全文検索システムにおけるフレーズインデックス保持戦略

井上　孝史† 　植松　幸生† 　安田　宜仁† 　片岡　良治†

† 日本電信電話株式会社/NTT サイバーソリューション研究所
E-mail: †{inoue.takafumi,uematsu.yukio,yasuda.n,kataoka.ryoji}@lab.ntt.co.jp

あらまし　多くの全文検索エンジンは，目的の文書を高速に同定するために転置インデックスを利用する．しかし，フレーズがクエリとして入力された場合の処理 (フレーズ検索処理) は各タームの連接の確認処理が必要なため処理コストが大きいことが知られている．これを解決する方法としてフレーズそのものをインデックスに保持するフレーズインデックスがある．また，フレーズインデックスは明示的なフレーズマーカの無い複数語クエリに対しても有効である可能性がある．しかし，どのような戦略に基づいて保持すべきフレーズを選択すべきかは明らかになっていない．先行研究では，クエリログ中での頻度に基づいてフレーズを選択する方法が提案されている．我々は，実際の処理コストに基づいてフレーズ選択を行う方法を提案する．本研究では，実際のクエリログを用いて上記二つのフレーズ選択戦略を比較し，処理コストに基づく方法が，従来手法に比較して高い効率を得られることを示した．
キーワード　全文検索, フレーズ検索, クエリログ

Takafumi INOUE†, Uematsu YUKIO†, Norihito YASUDA†, and Ryoji KATAOKA†

† NTT Cyber Solutions Laboratories, NTT Corporation
E-mail: †{inoue.takafumi,uematsu.yukio,yasuda.n,kataoka.ryoji}@lab.ntt.co.jp

Abstract　Many search engines exploit inverted indexes to locate documents fast. Processing phrase queries is expensive even with indexes. The *phrase index* is known to be an effective solution to it. The phrase index may also be useful for multi-term queries without explicit markers. However what is the best phrase selection strategy for phrase indexes is still an open problem. Using the Excite query log, we explore two phrase selection strategies for the phrase index: query frequency-based strategy and processing cost-based strategy. The experimental results show that the phrase selection strategy based on the processing cost achieves higher efficiency.
Key words　Full-text search, Phrase search, Query log

1. まえがき

多くの全文検索エンジンは，目的の文書を高速に同定するために転置インデックスを利用する．一般的な転置インデックスでは,「ターム」がインデックスのエントリであり，そのタームが出現する位置情報 (文書 ID，出現頻度，文書内での位置) のリストである転置リストをタームと関連付けて保持する．検索エンジンに与えられるクエリには，フレーズクエリが相当数含まれており，フレーズがクエリとして入力された場合の処理 (フレーズ検索処理) は各タームの連接の確認処理が必要なため処理コストが大きいことが知られている．これを解決する方法としてフレーズそのものをインデックスに保持するフレーズインデックスがある [6]．ユーザはフレーズマーカ (典型的にはダブルクォートが用いられる) を用いて明示的にフレーズクエリであると指定すること無しに複数タームからなるクエリを入力しながらも，フレーズとして処理されることを期待している場合が少なからずある．このようなクエリを暗黙的なフレーズクエリ (implicit phrase query) と呼ぶ．複数タームクエリの中から，暗黙的なフレーズクエリを何らかの方法で検出することができれば，フレーズインデックスを活用することが可能である [7]．さらに，フレーズ検出機能が無い場合でも，タームの近接性を考慮したスコアリングやランキングを行う検索処理にフレーズインデックスを利用する研究が近年行われており，その有効性が示されている [8]．

このように，フレーズインデックスは有効なものであるが，限られたリソースの下ですべてのフレーズをインデックスに保持することは不可能である．そのため，何らかの基準にしたがってインデックスに追加するフレーズを選択する必要が生じる．単純な方法として，過去のクエリログの中で頻繁に出現するものからインデックスに追加するという戦略がある．クエリログとは過去に全文検索エンジンに投入されたクエリの履歴である．このクエリログの頻度を利用する方法はある程度有効であることが報告されている [6] が，クエリログ中の頻度だけがフレーズ選択の基準ではない．なぜなら，フレーズクエリの処

