■7 群（コンピュータソフトウェア）－ 6 編（情報検索とデータマイニング）

2 章 テキスト情報検索の実際

（執筆者：竹野　浩）［2009 年 4 月 受領］

■概要■

今日最も広く利用されているテキスト情報検索は，Web サーチエンジンと呼ばれるインターネットの Web ページの検索サービスであろう．100 億を越える Web ページに対して多数のユーザが検索を実行し，1 秒に満たない時間で結果を得ることのできるサービスが広く提供されていることは驚異的ですらある．

このような高速大容量のテキスト情報検索を支えているのは，全文検索を高速に行うためのインデックス技術と，インターネットに存在する大量の Web ページを収集するクローラ技術や，収集した Web ページの検索を多数の計算機で分散して処理する分散検索技術などの Web サーチエンジン構成技術である．

【本章の構成】

本章ではインデックス構造（2-1 節）で，検索の高速化に関する基本技術について説明し，Web サーチエンジンの構成（2-2 節）で，大容量のテキスト情報検索を実現するための基本技術について説明する．

■7 群－6 編－2 章

2-1 インデックス構造

（執筆者：竹野　浩）［2009 年 4 月 受領］

2-1-1 転置ファイル

転置ファイル[1]は転置インデックスとも呼ばれ，単語を主キーとしてその単語が出現する文書の識別子（以下，文書 ID と呼ぶ）を格納したファイル（**図 1･1**）を作成することで，検索条件として与えられた単語を含む文書の一覧を高速に検索可能なインデックスファイルのことである．

図 1･1 を例にすれば，「東京」という単語が，文書 ID＝1 と文書 ID＝2 に出現し，「駅」という単語が，同じく文書 ID＝1 と文書 ID＝2 に出現することを知ることが可能であり，「東京」と「駅」が出現する文書を高速に検索することが可能である．

図 1・1　転置インデックス

文書 ID に加えて，その単語の文書内の出現位置も転置インデックスに格納しておくことにより，より精度の高い検索を行うことが可能になる．

図 1･2 を例にすれば，「東京」という単語が，文書 ID＝1 と文書 ID＝2 の文書中の 1 番目に，「駅」という単語が，それぞれ 2 番目と 4 番目に出現していることを知ることができる．これを利用し，検索条件として「東京駅」が与えられた場合，文書 ID＝1 のみを検索結果として出力するといった，文書集合の中から与えられた単語を持つ文書を精度よく抽出することが可能となる．

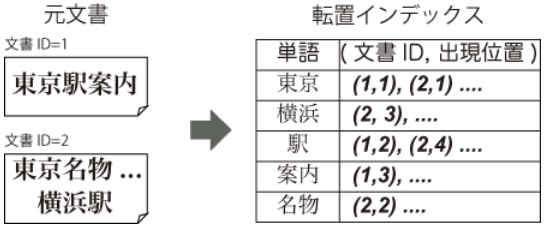

図 1・2　複数単語の検索

転置インデックスを用いた検索では検索の高速化が可能になるが，大規模な文書集合の転置インデックスはサイズが大きくなりがちであること，転置インデックスへの文書の追加，文書

の変更，文書の削除と言った更新が困難であるなどの短所も持つ．また，文書の数が増えると出現する単語の数も非常に多くなるので，転置インデックスの中から単語を発見するのが困難になり，単語を高速に検索するための工夫が必要になる．

一般に Web サーチエンジンの検索では大量の文書がヒットする．そのため，TF/IDF 法 [2] に代表されるスコアリング手法などを用いて，よりユーザのニーズにあった文書を検索結果の上位に出力することが重要である．転置インデックスでは図1･2で示した単語の出現位置に加え，その単語の文書内での出現回数など，スコアリングを行うための情報が追加される場合も多い．

2-1-2 トライとパトリシアトライ

トライは文字列の集合を表す木構造のデータ構造であり，根から葉までを辿る各経路が一つの単語に対応する．ある単語がトライに存在するか否かは，節の部分文字列を比較することで容易に判定が可能であり，存在しない単語をトライに追加することも容易である．この性質を用いて，任意の単語を一意な識別子（以下，単語 ID と呼ぶ）に変換することが可能である．

