テクニカルノート

特徴ベクトルによる全文検索の一改善法

有田 健[†] 森田和宏[†]
溝渕昭二[†] 青江順一[†]

特徴ベクトル法による全文検索の効率は，検索に不要な文書ブロックの転送をどれだけ排除できるかに依存するが，この排除率を保証したベクトル構成法の議論はない．本論文では，対象文書の文字列頻度を利用して，目標排除率を保証したベクトル構成法を提案する．同一ベクトル長では，提案手法が従来手法より排除率はつねに高くなることを，また排除率95%を達成する場合，提案手法のベクトル長は従来法の約1/6で実現できることを実証した．

A Method for Improving Full Text Search Using Signarure Vectors

TAKESHI ARITA,[†] KAZUHIRO MORITA,[†] SHOJI MIZOBUCHI[†] and JUN-ICHI AOE[†]

A well-known approach of searching texts is full text retrieval using signature vectors, but the method can not always remove efficiently redundant accesses of text blocks. This paper presents a method for constructing signature vectors holding the worst-case removing ratio. The concept of this approach is to define the bit position for arbitrary length strings. From the simulation results, it is shown that the vector length of the method presented is about 80% shorter than that of the traditional method under the same removing ratio.

1. はじめに

近年，電子化ファイルの検索要求はますます高くなってきており，特別な索引表を必要としないで，任意の文字列が検索できる全文検索は重要な技法である[1),3)]．特に，特徴ベクトルを使用した全文検索法は，多く利用されている[4)]．この手法は，分割された文書ブロック内の文字列情報に対応する特徴ベクトルからブロックベクトルを構成し，検索時にこのブロックベクトルを事前検査することで，探索の必要のないブロックの走査を排除する．日本語は分かち書きしないので，文字列から特徴ベクトルのビット位置を定義するハッシュ関数[2)]は，隣接する部分文字列の内部値から計算するのが一般的である．この手法で，ハッシュ関数を工夫し，排除率をいかに高くするかが重要な課題となる．しかし，文字の内部値，および文書中の文字列の出現確率は一様でないので，ハッシュ関数値に偏りが生じる．そのため，排除率が均一でなくなり，最悪の検索時間が予測できない問題点がある．

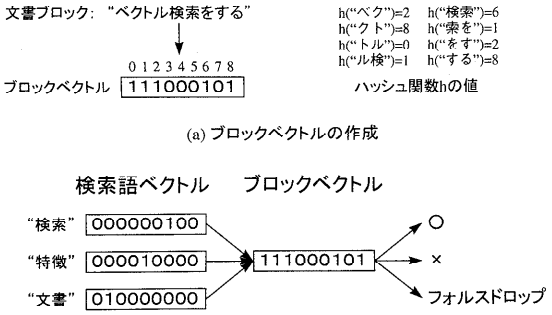
本論文では，最悪の排除率を考慮した特徴ベクトルの構成手法を提案し，有効性を実験により評価する．

2. 特徴ベクトルを用いた検索

特徴ベクトルの検索例を**図1**に示す．

図1(a)は，文書ブロックの文字列に対するブロックベクトルを示し，h("検索")$=6$よりベクトルの6番目のビットは1となる．図1(b)は，検索例である．検索語の特徴ベクトル（検索語ベクトル）とブロックベクトルの論理積が検索語ベクトルと一致しなければ，検索語は文書ブロックに存在しない[4)]．たとえば，"特徴" の検索語ベクトルはブロックベクトルとの照合により，文書ブロックの検索を排除できる．しかし，検索語 "文書" のベクトル照合は成功するが，"文書" は文書ブロックには存在しない．このことをフォルスドロップと呼び，この率が少ないほど，全文検索の速度は向上する．この率はハッシュ関数値や文字列の出現確率が一様である条件下で，理論的[2),4)]に評価され

† 徳島大学工学部知能情報工学科
Department of Information Science and Intelligent Systems, The University of Tokushima



(a) ブロックベクトルの作成

(b) 検索語ベクトルの照合

図 1 特徴ベクトルによる検索の例

Fig. 1 The example of the retrieval using the signature vectors.