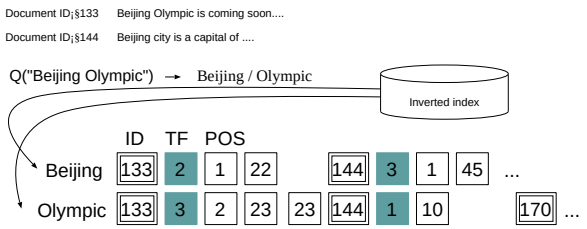

図 1　典型的な転置インデックス

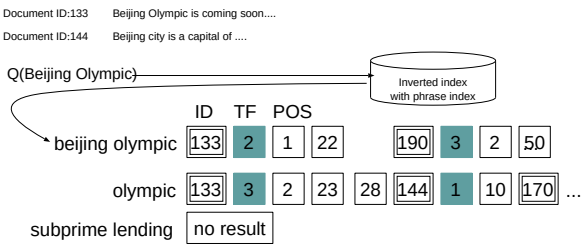

図 2　フレーズインデックスを持つ転置インデックス

理コストはフレーズ毎に異なるからでる．よって，クエリログ中で，単にあるフレーズが他のフレーズより多く現れるという理由だけでそのフレーズをフレーズインデックスに加えるのが妥当だとは言えない．フレーズをインデックスに加えるそもそもの動機は，より高い検索効率 (速度) を達成するためであった．そこで我々は検索サービスのある一定の運用期間において，より多くの処理コストを要したフレーズをインデックスに加える方法を提案する．全文検索エンジンがトレーニングセットのクエリログ中の各フレーズを実際に処理するのに要したトータル処理コストを測定し，これをそのフレーズの「累積処理コスト」と呼ぶ．我々はこの「累積処理コスト」の高いフレーズから優先的にフレーズインデックスに登録する戦略を取る．

公開されている Web 検索サービスのログと TREC-8 の文書集合を用いて上記 2 つのフレーズ選択戦略，すなわち「クエリログ中の頻度に基づく方法」と「累積処理コストに基づく方法」のそれぞれの効果を比較調査した．

2. フレーズ検索処理

本章では転置インデックスを用いた全文検索エンジンにおけるフレーズ処理方法を概観し，フレーズ処理の高速化のためのフレーズインデックスについて述べる．我々が想定する典型的な転置インデックスの構造と，それを用いた場合のフレーズの処理手順を図 1 に示す．図中 `ID` は文書 ID，`TF` (term frequency) は当該タームのその文書中での出現頻度，`POS` は当該タームのその文書中での個々の出現位置のリストである．これらを合わせて当該タームに対する転置リストと呼ぶ．たとえば，`beijing` というタームに対する転置リスト中の先頭の文書 ID は 133，TF は 2，beijing は文書 ID133 の中の先頭と，22 番目の位置に出現している．

“`beijing olympic`” というフレーズクエリに対して検索を行う場合，`beijing` というタームのすぐ次に `olympic` というタームが出現するような文書を特定することになる．この処理手順は以下の通りである．

まず，`beijing`，`olympic` それぞれの転置リストを取得する．次に取得した転置リストの位置情報を参照して連接処理を行い，`beijing` の次に `olympic` が出現している文書を特定する．図 1 の例では，文書 ID133 には `beijing` と `olympic` の両方のタームが出現し，その出現位置が 1,2 と隣り合っているので検索結果に含める．一方，文書 ID144 は両方のタームが出現しているものの，その出現位置が隣り合っていないため，フレーズクエリの検索結果として適当で無いと判定する．この連接処理はフレーズを構成する全てのタームについて行う必要がありコストが高いという問題点がある．

フレーズ検索処理のコストを軽減する方法がいくつか提案されている．一つはフレーズ検出のための隣接性チェックにおいて線形探索の代わりに二分探索を行う方法がある [1]．処理コストを下げる他の方法として，*Skip list* [4] が提案されている．それらの方法はある程度有効であるものの，複数の転置リストを読み込む必要があるため，各転置リストが長い場合には，フレーズ検索処理のためのコストは，やはり無視できるものではない．

