 = 4900万程度となる．そこで，我々は文字ごとに出現頻度ハッシュを施した後，結果のハッシュ値の組からエントリを計算することとした．この方式では，字彙分の大きさの参照テーブルを用意すればよい☆．さらに，二文字組索引に用いる文字種は検索語としてよく使用される漢字・片仮名等に限定できるので，参照テーブルをさらに小型化することも可能である．

本論文で提案する索引では，出現頻度ハッシュの特性を生かして，検索の高速化も同時に達成した．日本語では文字の出現頻度の差はきわめて大きいので，単一文字が1つのハッシュエントリを占有することがある．占有している文字を「占有文字」，占有されているハッシュエントリを「占有ハッシュエントリ」と呼ぶ．検索文字列に占有文字が含まれている場合，占有文字を含む二文字組に対応する二文字組エントリはこの文字の存在を確実に保障するので，この文字に対応する単一文字エントリを検索処理に用いる必要はない．したがって，ある文字が占有文字か否かが判定できるようハッシュ用の参照テーブルを作成すれば（具体的方法は次項に示す），検索時に処理すべきエントリ数を減少させ，検索を高速化できる．

たとえば，「携帯」という2文字の検索語が入力された場合，従来方式では「携」「帯」から決定されるハッシュエントリの組 (e_i, e_j) から算出される二文字組エントリと，「携」「帯」から直接決定される2つの単一文字エントリの計3つのエントリを用いて検索を行う．ここで，出現頻度ハッシュを導入し，参照テーブルにおいて e_i には「携」…「策」の複数の文字が対応するが，e_j は占有ハッシュエントリで「帯」のみが対応したとする．この場合，二文字組エントリ (e_i, e_j) には複数の二文字組「携帯」…「策帯」が対応し，このエントリが「帯」の存在を保証するので，単一文字エントリ「帯」は検索処理に不要となる．すなわち，2つのエントリのみで検索が実施でき，検索は高速化される．

3.2.2 ハッシュ用参照テーブルの作成

ハッシュ用参照テーブルの作成では，まず，ハッシュエントリ数を適当に決定し，次に，ハッシュエントリの頻度（そのハッシュエントリに割り当てられる文字の頻度の総和）ができるだけ均一になるように各文字のハッシュ先を決定する．このような割当て先の決定問題はNP完全なので，最適解を得るには時間がかか

☆ 参照テーブルを小型化する方法として，文字ごとに出現頻度ハッシュを施した後，ハッシュされた結果の組ごとの頻度に基づいて最終的なエントリを決定する二段階方式が提案されている[14]．しかし，二段階方式には参照テーブルが複数個必要であり，テーブル作成も複雑であるという問題がある．

```
1 minid=N-1;
2 for ( c=0; c<C; c++ ) {
3   if ( n<N-1 ) {
4     map[chr[c].code]=entry[n].id;
5     entry[n].total+=chr[c].count;
6     n++;
7   } else {
8     map[chr[c].code]=entry[N-1].id;
9     tmp_id=entry[N-1].id;
10    tmp_total=entry[N-1].total+chr[c].count;
11    for ( m=N-2; m>=0; m-- ) {
12      if ( entry[m].total>=tmp_total ) {
13        for ( l=N-2; l>m; l-- ) {
14          entry[l+1].id=entry[l].id;
15          entry[l+1].total=entry[l].total;
16        }
17        break;
18      }
19    }
20    entry[m+1].id=tmp_id;
21    entry[m+1].total=tmp_total;
22    if ( m<minid )
23      minid=m;
24  }
25 }
```

図1 参照テーブルの作成アルゴリズム
Fig. 1 Lookup table generation algorithm.

る．そこで，(1) 文字を頻度の降順にソートする，(2) 頻度の高い文字から順にハッシュエントリのなかで割り当てられた文字の頻度の合計が最少のものに割り当てる，というヒューリスティック解法を採用した．

そのアルゴリズムを図1に示す．ここで，C は字集数，N はハッシュエントリ数である．入力は，chr (文字コードと頻度☆を表すメンバ code，count を持つ構造体の C 個の配列を頻度の大きい順にソートしたもの) である．出力は，参照テーブルである map で，文字コード code に対応するハッシュエントリ ID (0～N-1) は map[code] で得られる．entry はハッシュエントリの ID と頻度を表すメンバ id，total を持つ構造体の N 個の配列で，entry[n].id = n となるよう初期化しておく．