ているが，最悪の状況を予測するのは難しい．本論文では文書を構成する文字列の出現確率を利用して，ブロックの目標排除率を満足するベクトルの構成法を提案する．

3. 文字列の出現確率を用いた手法

3.1 概 要

文書ブロックの目標排除率を q，文字数を f，文字列 s の出現確率を p とし，s をブロックベクトルの1つのビットに対応させた場合，そのビットによってブロックが排除される確率は $(1-p)^f$ となり，これを1ビット排除率と呼ぶ．本手法では，ブロックの目標排除率 q を保証するために，すべての1ビット排除率を q 以上とするので，出現確率が高い文字列ほど長い文字列に拡張して1ビットに対応させる必要がある．しかし，文書全体のすべての文字列の出現確率（全体確率と呼ぶ）を決定するのは，効率的でないので，本論文では，文書の一部分の文字列の出現確率（部分確率と呼ぶ）を全体確率の近似値として，ベクトルを効率的に構成する方法を提案する．

文字列 s の部分確率 $p(s)$ を図 2 で定義するが，s の文書中の分布は一様でないので，局所的な範囲では部分確率が全体確率より極端に大きくなることも考えられる．そこで，部分確率を計算する最低計測長 m を設定し，この問題点を解決する．

本手法の概要は次にまとめられる．

(a) 文書を頭から走査し，文字列 s の頻度を計測するが，最初の文字列は，1文字からスタートする．

(b) 最低計測長を超えたところで，目標排除率から決定される最大出現確率より大きい部分確率 $p(s)$ を持つ文字列 s があった場合，以降の走査からは文字列 s にすべての文字 a を連鎖した文字列 sa を頻度計測の対象文字列に加える．

(c) 文書の終わりに到達するまで (b) を繰り返す．

(d) 文字列 s を部分確率 $p(s)$ によりベクトルのビットに割り当てる．その際，$p(s)$ の小さいものは複数の文字列を同じビットに割り当てる．

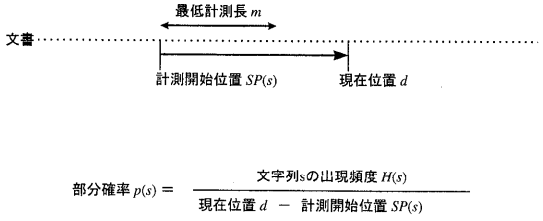


図 2 部分確率の定義

Fig. 2 The definition of the part probability.

3.2 ベクトル構成アルゴリズム

3.1 節で述べた手順 (a) ～ (c) の処理を文字列確定アルゴリズム，手順 (d) をビット割当てアルゴリズムとして，以下に与える．

文字を要素とする集合を I と定義する．部分確率の計算対象となっている文字列の集合を確率計測文字列集合 U とし，r を目標排除率を超えない文字列の最大出現確率とする．集合 U の要素の中で最長の文字列長を t とする．

[文字列確定アルゴリズム]

[入力] 入力文書=$c_1c_2\ldots c_n$

[出力] 確率計測文字列集合 U，出現確率 p

[方法]

手順 (1-1) { 初期設定 }

$U \leftarrow I$ とする．すべての文字 $a \in U$ に対して，頻度関数 $H(a)$ と計測開始位置 $SP(a)$ を0に初期化する．現在の計測位置 d と最大文字列長 t を1に初期化する．

手順 (1-2) { 計測対象文字列長の初期設定 }

計測対象の文字列長を制御する変数 i を1とする．

手順 (1-3) { 頻度計算と出現確率の判定 }

計測対象文字列を $s \leftarrow c_{d-i}c_{d-(i-1)}\ldots c_d$ とする．$s \in U$ でなければ，手順 (1-5) へ．$s \in U$ ならば $H(s) \leftarrow H(s)+1$ とし，$d - SP(s) > m$，かつ $p(s) = H(s)/(d-SP(s)) > r$ ならば，手順 (1-4) へ．そうでなければ手順 (1-5) へ．

手順 (1-4) { 確率計測文字列集合 U の拡張 }

$A = \{sa | a \in I\}$ を作成する．$A \subset U$ ならば手順 (1-5) へ，そうでないとき $U = U \cup A$ とし，$x \in A$ なるすべての x について $SP(x) \leftarrow d$，$H(x) \leftarrow 0$ と初期設定する．ここで，$i+1 > t$ ならば，$t \leftarrow i+1$ とする．

手順 (1-5) { 入力文書の計測対象文字列 s の拡張 }

$i \leftarrow i+1$ として，$i < t$ ならば手順 (1-3) へ．そうでなければ，$d \leftarrow d+1$ とする．$d > n$ ならば，計測が終了したので，集合 U と出現確率 p を出力する．そうでなければ手順 (1-2) へ進む．(アルゴリズム終)

集合 U は，**図 3** に示す構造体 C のリスト表現で実装できる．C は各文字の計測開始位置 SP，出現頻度 H，次の C へのリンク情報 Next を要素に持ち，先頭の C からリンクでたどる文字を連鎖した文字列が U の要素となる．文字列 "の" 拡張状況を説明する．図 3 (a)，(b) では，文字位置 $d = 2048$ のときに文字列 "の" の確率が r を超え，手順 (1-4) での文字列拡張に対応する．同様に，$d = 7536$ と $d = 9532$ のときに，"のが" と "のに" の確率が R を超え，文字列が拡張される（図 3 (c)，(d)）．