フレーズ検索処理の問題をより積極的に解決する方法として，フレーズインデックスが提案され，有効であると報告されている [6] [9]．フレーズインデックスでは，フレーズそのものをタームと同様に扱う．すなわち，語彙辞書にフレーズを登録し，対応する転置リストをインデックス中に保持する．

フレーズインデックスは通常の転置インデックスに比べてより多くの記憶容量を必要とする．しかしながら，フレーズインデックスを用いれば，フレーズを構成する各タームの転置リストを参照し，連接処理をする必要はなくなる．フレーズに対する転置リストを読み込むことでただちにそのフレーズの出現位置を得ることができるため，処理コストを大幅に削減することが可能である．

図 2 にフレーズインデックスを用いた際のフレーズ検索処理の概要を示す．“`beijing olympic`” がクエリとして入力された場合，まず，そのフレーズがフレーズインデックスに存在するかどうかを調べる．存在する場合には，フレーズに対応する転置リストを読み込むことで検索結果を得られる．存在しない場合には，先に本節で説明したのと同様に処理する．なお，図中の `subprime lending` のようにフレーズインデックスには検索対象の文書集合中に出現しないフレーズを登録する場合もある．

また，フレーズインデックスは暗黙的なフレーズの処理や，複数タームクエリにおいてターム間の近接性を考慮する検索処理にも活用できる可能性がある [8]．

3. フレーズインデックスのためのフレーズ選択方法

フレーズインデックスの有効性が示されているものの，考えられる全てのフレーズを登録することはリソース (速度とメモリ) の観点から現実的でない．よって，実用的には保持すべきフレーズをいくつか選択しフレーズインデックスを作成するが，どのようなフレーズを選択すべきかという問題が残る．

最も単純な方法としてクエリログに頻出するフレーズを選択する方法が考えられる．例えばクエリログに出現するフレーズの出現頻度の上位から選択する方法は Williams 等によって有効性が示されている [6]．しかしながら，先行研究ではあるフレーズの連接処理に必要なコスト (処理コスト) は考慮されていない．例えば前述した位置情報を保持する転置リストの長さによって処理コストは大きく異なる．そこで本研究では出現頻度のみではなく，検索に必要な処理コストを考慮してフレーズを選択する方法を提案する．

処理コストはあるフレーズを処理するのに必要な時間で表現することが出来る．この処理に必要な時間が小さい場合は，クエリログに相当頻度出現しているフレーズであっても，それをインデックスに追加することによって必ずしも大きな効果が得られるとは限らない．逆にあるフレーズの処理コストが大きい場合は，クエリログでの頻度が他と比べて小さい場合でも，インデックスへの追加の効果が高くなる可能性がある．我々はこうした特徴を捉えるために処理時間とクエリログでの頻度の両方を考慮した方法を提案する．

以上の考察に基づき，インデックスに登録するフレーズ数を変化させながら，下記 2 つのフレーズ選択のアプローチの効果を比較調査する．

（1） クエリログ中の頻度に基づく方法 (頻度ベース)

（2） 累積処理コストに基づく方法 (処理コストベース)

1. の頻度ベースのアプローチは先行研究と同様にクエリログに頻出したフレーズの上位から n 個のフレーズを選択する．クエリログを $Q = \{q_1, q_2, \cdots, q_m\}$ と表すと，頻度ベースのアプローチは q の Q 内での出現頻度 $freq(q)$ の値の大きなものからインデックスに登録すべきフレーズ q を選択する．

我々が提案する 2. の処理コストベースのアプローチはクエリログの頻度だけでなくフレーズの処理時間も考慮するアプローチである．処理コストは下記の式によって計算する．

$$v(q) = freq(q) \times processing\_time(q) \quad (1)$$

ここで $processing\_time(q)$ はフレーズ q を処理するのに必要な時間を測定し利用する．よって $v(q)$ はクエリログ Q の中で q を処理するのに利用した累積時間を表している．この v の値を降順に並べてその上位 n 個のフレーズを選択する．

