CD-ROM/DVD用日本語全文検索方式

4J－1

高野 浩義　鈴鹿 豊明
日立ソフトウェアエンジニアリング(株)

1 はじめに

近年、電子出版が盛んになってきており、百科辞典や辞書などがCD-ROMで出版されている。これらの検索機能は、予め登録されたキーワードによるキーワード検索が一般的である。しかし、キーワードだけの検索には次の問題点がある。

- データ作成時のキーワード付けにかかる工数が膨大である
- キーワードの洩れや間違いが生ずる可能性がある
- ユーザとしては、キーワードが付いていない本文中の言葉で検索したい場合がある

従って、CD-ROM出版物にも全文検索技術が必要であると考える。

しかし、従来の全文検索方式をCD-ROMに適用しても、性能を得ることは難しい。全文検索は複雑な処理であり、多数回の二次記憶へのアクセスが必要であるため、アクセスが高速なハードディスクには向いても、低速でランダムアクセスを苦手とするCD-ROMには適さないからである。

そこで我々はCD-ROMを対象とする、高速な日本語の全文検索方式を考案し、現在、試作システムを実装中である。なお、今後登場するであろうDVDにもこの技術は適用可能である。本稿ではその方式の概要を報告する。

2 索引の作成方法

既存の全文検索システムでは、元の文書の単純な検索では高速化に限界があるため、何らかの索引を用いている。この索引を作成する際に、元の文書から文法解析によって単語を切り出し、単語単位の索引を作るという手法があるが、日本語は膠着言語であるため完全な文法解析は難しい[1]。ここで間違った索引が作成されると、その結果、検索洩れが生ずることになる。

我々の方式では、索引として「文字成分表」と「凝縮本文」を用いるが、文法解析は行わない。元の文書を、ひらがな・かたかな・漢字・アルファベットの文字種が切り替わる位置で区切り(こうして区切られた文字列を以下「トークン」と呼ぶ)、索引を構成する。図1に文書をトークンに分解する例を示す。

このように、完全に字句解析のみで索引を作成する

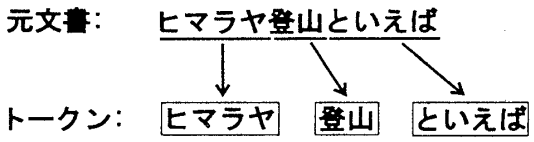

図 1: 文書のトークン分解

ので、前記の問題点を解決できる。

さて、文字成分表は各文章の構成文字をビットマップで表したものである。本方式では候補の絞り込みの効率を上げるため、ひらがな・かたかな・漢字の文字種毎にそれぞれのビットマップを用意する。

しかし、文書と文字の対応が１文字毎だけでは、充分に文書を絞り込むことができない。そこで複数文字のつながりに対応するビットマップを導入することにした。ただし、ひらがな１文字とかたかな１文字以外のビットマップはその定義域が巨大になってしまうため、ハッシングを用いてこの問題を回避する。例を図2に示す。トークンから1, 2, 3文字の部分文字列を取り出し、ハッシュ関数を通してビットマップに登録する(１文字のものはそのまま登録)。

検索処理においては、ユーザが入力した検索文字列を文書の登録時と同様にトークンに分解し、まず文字成分表で候補の文書を抽出するが、ハッシングを用いているので抽出された文書にその文字列が含まれているとは限らない。

A method of full-text search for Japanese text on CD-ROM and DVD
Hiroyoshi Takano, Toyoaki Suzuka
Hitachi Software Engineering Co.,Ltd.

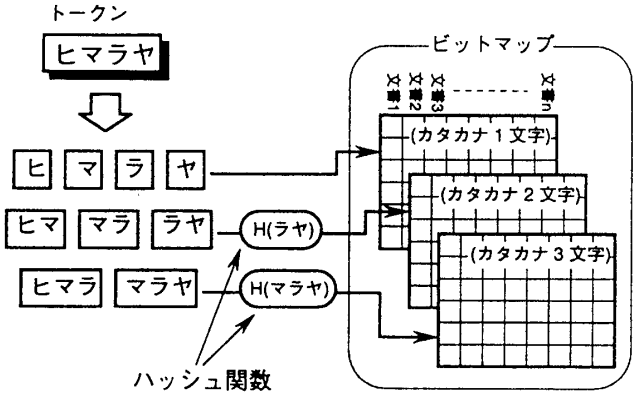

図 2: ハッシングを利用する文字成分表

このため、次に凝縮本文を検索し、さらに候補を絞り込む。凝縮本文は、素のトークンを、B 木を用いた木構造に格納したものである。

検索文字列が 1 トークンのときはここで検索を終るが、2 トークン以上のときはさらに本文を直接走査することになる。

3 索引の最適化

従来の全文検索方式は HD 上に実装されたものであり、その索引は文書の追加や変更に対応する設計になっている。これに対して CD-ROM は読み出し専用であり、更新操作を考慮する必要はない。したがって、索引構造を最初から固定することができる。なるべく絞り込みの効率が良くなるように最適化した索引を作成し、CD-ROM に格納しておけば、CD-ROM へのアクセスが減少し、検索速度は向上することになる。

そこで我々は、上述のハッシュ関数と凝縮本文の最適化を行う。

3.1 ハッシュ関数の最適化

ハッシングはその原理上、コンフリクトが発生する。本方式でのコンフリクトは、複数の文字(列)が同じハッシュ値を取ることを意味する。すなわち、コンフリクトが多ければ、文字成分表のビットマップで抽出された候補文書群の中に、検索文字列が存在しないものが多く含まれることになり、この無駄なものを含んだ文書群から“正しい”文書を選び出さなければならない。

どのようなハッシュ関数であろうとコンフリクトの期待値は一定であるが、各ハッシュ値でのコンフリクトの量の分散が大きいと、検索操作 1 回に要する平均時間も大きくなってしまう [3]。

ここで、ハッシュ値のコンフリクトの分散が小さい、効率の良いハッシュ関数を求め、これによって作成した、最適化されたビットマップを用いれば、検索時間は一定になり、その分高速化することが可能になる。

3.2 凝縮本文の最適化

凝縮本文は前述のように B 木を用いて実装している。この木を検索する処理は、検索文字列のトークンが前方一致で見つからなければ後方一致で、それでも見つからないときは中間一致でのマッチングという流れになる。

前方一致と後方一致の検索は木をたどれば良いので、それほど時間はかからないが、中間一致検索は全キーワードを調べなければならないことを意味する。このとき、HD 上のものをそのまま CD-ROM に移行した木構造を走査すると、CD-ROM に対するシーク動作が大量に発生するため、検索速度の致命的な低下を招く。

そこで、トークンの中間一致検索に備え、B 木のリーフノードが CD-ROM 上で連続した領域に配置されるようなデータ構造を取ることにする。これで、中間一致検索での CD-ROM のシーク動作を最小限に抑えることができ、検索速度が向上する [2]。

4 おわりに

現在、本方式による試作システムを実装中であるが、ハッシュ関数の最適化が未解決である。我々はこれに GA[4] を適用することを考えている。

また、試作システムの完成後、本方式の評価実験を行っていく予定である。

参考文献

[1] 高木, 伊東: “自然言語の処理”, 丸善 (1987)

[2] 古舘, 鈴鹿: “統合化ファイルアクセス法の CD-ROM への適用”, 情報処理学会第 50 回全国大会 2G-8 (1995)

[3] : Aho, Hopcroft, Ullman 著、大野 訳, “データ構造とアルゴリズム”, 培風館 (1987)

[4] 北野: “遺伝的アルゴリズム”, 産業図書 (1993)