図1において，2行目の for ループで文字の割当て先を決定する．4～6行目は頻度の高いハッシュエントリ数分の文字を各ハッシュエントリに1つずつ割り当てる処理，8～23行目は残りの文字をその時点で割り当てられている頻度が最少のハッシュエントリに割り当てる処理である．後者では，8行目で文字の割当て先を決定した後，9～21行目で配列 entry をハッシュエントリの頻度順にソートし直している．このアルゴリズムでは占有ハッシュエントリは ID から判定でき，ID が 0～minid のものが占有ハッシュエントリである．なお，このアルゴリズムで作成された参照テーブルの例を付録の表4に示す．

☆ このアルゴリズムは，頻度0の文字を，割り当てられた頻度が最小であるハッシュエントリに集中的に割り当てる．登録文書が事前に得られない場合，未出現文字が特定のハッシュエントリに集中することは好ましくないので，このような集中を防ぐべく文字の頻度計数における出現頻度の初期値は1とした．

3.3 拡張エントリの構成

拡張エントリは，長さ3以上の高出現頻度の部分文字列で構成する可変長部分文字列索引である[20),21)]．

2.2節で述べたように可変長部分文字列索引を日本語に直接適用することは困難である．しかし，本手法では固定長部分文字列索引と組み合わせることで，大規模部分文字列辞書に起因する記憶負荷と辞書アクセスによる速度低下という問題点を回避した．すなわち，固定長部分文字列索引と組み合わせることにより，任意の検索語は固定長部分文字列索引だけでも処理可能となるので，部分文字列辞書の大きさは自由に選択できる．そこで，辞書がメモリ上にロード可能な大きさになるように登録する部分文字列数を限定することで，記憶負荷を減少し，辞書アクセスを高速化する．

しかし，単純に辞書エントリ数を削減することには問題がある．部分文字列の頻度は一般的に短いものが高いので，辞書を小さくすると短い部分文字列のみが選択され，検索を高速化できないからである．そこで，本手法では，登録する部分文字列を固定長部分文字列より長い3文字以上のものに限定し，辞書を小さくしても長い部分文字列が選択されるようにして，検索処理を高速化した．

部分文字列辞書は，検索対象文書における部分文字列の頻度を測定し，頻度の高いものを辞書エントリ数だけ選択することで自動作成できる．その際，検索に用いられる文字列は同一文字種から構成され，しかも漢字あるいは片仮名から構成されることが多いので，漢字あるい片仮名のみから構成される部分文字列の頻度を計数することとした．この方式では，極端に長い部分文字列が生じないので，頻度計数を効率化できる．なお，この方式で生成された部分文字列辞書の例を付録の表5に示す．

3.4 登録・検索処理

本論文で提案する部分文字列索引における登録・検索処理を説明する．図2の中央部に示したように，単一文字索引および文字種分割型出現頻度ハッシュを用いた二文字組索引から成る基本エントリ，および可変長部分文字列索引から成る拡張エントリから全体は構成される．この図では，各エントリは1行で表されており，1/0 で対応する文書にそのエントリが出現した

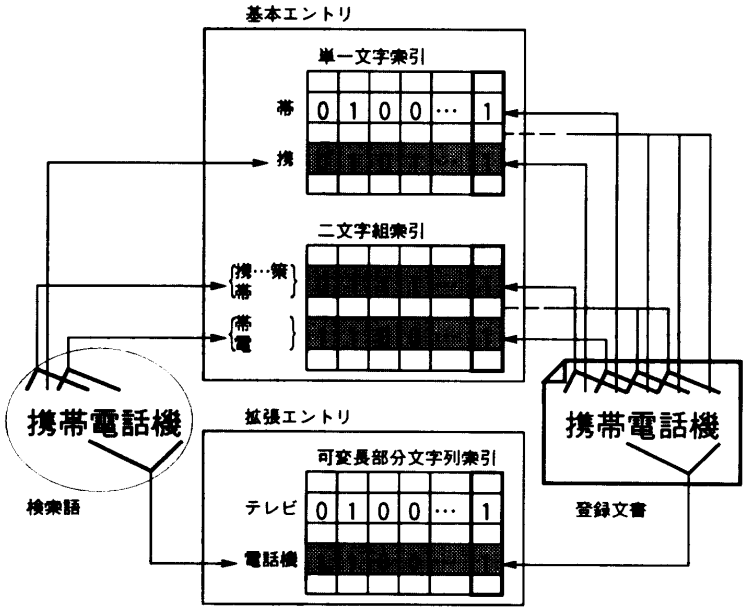

図2 本手法による登録・検索処理
Fig. 2 Registration and retrieval processes using the proposed method.