4. 評価実験

本実験では追加するフレーズインデックスのサイズとその効果の関係を明らかにする．フレーズインデックスに登録するフレーズ数が大きくなればフレーズインデックスで処理可能なフレーズ数は大きくなる．しかしながら，フレーズインデックスを参照するオーバヘッドが大きくなる可能性があり，その効果は次第に低減するはずである．我々は実際のクエリログと情報検索の標準的なデータセットを利用して，この関係を明らかにする実験を行った．フレーズインデックスのサイズはインデックスファイルのサイズと選択したフレーズ数で表すことが可能なため，その両方を用いて実験結果を示す．

以上示した観点で前述した頻度ベース (先行研究) と処理コストベースの (提案手法) のアプローチを比較し，提案手法の有効性を示す．

4.1 実験準備

我々はクエリログとして広く公開されている Excite ログ 1999 (注1) を使用した．Excite ログは 2,477,283 件のクエリで構成されている．このクエリログを 2 つに等分割し，最初の半分をトレーニングセット，後半をテストセットとして利用した．フレーズインデックスに追加する候補の対象はトレーニングセット中で頻度 5 以上出現する複数タームのクエリ 27,649 個とした．

データセットには情報検索の分野で広く使われている TREC-8 の ad-hoc 検索トラックで使われたデータセットを用いる．TREC-8 は 528,155 のニュース記事より構成されている．これらのクエリログ及びデータセットに対して Porter stemming algorithm [5] を適用し，stemming を行った．データセットのサイズは約 2 ギガバイトで文書中に含まれる全てのタームを使用し，*stop word* などは設けていない．

全文検索エンジンには転置インデックスベースで実装されている *ftsearch* (注2) を用いた．ftsearch では 2 つのファイルを用いて転置インデックスが構成されている．1 つはタームと位置情報を関連付けるための語彙辞書で B-tree で実装されている．他方は転置リストを実際に保持するファイルである．我々のフレーズインデックスは，語彙辞書に通常のターム (シングルターム) と同様にフレーズを登録できるように ftsearch を改造することによって実現している．

これらの実験を 2 つの Xeon X3210 2.13GHz CPU，8GB メモリのマシンで行った．インデックスの大きさよりもメモリの方が大きいため，インデックスはメモリ上で処理される．

我々は，クエリログに含まれる複数の単語で構成される全てのクエリをフレーズとみなし検索処理時間を測定した．一般的にはダブルクォーテーションで囲まれた明示的なフレーズのみをフレーズとして扱うが，クエリログに含まれる明示的なフレーズは 142,720 件で全体の約 5%と少ないためフレーズインデックスの効果を検証するのに十分ではない．よって本実験では全ての複数のタームをフレーズとみなしている．これは相当粗い近似であるが，我々はまったく非現実的な仮定だとは考えていない．一つには，複数のタームで構成されるクエリの頻度上位を見てみると，例えば `britney spears` や `search engine` のようにフレーズとみなすべきクエリが多いという事実がある [3]．またクエリがフレーズと意図されているかどうか判別できない場合でも，フレーズインデックスを利用することによって，ターム間の近接性を考慮した検索処理を効率的に行える可能性がある [8]．その場合にはまず，クエリがフレーズとして存在するかどうかをフレーズインデックスを用いて調べ，存在しなかった場合には通常の AND 検索として処理するという方法が戦略の一つとして考えられる．以上の考察から，本実験ではテストセット中のすべての複数タームクエリをフレーズインデックス追加の候補とすることにした．

(注1)：`http://ist.psu.edu/faculty_pages/jjansen/academic/transaction_logs.html`

(注2)：`http://www.ingrid.org/~harada/interface/index.html`

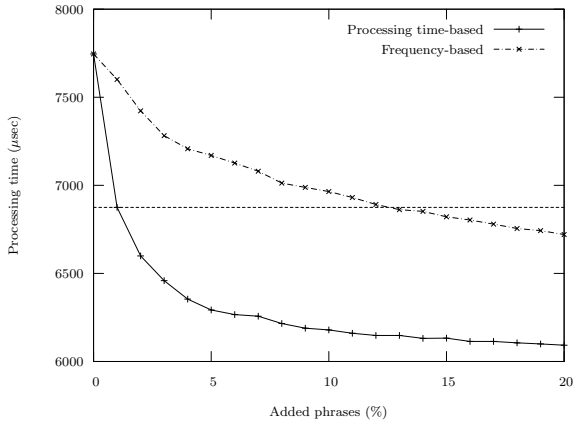

図 3　検索処理時間とフレーズ数の関係 (頻度ベース:点線，処理コストベース:実線)