通常のトライとパトリシアトライの概念を**図 1･3** に示す．

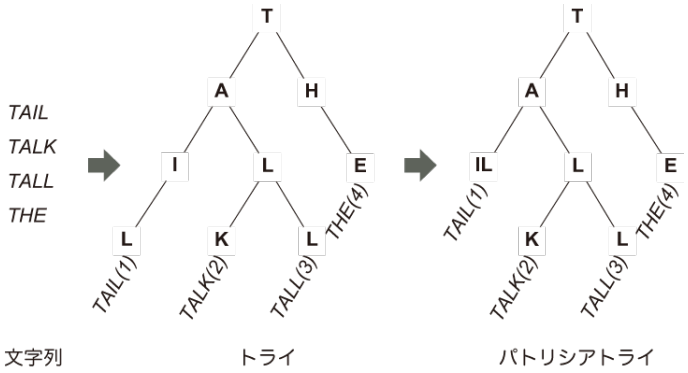

図 1･3　トライとパトリシアトライ

パトリシアトライ（patricia trie）[3] では，トライにおいて分岐していないノードを一つにまとめ，部分文字列を別途格納することを許容する．例えば，“TAIL”，“TALK”，“TALL”，“THE”という単語の効率的な格納を考えた場合，“TAIL”をトライで表すと T-A-I-L となるが，I のノードは L しかないため，この部分は“IL”とまとめることができる．

ノードをまとめることにより，通常のトライより少ないメモリで文字列の集合を表すことが可能である．また，ツリーの深さが浅くなるので検索速度も向上する．そのため，図 1･1，図 1･2 のような転置インデックスにおいて，単語の検索にパトリシアトライが使用される場合が多い．

2-1-3 インデックスレコード構成

図 1･4 に，パトリシアトライを用いた転置インデックスの例を示す．

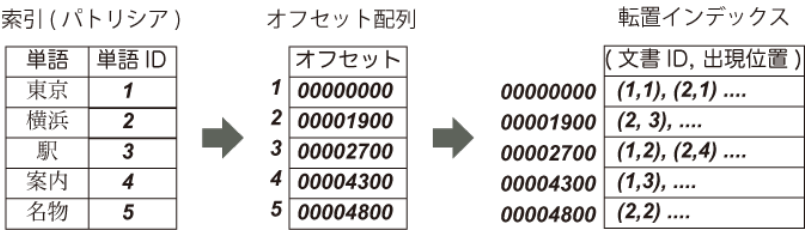

図 1・4　パトリシアトライを用いた転置インデックス

検索条件として与えられた単語は，パトリシアトライを用いて実現された索引を用いて単語 ID に変換される．例えば，「駅」が与えられれば，索引より単語 ID＝3 が得られる．この単語 ID を用いて，オフセット配列からオフセット 00002700 が得られる．このオフセットを用いて転置インデックスの「駅」に関連する文書 ID と出現位置のセットが得られる．

索引と転置インデックスの間にオフセット配列を置く理由は，一度付与された単語 ID が変更されることは少ないのに対して，転置インデックスの中身は文書が追加・更新・削除されるごとに変化し，オフセットが変わる可能性が高いためである．文書が追加され，転置インデックスのオフセットが変更された場合，オフセット配列のみを変更すればよいのでプログラムの効率が良くなる．

実際の検索インデックスでは，文書 ID，検索スコア，単語の出現回数（TF），単語の出現位置など（POS）のレコードが保持される場合が多い．また，大規模な転置インデックスでは，文書 ID，単語の出現位置について，それぞれ文書間差分，文書内の単語出現位置の差分情報を利用してインデックスの圧縮を図る場合がある．

■参考文献

1) Justin Zobel, Alistair Moffat：“Inverted Files for text search Engines,” ACM Computing survery, vol.38, no.2, 2006.
2) William Frakes, Ricardo Baeza-Yates：Information Retrieval: Data Structures & Algorithms, Prentice Hall, ISBN: 0-13-463837-9.
3) MORRISON, DONALD R.:“PATRICIA: Practical Algorithm To Retrieve Information Coded in Alphanumeric,” Journal of the ACM, vol.15, no.4, pp.514-534, 1968.