か否かを示している．単一文字索引では，文字がエントリと一対一に対応する．二文字組索引では，"{}"で囲まれた上側の文字集合と下側の文字集合から生成される二文字組がエントリに対応する．この例の二文字組索引において，上側のエントリには「携帯」…「策帯」が，下側のものには「帯電」のみが対応する（ここでは「帯」「電」が占有文字であるとしている）．可変長部分文字列索引では，「テレビ」「電話機」等の高頻度文字列がエントリと一対一に対応する．

文書登録では，抽出された全エントリの出現を索引に記録する．図2の右側が文書の登録の様子であり，「携帯電話機の…」という文書内容から，可変長部分文字列「電話機」，二文字組「携帯」「帯電」「電話」…，単一文字「携」「帯」「電」…が抽出され，対応するエントリの末尾に記録される．

文書検索では，検索語から抽出されるエントリのなかから他のエントリに包含されるものを除外したエントリをすべて含む文書を検索結果とする．図2の左側が検索語「携帯電話機」の検索の様子を示す．この例において，検索語から抽出されるエントリで実際に使用されるのは可変長部分文字列「電話機」，二文字組「携帯」「帯電」，単一文字「携」の4つだけになる．このエントリ数の減少により，検索処理は高速化される．

4. 評価と考察

4.1 評価法

本論文で提案した部分文字列索引の有効性を評価・確認するため，実験を行った．評価に用いたのは特許の要約文であり，1件が平均70文字（140バイト）のものを100,000件用意した（総計14Mバイト）．検索語としては，長さが2，4，6，8，10の漢字および片仮名の文字列をそれぞれ30ずつ，合計600用意した．これは，上記要約文中の単語からランダムに選択したものである．

性能評価は検索精度・検索時間・索引サイズの3点で行った．検索精度には，下式で算出される False Drop Rate[3]（以下 fdr と書く）を使用した☆．

$$fdr = \frac{\text{検索文字列を含まないが検索された文書数}}{\text{検索文字列を含まない文書数}}$$

測定にはSUN SPARCstation 20を用いた．検索速度の測定では，ディスクキャッシュが空の状態から検索処理を終了するまでの応答時間を測定した．

検索時間・索引サイズは，各エントリに対応する文書IDリストの管理形式に大きく依存する．今回の実験では，文書IDリストを文書IDのビット列で表現し，それをランレングス法で圧縮して固定長レコードで管理した[13]．文書IDリストの管理には，64バイトと2048バイトの2種類の大きさのブロックを併用す

☆ 意味的検索の精度評価では，検索文書の意味的な適合性判定（relevance judgment）に基づく再現率・適合率を使用するのが一般的である[23]．しかし，本論文では，検索文字列を含む文書を特定する表層的検索を対象としているので，シグニチャーファイルの誤検索率の評価で一般的に用いられている fdr を使用した．

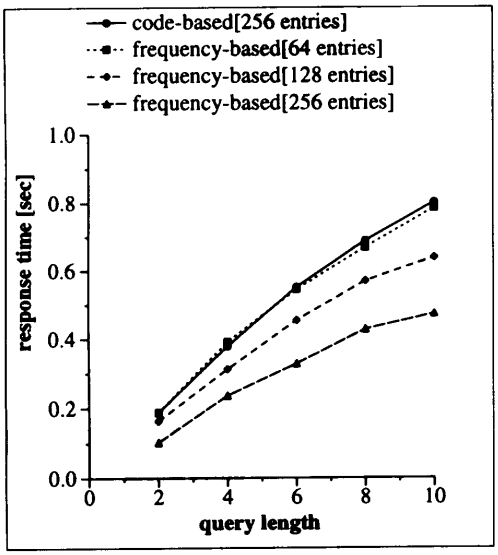

図3 検索時間の測定結果（漢字）
Fig. 3 Retrieval response time (Kanji).

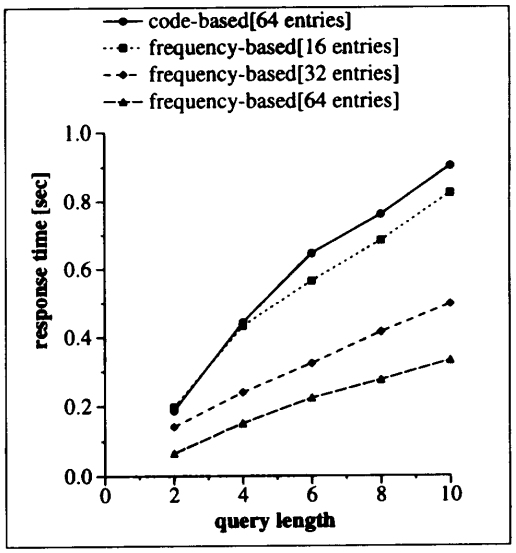

図4 検索時間の測定結果（片仮名）
Fig. 4 Retrieval response time (Katakana).