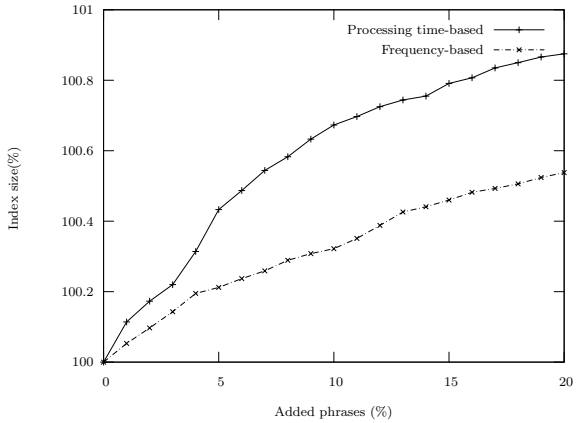

図 4　インデックスサイズとフレーズ数の関係 (頻度ベース:点線，処理コストベース:実線)

4.2 実験結果

図 3 に平均検索処理時間とフレーズインデックスに追加したフレーズ数の関係を表すグラフを示す．縦軸は検索処理時間で単位は μsec(マイクロ秒)，トレーニングセットから抽出された全てのフレーズ 27,649 個を 100%とした際にフレーズインデックスに追加したフレーズの割合である．点線が頻度ベース，実線が提案する処理コストベースのアプローチによる結果を示している．実験結果は登録するフレーズ数が大きくなるほどその処理時間が小さくなっているが，その効果は小さくなっていくことが分かる．また，処理コストベースのアプローチの方が頻度ベースと比較して処理時間を低減可能なことが分かる．図中の水平線をみると，処理コストベースで全体の 1%のフレーズを追加することで，頻度ベースで 12%のフレーズを追加する場合と同等の効果が得られたことが分かる．

図 4 はフレーズインデックスを追加した場合のインデックスサイズの変化を表す．縦軸がフレーズインデックスが無いベースラインのインデックスサイズ 570,294KBytes を 100%とした場合の割合で，横軸が追加したフレーズの割合である．点線が頻度ベースで実線が提案する処理コストベースでのインデックスサイズの変化を表している．フレーズインデックスを追加することで，ベースラインと比較して 0.1%から 1.0%のインデックスサイズの増加が見られる．

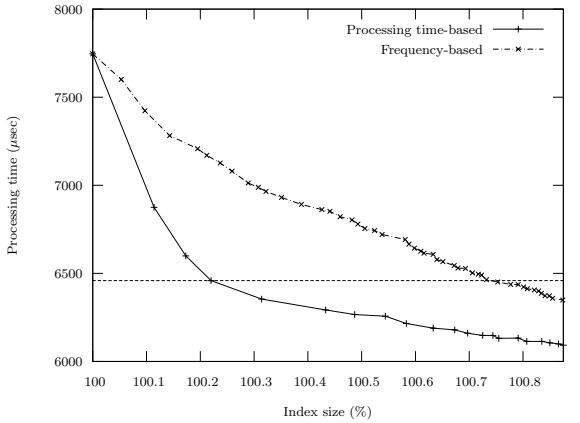

図 5　検索処理時間とインデックスサイズの関係 (頻度ベース:点線，処理コストベース:実線)

このグラフから処理コストベースの方が頻度ベースよりも同フレーズ数で比較した場合はインデックスサイズが大きくなることが分かる．これは長い転置リストの処理コストが高く，処理コストベースのアプローチではそれらの転置リストが選択されたことが原因と推測する．