[ビット割当てアルゴリズム]

[入力] 集合 U，出現確率 $p(s)$，最大出現確率 R

[出力] 特徴ベクトルのビット位置 j に割り当てる文字列集合 $B(j)$．ただし，ビット長は処理中に決定される．

[方法]

手順 (2-1) { ビット位置変数 j の初期化 }

$j \leftarrow 0$ とする．

手順 (2-2) { ビット位置 j への文字列の割当て }

$s \in U$ なる任意の s を抽出し，U から削除する．$B(j) \leftarrow \{s\}$ とする．$sum \leftarrow p(s)$ とする．

手順 (2-3) { ビット位置 j への次の文字列の割当て }

$sum + p(s') \leq r$ なる $s' \in U$ に対して，U から s' を除去し，$B(j) \leftarrow B(j) \cup \{s'\}$，$sum \leftarrow sum + p(s')$ とする．この手順を候補 s' がなくなるまで繰り返す．

手順 (2-4) { ビット位置の更新と割当て終了判定 }

U が空集合ならば終了．そうでなければ $j \leftarrow j+1$ とし，手順 (2-2) へ．　(アルゴリズム終)

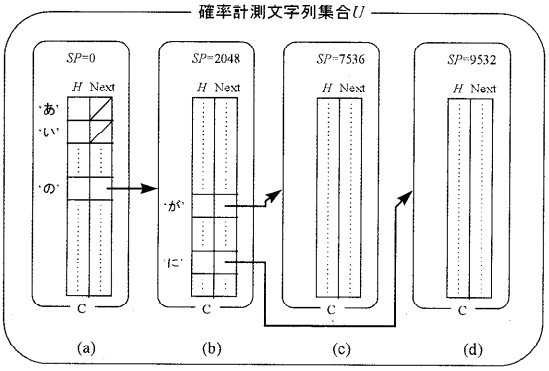


図 3 アルゴリズムの例
Fig. 3 An example of the presented algorithm.

手順 (2-3) で，$sum + p(s') \leq r$ の条件を満足する s' を $B(j)$ の候補とするので，j 番目のビットが r を超えないこと，すなわち目標排除率を満足することは，明らかである．

4. 評　　価

実験に使用した言語は Microsoft Visual C++，マシンは GATEWAY G6-200，データは広辞苑第 4 版（岩波書店）の文書 28 MB である．文書ブロックサイズ 256，512，1024 バイトに対して，目標排除率とベクトル長の関係を**図 4** の実線で示す．なお，最低計測長は 250 KB から 5 MB に変化させたが，ベクトル長は最大 2%しか変動しないので，500 KB のものを示す．図 4 より，目標排除率が 70%から 90%になると，ベクトル長は文書ブロック長の 40%から 70%へと緩やかに増加するが，それ以上の増加は急峻となり，目標排除率 93%付近で文書ブロックと同等のサイズが必要となることが分かる．

提案したベクトル構成法は，文書走査が 1 回で済む点で効率的だが，同じ目標排除率を得るには近似的に構成されたベクトル長は全体確率のものより長くなる．図 4 の破線は全体確率によるベクトル長を示す．次の**図 5** で評価するように実際の排除率は目標排除率 80%程度でも十分高いので，その 80%付近の全体確率に対するベクトル長の増加率は非常に少なく，提案手法は有効であるといえる．部分確率の計算時間は，目標排除率 70%から 95%に対して 2 ～ 3 分であり，全体確率の場合に対して 6 から 10 倍高速化にされた．

実際の検索結果による排除率と目標排除率の関係を図 5 に示す．提案手法で構成されたベクトル長と同じ長さのブロックベクトルを 2 文字連接のハッシュ関数値で作成した従来法と比較する．この関数は，2 文字の内部コードに重みを付与し，関数値が一様になるように工夫したものである．検索語として，無作為に抽出した名詞 100 個を用いた．また，ブロックサイズ

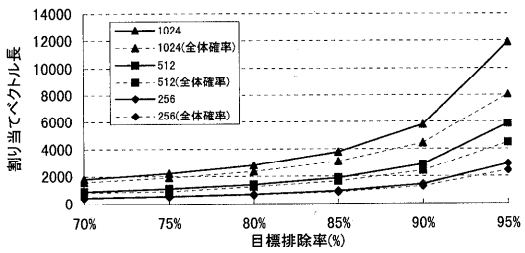


図 4 目標排除率と割当てベクトル長の関係
Fig. 4 The relation between the target removing ratio and the vector length.