る方式を採用した[20),21)].

4.2 基本エントリの評価結果

まず，基本エントリの性能評価を行う．ここでは，文字種依存型文字コードハッシュと3.2.1項で提案した文字種分割型出現頻度ハッシュ方式を比較し，後者の有効性の確認を行う．ハッシュエントリ数をパラメータとして測定を行った．漢字と片仮名に属する文字数の相違を考慮し，二文字組に対するハッシュエントリ数は，漢字では16，32，64，128，256，片仮名では16，32，64とした．なお，索引エントリ数は対応するハッシュエントリ数の2乗となる．

(1) 検索時間

検索語長と検索時間の関係を図3（漢字）と図4（片仮名）に示す．いずれの場合も，ハッシュエントリ数の増加により検索時間が短縮されることが確認できる．これは，ハッシュエントリ数を増加することにより占有文字も増加するので，検索時に使用される索引エントリ数が減少するためである．実際，ハッシュエントリ数の増加にともない占有文字数は，漢字では0，0，3，21，97，片仮名では2，18，57と増加している．漢字と比較して，片仮名の方が高速化の効果が大きい．これは，字彙（漢字6353，片仮名数87☆）に対する占有文字の割合が大きいためである．

図示していないが，文字コードハッシュ法でもエントリ数の増加により検索時間は若干短縮している．漢字検索語に対する平均検索時間は，エントリ数16で0.586 sec，256で0.521 sec，片仮名検索語では，エントリ数16で0.627 sec，64で0.588 secであった．これは，エントリ数を増やすことで各エントリに含まれる文書数が減少し，検索時にアクセスされる文書IDリストが短くなるためである．しかし，出現頻度ハッシュと比較して高速化の効果は非常に小さい．

(2) 検索精度

検索精度の測定結果を図5（漢字）と図6（片仮名）に示す．誤検索率は文字コードハッシュ法よりも出現頻度ハッシュ法の方が低く，検索速度だけでなく検索精度の点でも出現頻度ハッシュ法が優れていることが確認できる．なお，漢字と比較して片仮名の方が検索精度の改善率が高い．これは，片仮名においては，字彙数が少ないので字彙数に対する占有文字の割合が高くなり，それだけ誤検索が起こりにくくなるからである．次に，出現頻度ハッシュ法における検索語長と検索精度の関係を図7に示す．ハッシュエントリ数は漢字64，片仮名32とした．検索語を含まない文書が誤検索されるには，検索語から抽出される索引エントリのすべてが，その文書中の検索語以外の部分から抽出されなければならない．しかし，検索語が長くなるに従って抽出される索引エントリ数は増大するため，このような偶然の一致は減少し，誤検索も減ることが期待できる．図7から，この性質が確認できる．

一般に検索語が長くなると，その単語を含む文書数も増大する傾向がある．本実験でも，正解文書数の平均値は，検索語長2では3376なのに対し，検索語長10では2.13であった．長い検索語に対する誤検索率が低いという前述の性質は，正解文書が少ない場合には

☆ 「ー」（長音記号）はJISでは記号に分類されているが，ここでは片仮名として扱った．

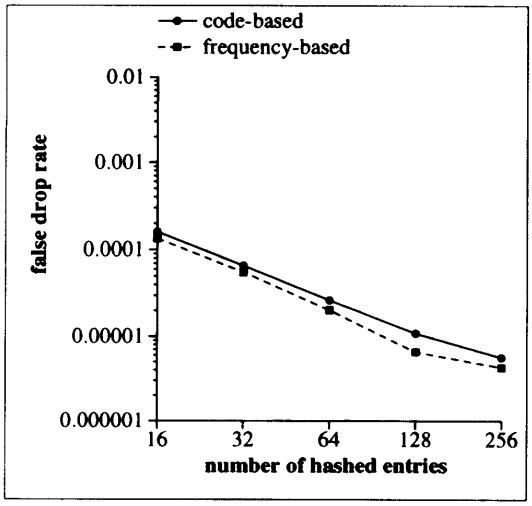

図5 誤検索率の測定結果（漢字）
Fig. 5 False drop rate (Kanji).