■7 群－6 編－2 章

2-2 Web サーチエンジンの構成

（執筆者：竹野　浩）［2009 年 4 月 受領］

2-2-1 Web サーチエンジンの全体構成

一般的な Web サーチエンジンは，検索エンジンに加えて，Web サーバから Web ページを収集するクローラ，収集した Web ページを解析するテキスト処理機能，テキスト処理機能の解析結果から生成される検索インデックスから構成される（**図 2･1**）．本節ではこれらの要素に対する基本的な動作原理と実現方法に関して説明する．

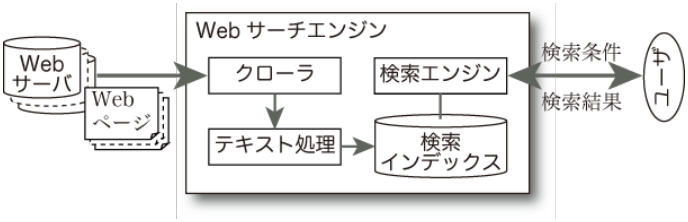

図 2･1　Web サーチエンジンの全体構成

(1)　クローラ

クローラは Web ページを収集するためのシステムである．インターネットの Web サーチエンジンに使用するクローラは，インターネット上の大量の Web ページを収集する能力を持つことが第一の要求条件であることは自明である．

インターネットの Web ページは，常に更新・追加・削除が行われているので，クローラの収集に時間を要すると，検索インデックスの内容とインターネットの Web ページの内容に乖離が生じ，検索サービスの質が低下してしまう．そのため，収集の速度も重要となる．これより，クローラには第二の要求条件として，Web ページを高速に収集する能力が要求される．

インターネット上の Web サーバは，ユーザが Web ページを閲覧するために存在すると考えるのが自然である．クローラによる Web ページの収集は，Web サーバの本来目的と異なるため，できるだけ負荷を与えないことが望ましい．そのため，クローラには第三の要求条件として，Web サーバに不要な負荷をかけずに Web ページを収集する能力が要求される．

クローラが Web サーバからページを収集する速度は，クローラに十分な能力があったとしても，Web サーバの能力と，クローラと Web サーバの間の通信速度に大きく依存する．言い換えれば Web ページ収集の速度はクローラでは制御できない．このため，Web ページを高速に収集するには，**図 2･2** で示す方法を用いる．つまり，Web ページを収集する速度を上げるのではなく，多数の Web サイトからページを同時に収集することにより，トータルの収集速度を向上させる戦略である [1),2)]．これは，個々の Web サーバに過大な負荷をかけることなく，全体の収集速度を向上させる方法であるので，第三の要求条件にもマッチする．なお，大量の Web ページを収集するには，収集した Web ページを格納する大容量のストレージ機能に加えて，収集した Web ページの管理（例えば，Web ページの重複収集を避けるために，収集した URL をす

べて記録するなど）のための大容量のデータベース機能が要求される．

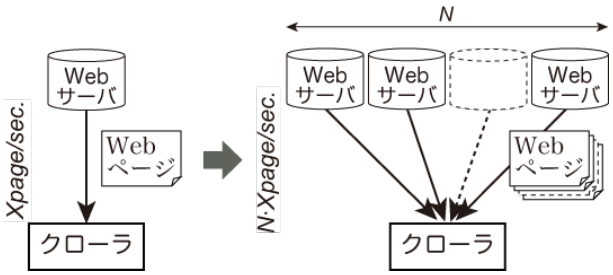

図 2・2　クローラの高速化戦略

(2)　テキスト処理機能

テキスト処理機能は，クローラが収集した Web ページのデータに対して検索精度を向上させるための処理を行い，検索インデックスに反映するための機能である．テキスト処理機能には，ファイル種報や HTML のタグ抽出などの基本的な情報の抽出に加えて，PageRank [3] などの Web ページの関係性の抽出，評価表現 [4] や固有表現 [5] など，自然言語解析による情報の抽出がある．

