3C-2

転置インデックスに位置情報を付与する新手法

安田 知弘† 岩山 真†

(株)日立製作所 中央研究所†

1. 背景と課題

現代社会では、膨大な量の電子文書が日々作成されており、それらを活用するための文書検索技術が不可欠である。文書検索の基本となる技術が、指定された索引語を含む文書を、与えられた文書集合から探索する技術である。ここに索引語とは、検索対象文書に含まれる単語や *N*-gram と呼ばれる短い固定長の部分文字列である[1]。高速な検索を実現するために、**図 1**のように各索引語が出現する文書の番号、出現回数および出現位置を記録した転置インデックスを構築する方法が広く用いられている[1][2][3]。大きな文書集合を扱う場合、転置インデックスのサイズを抑制することが重要である。そのため、転置インデックスは通常圧縮して格納される。

転置インデックスに格納された各索引語の出現位置を利用すれば、複数の索引語が連続して現れる箇所を検索するフレーズ検索や、複数の索引語が指定された距離内に出現する箇所を検索する近傍検索が可能となる。しかし、位置情報(各索引語の出現位置のリスト)はサイズが大きいデータである上、各文書での索引語の有無だけがわかればよい検索では、位置情報を読み飛ばす処理が必要になってしまい検索速度を低下させる要因となる[3]。そこで位置情報を文書番号や出現回数から分離し、必要ときのみアクセスすることが考えられる。だがこの場合、各索引語の個々の文書での位置情報が格納されている場所を知る方法が必要である。そのための方法のひとつが、**図 2**に示すように文書の番号とともに対応する位置情報へのポインタを格納することである。しかし、ポインタは通常１つあたり４バイトであり、転置インデックス全体のサイズが大幅に増加してしまう。

本研究では、位置情報を他のデータから分離することにより位置情報が不要な検索の速度低下を防ぎつつ、少ないデータ量で位置情報へアクセス可能とする方法を開発した。

A new structure of inverted indexes that contain positions of indexing terms

†Tomohiro Yasuda and Makoto Iwayama

†Central Research Laboratory, Hitachi, Ltd.

2. 提案手法

本研究で開発した技術では、101010100101101 のような、0 と 1 の有限の列であるビットベクトルを使用する。任意のビットベクトル B と整数 i に対し、B 中で i 番目の 1 の場所を select(B,i)と定義する。ディクショナリと呼ばれる、一定間隔で select(B,i)の値を抽出した表と短い任意のビットベクトル b に対する select(b,i)の値を格納した表から成るデータを用いると、限られた回数の表引きだけで定数時間で select(B,i)を計算できる[4]。B に 0 が長く連続する箇所があるとサイズの大きな表が必要になる場合があるため、そうした箇所の周辺での select(B,i)の値は事前に計算し格納しておき表引きを避ける。

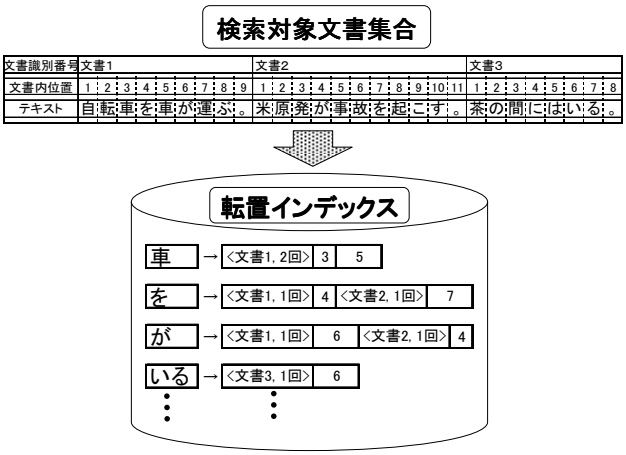

図 1 位置情報を含む転置インデックス

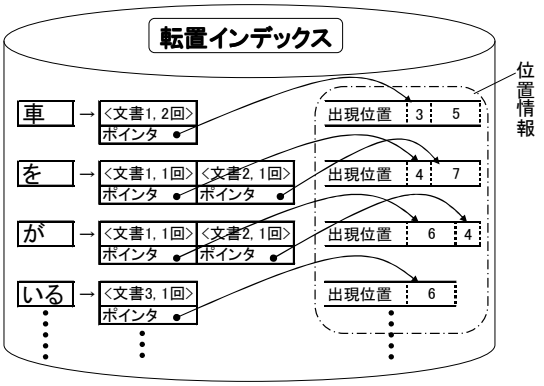

図 2 位置情報へのポインタを持つ転置インデックス

まず各索引語 w について、w の位置情報の大きさに等しい長さのビットベクトル BB(w)を用意する。BB(w)は、個々の文書での位置情報の先頭に対応する箇所で 1、それ以外の箇所では 0 となるよう構築する(**図 3**)。次に、BB(w)のディクショナリを構築する。これにより、w が出現する i 番目の文書での位置情報の先頭の場所は、select(BB(w),i)を計算すれば定数時間で得られる。位置情報の圧縮に variable byte 法[3]のような、位置情報の境界がバイト境界に整列される方法を用いれば、BB(w)の長さを位置情報のビット数ではなくバイト数とでき、短くできる。