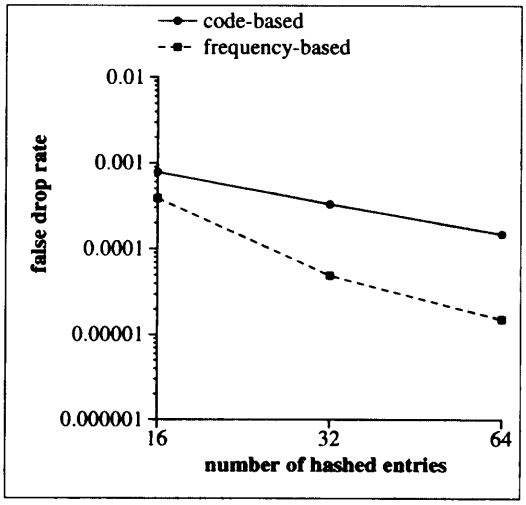

図6 誤検索率の測定結果（片仮名）
Fig. 6 False drop rate (Katakana).

誤検索文書も少なく，検索された文書に含まれる誤検索文書数の割合が極端には悪化しないことを意味し，ユーザにとって望ましい性質といえる．

(3) 索引サイズ

索引サイズを**表1**に示す．この表において，漢字・片仮名で示されているのは二文字組索引部分であり，それ以外の文字種に対する二文字組☆および単一文字索引はその他の欄に合計した値を示している．予想どおり，ハッシュエントリ数の増加につれて索引サイズは増大している．文字コード法と出現頻度法を比較すると，出現頻度の方が若干大きくなっており，差はハッシュエントリ数の増加にともない増大している．しかし，両者の差はせいぜい数ポイントで，きわめて小さい．

☆ 漢字・片仮名以外に関しては，ハッシュエントリ数を16に固定し，従来型の文字コードハッシュを適用した．

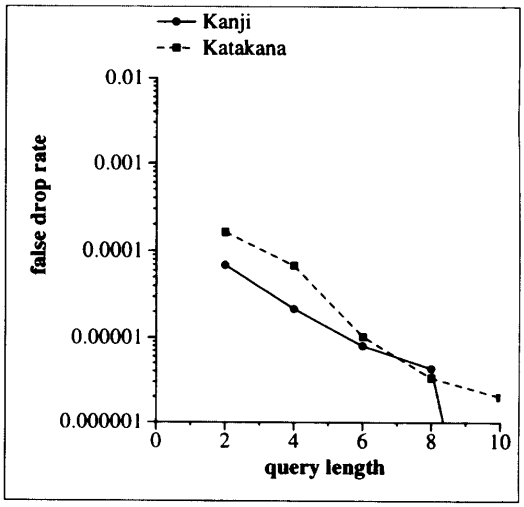

図7 誤検索率の測定結果（片仮名）
Fig. 7 False drop rate (Katakana).

表1 基本エントリの索引サイズ
Table 1 Index sizes for fixed-length entries.

	ハッシュエントリ数	索引サイズ [MB] (対文書比 [%])	
		文字コード法	出現頻度法
漢字	16	1.15 (8.21)	1.15 (8.21)
	32	1.70 (12.1)	1.72 (12.3)
	64	2.27 (16.2)	2.30 (16.4)
	128	3.11 (22.2)	3.17 (22.6)
	256	4.90 (35.0)	5.28 (37.7)
片仮名	16	0.56 (4.01)	0.58 (4.14)
	32	0.71 (5.07)	0.77 (5.50)
	64	0.90 (6.43)	0.95 (6.79)
その他	—	3.66 (26.1)	—

4.3 拡張エントリの評価結果

次に，拡張エントリの効果を測定した．以下では，基本エントリは漢字に対するハッシュエントリ数は64，片仮名は32の出現頻度ハッシュ法とし，漢字・片仮名それぞれの拡張エントリ数を128，256，512とした．

(1) 検索時間

検索時間の測定結果を**図8**（漢字）と**図9**（片仮名）に示す．文字列長が4以上では検索時間が短縮されており，高速化の効果が確認できる．文字列長が2の場合に短縮されないのは，拡張エントリの部分文字列は長さ3以上だからである．また，拡張エントリ数が多いほど高速化の効果は大きい．漢字と片仮名を比較すると，片仮名の方が高速化の効果が大きい．これは，異なり単語数が漢字と比較して片仮名の方が少ないので，拡張エントリ中の部分文字列が検索文字列に含まれる確率が高くなるためである．

単語長は漢字2.5文字に対し片仮名は4.3文字であった．この点を考慮すると，漢字の検索時間が遅いことも大きな問題ではないと考えられる．

5. おわりに

本論文では，日本語文書のための効率的部分文字列索引の構成法を提案した．本手法は，頻度情報を有効利用することで，誤検索率・索引サイズを悪化させることなく検索高速化を実現した．本手法の第一の特徴は，固定長部分文字列索引に文字の出現頻度に基づいた文字種分割型出現頻度ハッシュ法を採用したことである．文字レベルの頻度情報を用いることで，誤検索を低減するだけでなく，検索処理を高速化した．第二の特徴は，固定長部分文字列索引と可変長部分文字列索引を組み合わせた構成とし，索引に用いる可変長部分文字列を出現頻度に基づいて選択したことである．部分文字列レベルの頻度情報を用いることで，出現頻度が高く長い部分文字列の検索処理をさらに高速化した．