インターネット上の Web ページを対象としたテキスト処理機能には，大量のデータに対して複雑な解析処理を高速に行う機能が必要になる．また，クローラが収集した Web ページを直接アクセスするのでは複雑な処理を行うことが困難であるため，大量のデータを蓄積して効率良くアクセスするための機能が必要になる．そのため Google による，GFS [6]，MapReduce [7]，Bigtable[8] や，OSS による Hadoop/hBase など [9] の大量のテキストデータを容易に，かつ，並列に処理するためのプラットフォームが発表されている．

(3)　検索インデックス

検索インデックスには‘レコード’として個々の文書情報を登録する．また，文書の本文，作成者，作成日時のような項目をそれぞれのレコード内に‘カラム’として定義し，更に 2-1 節で述べたような転置インデックスを用いることで検索の高速化を図っている．

(4)　検索エンジン

検索エンジンでは，ユーザのニーズにあった検索を行うため，様々な検索が実行可能である必要がある．そのため，個々のカラムに対する検索条件や，複数のカラム検索条件に対する論理演算が必要となる．

2-2-2 分散サーチエンジンの構成

検索インデックスのサイズが大きくなると，単一の計算機では，これを高速に検索することが困難になる．そのため，検索インデックスを分割して複数の計算機に割り当て，協同で検索を行う「分散サーチエンジン」が必要になる．

検索インデックスの分散方式には，単語単位に分散する「単語分散方式」（**図 2・3** の(1)，(2)，(3)）と，文書単位に分散する「文書分散方式」（図 2・3 の(A)，(B)，(C)）が考えられる [10]．