図 5 にインデックスサイズと処理速度の関係を知るために，図 3 と図 4 を再構成しインデックスサイズと処理時間の変化の関係を示す．縦軸が検索処理時間，横軸がインデックスサイズである．図より処理コストベースの方が頻度ベースのアプローチよりも同じインデックスサイズでも処理コストを削減する効果が見られることが分かった．

図中の水平線を見ると，提案手法である処理コストベースにおける 0.2%のインデックスサイズの増加と頻度ベースの 0.7%がほぼ同じ検索処理時間であることが分かる．よって，提案法がよりインデックスサイズを低減しつつ，検索処理時間の削減効果が高いことが分かる．

これらの実験結果から処理コストベースの方が頻度ベースよりもフレーズを選択する方法として適当であると言える．

5. 結　　論

我々は，頻度ベースと処理コストベースの 2 つのフレーズ選択手法について検索ログと情報検索の標準的なデータを利用した実験を行い，提案する処理コストベースの方が頻度ベースと比較して効果的であることを実験より示した．

利用した検索ログにはその検索ログと対応するインデックスが含まれていないため TREC-8 の ad-hoc 検索トラックのデータを利用したが，クエリとデータセットにドメインのミスマッチがおきてしまった．例えば `britney spears` のように大量の検索結果が返却されることが想定されるクエリでも，TREC-8 にはほとんど含まれていなかった．よって，実際の効果を確かめるために Web のインデックス等を利用した実データに近い実験を行うことが必要である．また，TREC-8 のデータセットはサイズが小さく，メモリに載っていることが前提となってい

たが，大量のデータを扱う場合はメモリに載らないことも想定されるので，メインメモリよりも大きなデータを扱う場合についても同様の傾向かを調べる必要がある．

また，本実験のフレーズインデックスの転置リストには他の転置リストと同様にフレーズの位置情報も格納したが，実験ではクエリとインデックスの完全マッチのみを適用したため利用していない．我々はこの位置情報を利用してクエリの部分マッチも考慮した方法も検討している．

フレーズインデックスはトレーニングセットから静的に決定されたが，クエリログは日々刻々と変化するためどのようにフレーズインデックスに登録するフレーズを更新するかは課題となる．

実用的な大規模検索サービスでは，検索処理を行う前に検索結果のキャッシングを行い，実際に検索処理が行われるクエリとキャッシュで解決可能なクエリが存在する．したがってキャッシュとフレーズインデックスとの関係を明らかにする必要がある [2]．

本実験では，クエリログが英語であったため英語のフレーズインデックスを対象としたが日本語や中国語のように単語境界が与えられていない言語の場合，明示的なフレーズとして複合語が入力されその頻度は英語よりも大きいことが想定される．よって，本手法はそのような言語に対して大きな効果を得られる可能性がある．

文　　献

[1] R. Baeza-yates, R. Salinger, and S. Chile. Experimental analysis of a fast intersection algorithm for sorted sequences. pages 13–24, 2005.

[2] V. P. G. Skobeltsyn, F. Junqueira and R. Baeza-Yates. Resin: A combination of results caching and index pruning for high-performance web search engines. In *Proceedings of SIGIR'08*, 2008.

[3] C. D. Manning, P. Raghavan, and H. Schutze. *Introduction to information retrieval*. Cambridge university press, 2008.

[4] A. Moffat and J. Zobel. Self-indexing inverted files for fast text retrieval. *ACM Transactions on information systems*, 14(4):349–379, 1996.

[5] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[6] H. E. Williams, J. Zobel, and D. Bahle. Fast phrase querying with combined indexes. *ACM Transactions on information systems*, 22(4):573–594, 2004.

[7] W. Zhang, S. Liu, C. Yu, C. Sun, F. Liu, and W. Meng. Recognition and classification of noun phrases in queries for effective retrieval. In *Proceedings of the 16th ACM conference on information and knowledge management*, pages 711–720, 2007.

[8] M. Zhu, S. Shi, N. Yu, and J.-R. Wen. Can phrase indexing help to process non-phrase queries? In *Proceedings of the 17th ACM conference on information and knowledge management*, pages 679–688, 2008.

[9] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM computing surveys*, 38(2):6, 2006.