本手法の有効性を確認するため，特許要約文10万件（14 MB）を用いた検索時間・検索精度・索引サイズの評価実験を行った．その結果，従来方式と比較して索引サイズをほとんど増加させることなく，検索時間を短縮し，検索精度を改善できることが確認できた．

今後の課題として，1つは，新聞・雑誌・論文などでの性能評価に基づく文書タイプの影響の調査があげられる．これは，本手法は頻度情報を用いているので，文書タイプによる文書長，出現頻度等の相違が性能に影響を与える可能性があるからである．もう1つは，文書検索手法としては全文検索のような完全照合法よりも検索文書を検索条件に対する類似度でランキングする最適照合法の有効性が高いことから[17),23)]，本索引を用いた効率的ランキング手法の検討があげられる．

謝辞 本研究を進めるにあたり多くのご協力をいただいたリコー情報通信研究所の岩崎雅二郎氏，林大川氏ならびに，本論文をまとめるにあたり貴重な意見をくださった春日正男氏（現在宇都宮大学）および江尻公一氏に感謝いたします．

参考文献

1) Barton, I. and Creasey, S.: An Information Theoretic Approach to Text Searching in Direct Access Systems, *Comm. ACM*, Vol.17, No.6, pp.345–350 (1974).
2) Faloutsos, C.: Access Method for Text, *Comput. Surv.*, Vol.17, No.1, pp.49–74 (1985).
3) Faloutsos, C. and Christodoulakis, S.: Signature Files: An Access Method for Documents and Its Analytical Performance Evaluation, *ACM Trans. Office Information Systems*, Vol.2, No.4, pp.267–288 (1984).
4) Frakes, W. and Basza-Yates, R.: *Information Retrieval: Data Structures and Algorithms*, Prentice-Hall, New Jersey (1992).
5) Fujii, H. and Croft, W.B.: A Comparison of Indexing Techniques for Japanese Text Retrieval, *Proc. 16th ACM SIGIR Conf.*, pp.237–246 (1993).
6) 藤井洋一，望月泰行，鈴木克志，丸山冬樹：全文検索システムにおける文字成分表の作成手法，第48回情報処理学会全国大会論文集（4），pp.159–160 (1994).
7) 藤澤浩道，絹川博之：情報検索における自然言語処理，情報処理，Vol.34, No.10, pp.1259–1265 (1993).
8) 福島俊一，田村美保子，垣原睦治：全文検索用文字成分表の一圧縮方式，第47回情報処理学会全国大会論文集（4），pp.83–84 (1993).
9) 古瀬一隆，浅田一繁，飯沢篤志：DBMSへのシグネチャファイルの実装について，信学技報，DE94-58, pp.23–30 (1994).
10) 畠山 敦，浅川悟志，加藤寛次：ソフトウェアによるテキストサーチマシンの実現，情報処理学会研究会報告，FI25, pp.19–26 (1992).
11) 稲葉光昭，野口直彦，管野祐司，倉知一晃：日本語文書に対する新しい索引検索方式～試作・実験および評価～，第50回情報処理学会全国大会論文集（4），pp.43–44 (1995).
12) 岩崎雅二郎，小川泰嗣：テキストデータベースのための文字成分表を用いたプリサーチ，第45回情報処理学会全国大会論文集（4），pp.223–224 (1992).
13) 岩崎雅二郎，小川泰嗣：文字成分表による文字列検索の実現と評価，情報処理学会研究会報告，DBS97, pp.1–10 (1993).
14) 川下靖司，浅川悟志，坂田 順，畠山 敦：フルテキストサーチシステムBibliotheca/TSの開発(2)，第45回情報処理学会全国大会論文集（3），pp.241–242 (1992).
15) 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，Vol.J75-D-I, No.9, pp.836–846 (1992).
16) 中渡瀬秀一：統計的手法によるテキストからのキーワード抽出，信学技報，DE95-1, pp.9–16 (1995).
17) 仁木和久，田中克巳：ニューラルネットワーク技術の情報検索への応用，人工知能学会誌，Vol.10, No.1, pp.45–51 (1995).
18) 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，Vol.33, No.4, pp.404–412 (1992).