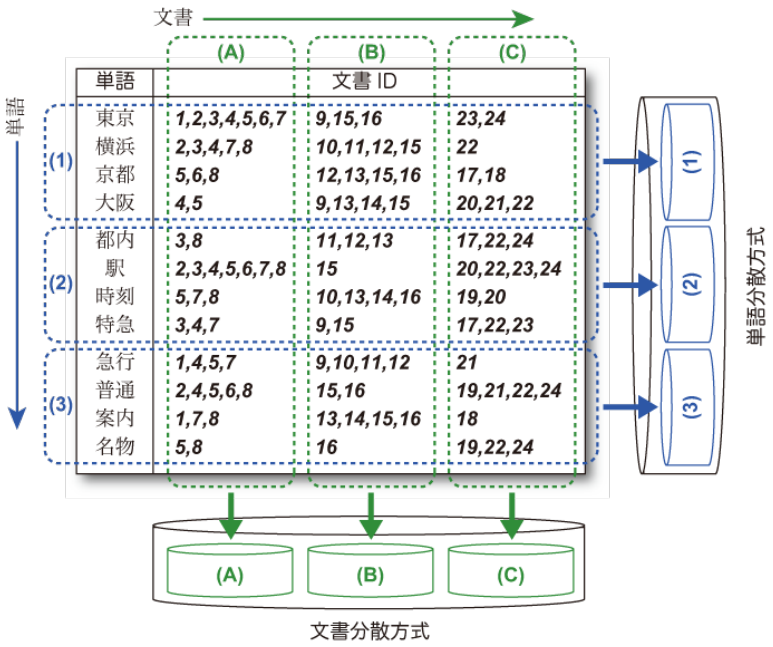

図 2・3　インデックスの分散方式

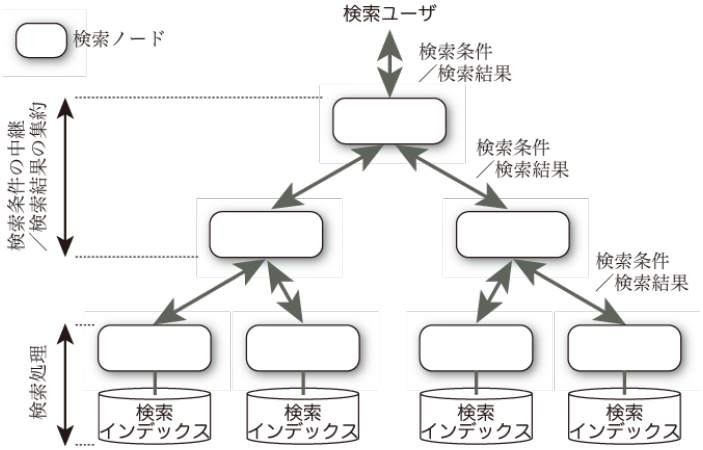

図 2・4　分散サーチエンジンでの検索

単語分散方式では，検索条件として与えられた単語を担当する計算機のみが検索処理を行うのに対し，文書分散方式では，すべての計算機が同じ検索条件を受けて検索処理を行う．また，

単語分散方式では，検索条件に複数の単語が含まれる場合，それぞれの単語を担当する計算機が検索結果をすべて出力して比較しなければ最終的な検索結果が得られないのに対し，文書分散方式では，検索結果の上位のみを知りたい場合，各計算機は，それぞれの上位の検索結果のみを出力して比較すれば，最終的な検索結果を得ることができる．計算機間の通信能力は有限であるため，単語分散方式より，文書分散方式の方が，文書数に対してスケーラビリティが高い方式であるといえる．

文書数が増加して計算機の数が更に増加すると，文書分散方式でも計算機間の通信能力が問題になってくる．そのため，大容量の文書分散方式では，**図 2･4** に示すように，計算機を検索ノードとしてツリー状に配置することにより，通信量の増大を防ぐ構成をとることが一般的である．この場合，検索ユーザが入力した検索条件は，中間の検索ノードを経由して最下位の検索ノードへ伝達される．最下位の検索ノードは，自分が担当する検索インデックスに対して行った検索結果を出力し，中間のノードは，下位のノードから受け取った検索結果を集約し，上位の結果のみを出力する．この処理を繰り返すことにより，すべての検索ノードの出力を一箇所に集めるような大規模な通信を行うことなしに，すべての検索結果の中から上位の結果のみが，検索ユーザに出力される．

■参考文献

1) 速水，竹野，永瀬，藤本，萩原：“スケーラビリティのある WWW 全文検索システム構築法の提案と評価,”情報処理学会研究報告，データベース・システム研究会報告, vol.2001, no.8, pp.45-52, 2001.
2) Hiroshi Takeno, Makoto Muto, Noriyuki Fujimoto, and Kenichi Hagihara：“Developing a Web Crawler for Massive Mobile Search Services,”In Proc. of 7th International Conference on Mobile Data Management (MDM 2006), Nara, Japan, 2006-05.
3) Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd：“The PageRank Citation Ranking: Bringing Order to the Web,”Technical Report, Stanford InfoLab, 1999-66.
4) Nobuhiro Kaji and Masaru Kitsuregawa：“Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents,”In Proc. of the 2007 Joint Conference on Empirical Methods in Natural Language Processing (EMNLP-CoNLL2007), pp.1075-1083, 2007.
5) 高橋，浅野，松尾，菊井：“単語正規化による固有表現の同義性判定,”言語処理学会第 14 回年次大会, pp.821-824, 2008.
6) Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung：“The Google file system,”In Proc. of the 19th ACM SOSP (Dec. 2003), pp.29-43.
7) J. Dean, and S. Ghemawat：“MapReduce: Simplified data processing on large clusters,”In Proc. of the 6th OSDI (Dec. 2004), pp.137-150.
8) F. Chang, J. Dean, et al. ：“Bigtable: A Distributed Storage System for Structured Data,”In Proc. of the 7th OSDI (Dec. 2006), pp.205-218.
9) A. Bialecki, M. Cafarella, D. Cutting, O. O'Malley：“Hadoop: a framework for running applications on large clusters built of commodity hardware,”http://hadoop.apache.org/, 2005.
10) 澤田，竹野，藤本，萩原：“並列型全文検索システム構築のための手法の提案とその評価,”情報処理学会研究報告, 2000-AL-71, pp.1-8, 2000-01.