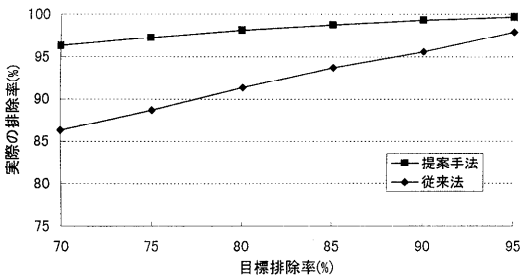


図 5 実際の検索による排除率
Fig. 5 The ratio of canceled blocks in searching keywords.

の変化に対する変動は1%以内となったので，サイズ512のものを示す．提案手法の実際の排除率が目標排除率よりもつねに高い値となっているのは，目標排除率を満足するビットが複数立つからである．図5より，目標排除率が70%で構成された場合でも，実際の検索では，96%以上の排除率が達成されており，従来法の86%を十分改善している．特に，従来法で95%以上の排除率を得るためには，提案手法のベクトル長の約6倍必要となるので，提案手法の有効性が分かる．

文字列 s からベクトル位置 i を計算するハッシュ関数 $h(s)=i$ は，ダブル配列[5]構造を使用した．集合 U の文字列数は約21万となり，約1MBのダブル配列が主記憶上に置かれた．この構造により文字列の長さに比例した計算量で $h(s)=i$ が決定でき，この計算時間は実際の検索時間の1%以下となり，問題とならなかった．

5. まとめと今後の課題

以上，文字列情報を用いた特徴ベクトルの作成法を提案し，その有効性を実証した．今後は，分野の異なる文書データによる実験・評価と考察が課題となる．

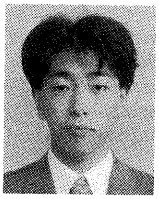
謝辞 実験に辞書データの使用を許可していただきました岩波書店に深謝する．

参 考 文 献

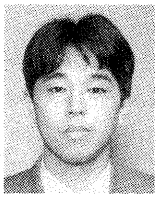
1) Aoe, J.: *Computer Algorithms – String Pattern Matching*, IEEE Computer Society Press (1994).
2) Harrison, M.C.: Implementation of the substring test by hashing, *Comm. ACM*, 14, pp.777–779 (1971).
3) 小川隆一，菊池芳秀，高橋恒介：フルテキスト・データベースの技術動向，情報処理学会論文誌，Vol.33, No.4, pp.404–412 (1992).
4) Faloutsos, C.: Signature Files, *Information Retrieval: Algorithms and Data Structures*, Frakes, W. and Baeza-Yates, R.A. (Eds.), pp.44–65, Prentice Hall, Englewood Cliffs, N.J. (1992).
5) Aoe, J.: An efficient digital search algorithm by using a double-array structure, *IEEE Trans. Softw. Eng.*, Vol.SE-15, No.9, pp.1066–1077 (1989).

(平成 9 年 7 月 30 日受付)
(平成 10 年 1 月 16 日採録)

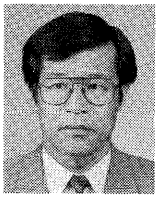
有田 健（正会員）
昭和 46 年生．平成 5 年徳島大学工学部知能情報工学科卒業．平成 7 年同大学院博士前期課程修了．現在同大学院博士後期課程在学中．情報検索，自然言語処理の研究に従事．



森田 和宏（正会員）
昭和 47 年生．平成 7 年徳島大学工学部知能情報工学科卒業．平成 9 年同大学院博士前期課程修了．現在同大学院博士後期課程在学中．情報検索，自然言語処理の研究に従事．



溝渕 昭二（正会員）
昭和 47 年生．平成 7 年徳島大学工学部知能情報工学科卒業．平成 9 年同大学院博士前期課程修了．現在同大学院博士後期課程在学中．自然言語処理の研究に従事．



青江 順一（正会員）
昭和 26 年生．昭和 49 年徳島大学工学部電子工学科卒業．昭和 51 年同大学院修士課程修了．同年同大学工学部情報工学科助手．現在同大学工学部知能情報工学科教授．この間コンパイラ生成系，パターンマッチングアルゴリズムの能率化の研究に従事．最近，自然言語処理，特に理解システムの開発に興味を持つ．著書「Computer Algorithms — Key Search Strategies」,「Computer Algorithms — String Matching Strategies」(IEEE CS press)．平成 4 年度情報処理学会「Best Author 賞」受賞．工学博士．電子情報通信学会，人工知能学会，日本認知科学会，日本機械翻訳協会，IEEE，ACM，AAAI，ACL 各会員．