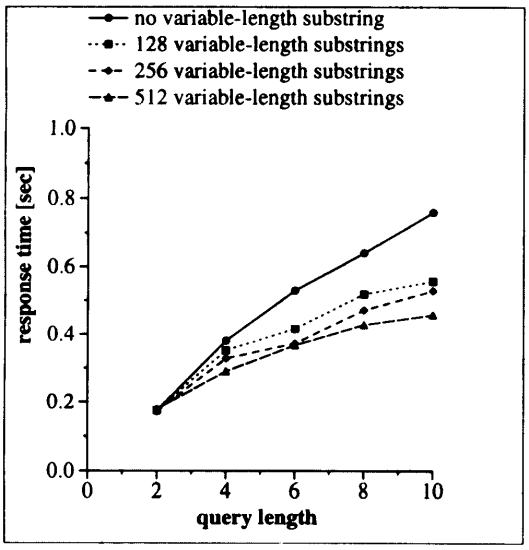

図8 拡張エントリ使用時の検索時間の測定結果（漢字）
Fig. 8 Retrieval response time using variable-length entries (Kanji).

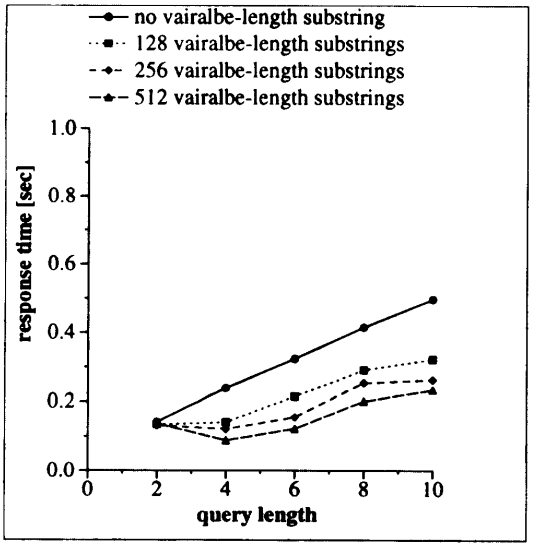

図9 拡張エントリ使用時の検索時間の測定結果（片仮名）
Fig. 9 Retrieval response time using variable-length entries (Katakana).

(2) 検索精度

検索語に拡張エントリが含まれていた場合，これまで複数の基本エントリで処理されていた文字列が単一のエントリとして処理されるようになるので，誤検索が減少する可能性がある．実際 fdr を比較すると，漢字では 2.067×10^{-5}（拡張エントリなし）が 1.713×10^{-5}（拡張エントリ数512），片仮名では 4.934×10^{-5}（拡張エントリなし）が 3.710×10^{-5}（拡張エントリ数512）と向上した．

表2 拡張エントリの索引サイズ
Table 2 Index sizes for variable-length entries.

拡張エントリ数	索引サイズ [MB] (対文書比 [%])	
	漢字	片仮名
128	0.15 (0.36)	0.18 (1.28)
256	0.22 (1.57)	0.27 (1.91)
512	0.38 (2.71)	0.56 (3.99)

(3) 索引サイズ

拡張エントリの大きさを**表2**に示す．この実験で採用した基本エントリの大きさは6.73 MBであり，漢字・片仮名の拡張エントリ数を512にした場合でも拡張エントリの大きさは合計0.94 MB（基本エントリの14.0%）と小さい．

4.4 考　　察

本手法では，漢字・片仮名に対するハッシュエントリ数あるいは拡張エントリ数をパラメータとして，索引の特性を調整することが可能である．索引の小型化を優先して，ハッシュエントリ数が漢字128，片仮名32の基本エントリのみで索引を構成した場合，従来方式と比較して，索引サイズを1.4%増加させるだけで，検索時間を18%（漢字），47%（片仮名）短縮，検索精度を40%（漢字），85%（片仮名）向上できる．高速化を優先して，ハッシュエントリ数が漢字64，片仮名32の基本エントリと1024個の部分文字列からなる拡張エントリで索引を構成した場合，索引サイズは14%増加するが，検索時間を34%（漢字），74%（片仮名）短縮，検索精度を35%（漢字），88%（片仮名）向上できる．

これらの結果から，今回提案した文字種分割型出現頻度ハッシュ法および拡張エントリによって，設計方針どおり検索時間を短縮できることが確認できた．さらに，提案手法が検索精度向上にも有効であることも確認できた．これは同一検索精度を達成するために必要な索引サイズを小さくできることをも意味する．以上より，本部分文字列索引は検索時間・検索精度・索引サイズの点で従来方式を上回ると結論できる．

漢字と片仮名を比較すると，字彙数の多い漢字の方が検索時間・検索精度の向上が少ない．しかし，検索精度（fdr 値）自体は片仮名より漢字の方が良いので，向上率が低くても問題ではない☆．検索時間に関しては，絶対値でも漢字が劣っている．しかし，漢字より片仮名の単語の方が一般に長く，特許要約文中の平均