各索引語が出現する文書数の総和を N、位置情報の大きさの総和を L とする。ポインタを用いる場合、ポインタのサイズの総和は $4N$ バイトである。一方、ビットベクトルの長さ L に対しディクショナリの大きさが rL ビットとすれば、ビットベクトルとディクショナリの大きさの和は$(1+r)L$ ビットである。従って、索引語が現れる各文書について、位置情報の長さの平均 L/N が $32/(1+r)$バイト未満であれば、ビットベクトルを用いたほうが少ないデータ量で済む。このため、個々の索引語が同一文書中に現れる回数が少なければ、データ量削減の効果が高くなると期待される。なお、r の値については、L が大きくなるにつれ r が 0 へ収束するディクショナリ構築法が知られており[4]、r は小さな値だと仮定できる。

3. 提案手法の効果

下記３つの手法の、転置インデックスのサイズおよび読取時間を計測した。

(A) 位置情報が他の情報と混在 (**図 1**)。
(B) 位置情報を分離し、ポインタを使用(**図 2**)。
(C) 位置情報を分離し、ビットベクトルを使用、すなわち、本研究の手法(**図 3**)。

各転置インデックスは、オンメモリで構築しアクセスした。

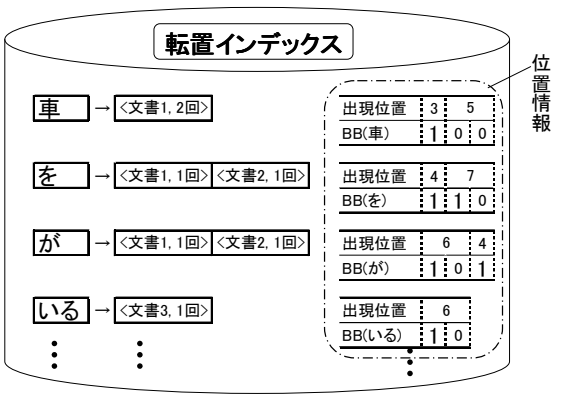

図 3 ビットベクトルを用いる転置インデックス

表1 各手法の比較

	(A)	(B)	(C)
全体のサイズ	38.5MB	52.2MB	44.2MB
追加データのサイズ	－	13.7MB	5.7MB
全体の読取時間	5.53 秒	6.01 秒	10.10 秒
位置以外の読取時間	5.50 秒	1.34 秒	1.32 秒

実験データとして、2005 年の公開特許公報 2000 件のテキストを使用した。数字、記号、アルファベット以外の２文字からなる２グラムを索引語とした。使用した計算機は、 OS が Linux、CPU が Intel 社製 Core2 Quad Q9550 の PC サーバである。

表１に、各手法で構築した転置インデックスのサイズを示す。(B)は追加データが約 13.7MB 必要であったが、(C)では約 5.7MB であり、追加データの量を約 42%に留めることができた。一方、転置インデックス全体の読み取り時間は、(C)は(A)の約 1.9 倍にとどまる一方、位置情報を無視する場合には、(A)に比べて４倍以上高速であった。この結果から、本手法は通常は位置情報を使わない検索を主に実行し低頻度で位置情報を用いる検索が要求される状況において、位置情報をコンパクトに保持するために有用な技術と言える。

4. まとめと今後の課題

ビットベクトルを用いて、転置インデックスに格納した位置情報にアクセスする手法を開発した。これにより、位置情報を用いない検索では検索速度の低下が発生しない一方、ポインタに比べ少ないデータ量で位置情報へのアクセスを可能とした。今後の課題として、ビットベクトルを圧縮[4]し追加データをさらに削減すること、本研究の転置インデックスを用いた検索システムを実装することが挙げられる。

5. 参考文献

[1] 北研二、津田和彦、獅々堀正幹 著、情報検索アルゴリズム、共立出版、東京、 2002.
[2] I.H. Witten, A. Moffat, and T.C. Bell: *Managing Gigabytes (2nd Ed.): Compressing and indexing documents and images*, Morgan Kaufmann, San Francisco, 1998.
[3] F. Scholer, H.E. Williams, J. Yiannis, and J. Zobel: Compression of inverted indexes for fast query evaluation, *Proc. 25th ACM SIGIR*, pp.222-229, 2002.
[4] G. Navarro and V. Makinen, Compressed full-text indexes, *ACM Computing Surveys*, 39(1): Article 2, 2007.