☆ 部分文字列索引は検索範囲の絞り込みに使うのが一般的である[18),19)]．本手法で実現される $fdr \sim 10^{-5}$ は，英語文書を対象としたシグニチャーファイル[2)]と同等であり，実用上十分な値といえる．

19) 小川泰嗣：テキストデータベース技術の最近の動向，アドバンストデータベースシステムシンポジウム 93, pp.153–162 (1993).
20) 小川泰嗣，岩崎雅二郎：全文検索のための文字成分表の改良，情報処理学会研究会報告，DBS99, pp.261–264 (1994).
21) Ogawa, Y. and Iwasaki, M.: A New Character-Based Indexing Method Using Frequency Data for Japanese Documents, *Proc. 18th ACM SIGIR Conf.*, pp.121–129 (1995).
22) Roberts, C.: Partial-Match Retrieval via the Method of Superimposed Codes, *Proc. IEEE*, Vol.67, No.12, pp.1624–1642 (1974).
23) Salton, G. and McGill, M.J.: *Introduction to Modern Information Retrieval*, McGraw-Hill, New York (1983).
24) Salton, G. and Smith, M.: On the Application of Syntactic Methodologies in Automatic Text Analysis, *Proc. 12th ACM SIGIR Conf.*, pp.137–150 (1989).
25) 玉置志津，藤原健史，西谷紘一：シグニチャ法を用いた日本語文書検索システム，第 50 回情報処理学会全国大会論文集 (4)，pp.39–40 (1995).
26) Witten, I., Moffat, A. and Bell, T.: *Managing Gigabytes: Compressing and Indexing Documents and Images*, Van Nostrand Reinhold (1994).
27) Zipf, G.: *Human Behavior and the Principle of Least Effort*, Addison-Wesley (1949).

付録 頻度情報サンプル

評価に用いた特許要約文から得られた頻度情報をサンプルとして示す．**表 3** は文字ごとの出現頻度，**表 4** は漢字・片仮名ともにハッシュエントリ数を 64 として作成した参照テーブル，**表 5** は拡張エントリに使用した部分文字列である．表 3 では，3.2.2 項の脚注で述べたように文字の出現頻度は 1 から数えるので，漢字の 1958 位の「廷」…「瑤」，および片仮名の 84 位の「ヱ」…「ゥ」は評価用文書には 1 回も出現しなかったことになる．表 4 の参照テーブルにおいて，ID はハッシュエントリ ID を指す．この表では，文字とハッシュエントリが一対一対応しているものが占有文字/占有ハッシュエントリであり，この場合，漢字では 3 個，片仮名では 57 個が占有文字/占有ハッシュエントリになっている．

表 3 文字の出現頻度
Table 3 Character's frequencies.

	漢字		片仮名	
順位	出現頻度	文字	出現頻度	文字
1	73748	的	101119	ー
2	60397	目	60507	ン
3	46838	電	48261	ツ
4	39564	置	45332	ト
5	36400	用	45140	イ
…	…	…	…	…
84	10047	加	1	ヱ，ヲ，ヵ，ヴ
…	…	…		
1958	1	廷，…，瑤		

表 4 参照テーブルの例
Table 4 Sample lookup table.

	漢字		片仮名	
ID	出現頻度	文字	出現頻度	文字
0	73748	的	101119	ー
1	60397	目	60507	ン
2	46838	電	48261	ツ
3	45917	変，圧，所，…	45332	ト
…	…	…	…	…
56	45916	光，常，堤，…	2372	ハ
57	45916	信，移，書，…	2289	ワ，ゴ
…	…	…	…	…
63	45916	気，状，付，…	2043	ゲ，ゾ，ュ，…

表 5 部分文字列辞書の例
Table 5 Sample substring dictionary.

	漢字		片仮名	
順位	出現頻度	文字	出現頻度	文字
1	3084	感光体	10382	ヘッド
2	2824	表示装置	9343	データ
3	2484	周波数	5330	テープ
4	2483	記録媒体	4458	トナー
5	2312	電子写真	4395	インク
…	…	…	…	…
512	166	結着剤	112	バリテイ

(平成 7 年 10 月 19 日受付)
(平成 8 年 7 月 4 日採録)

小川 泰嗣（正会員）
昭和 37 年生．昭和 63 年東京大学大学院工学系研究科情報工学専攻修士課程修了．同年 (株) リコー入社．情報検索・データベース等の研究開発に従事．現在，情報処理学会・データベースシステム研究会・連絡委員．言語処理学会，IEEE，ACM 各会員．