N-gram索引における複合検索条件の効率的な処理方法

Efficient Query Evaluation Method for Complex Queries in n-gram Indexing

小川　泰嗣*
Yasushi OGAWA

要　　旨

AND，OR，ANDNOT演算子の検索処理を対象に，無駄な位置の突き合わせ処理を行わないことで，大幅に検索時間を短縮する効率的なアルゴリズムを提案した．さらに，AND・OR演算子が入れ子になった検索条件に対しては，ノード数に応じて選択的にOR標準形に変換することで検索処理を高速化する手法を開発した．日本語文書に対する代表的な索引手法としてn-gram(連続するn文字組)を索引単位とするn-gram索引がある．N-gram索引検索では，検索語が文書中に出現しているかを確認するためのn-gramの出現位置の突き合わせが検索時間を増大させるが，本手法はこの増大を可能な限りおさえるものである．新聞記事5年分を用いた評価により，本手法の有効性が確認された．

ABSTRACT

An efficient query evaluation method is proposed, which reduces unnecessary proximity checks for AND, OR and ANDNOT operators. Also a method is developed which selectively converts complex queries with both AND and OR operators to the OR normal form according to the number of child nodes in these operators. N-gram indexing which uses n-grams (n successive character strings) as indexing units is widely used in document retrieval for Japanese texts. The proposed methods prohibit time increase, in retrieval using n-gram index, caused by checking the proximity constraints among the n-grams in order to determine whether a given word exists in a document. The usefulness and the effectiveness of the methods is shown through experimentation using five years newspapers.

* 画像システム事業本部　ソフトウェア技術開発センターソフトウェア研究所
Imaging System Business Group, Software Technology Development Center, Software Research Center

1. 背景と目的

文書電子化とコンピュータネットワークの進展に伴い，蓄積された大量の文書群から所望の文書を見つけ出す文書検索技術の重要性が高まっている．特に，最近ではインターネットのサーチエンジンや電子図書館に見られるように検索対象が大規模になっているので，検索速度の向上が強く求められている．

高速検索の実現には，索引の利用が不可欠である[1]．英語など多くの言語においては，単語を索引付けの単位として索引を作成することが一般的である．しかし，日本語文書を対象とする場合，日本語ではスペースなどによって単語の切れ目が明示的に示されないことが問題となる[2]．形態素解析を利用して単語を切り出し，英語と同様に単語単位の索引を作成する方法も考えられる．しかし，形態素解析を用いる場合，辞書の作成・維持に要するコスト，形態素解析の誤りに起因する検索誤り，処理オーバヘッドによる処理速度の低下などの問題が発生する．そこで，単語の代わりに連続するn文字組(以下，n-gram)を索引単位とするn-gram索引[3]が広く使用されている．

しかし，n-gram索引にも問題がある．検索速度の観点からは，単語索引であれば単一の索引単位として処理される検索語が複数のn-gramに分割処理されることが問題となる．すなわち，nより長い検索語の処理では，分割の結果生じたn-gramが文書中で検索語を構成していることを確認する必要がある．出現文書ごとのn-gramの出現位置(先頭からのオフセット)を索引に記録しておき，検索時にその出現位置を突き合わせることで検索語の存在を確認できるが，この位置検査処理が検索時間を増大させるのである．そこで，検索高速化のために，文字種等に応じてnを調整する[4][5]，n-gramの処理順序や選択方法を工夫する[6][7]といった手法が提案されている．

実際の検索場面では，単一の検索語のみから成る検索条件だけでなく，複数の検索語を様々な演算子で組み合わせた検索条件(以下，複合条件)が使用されることが多い．代表的な演算子には以下のものがある[1]．

・AND演算子
　子ノードの検索結果の集合積を検索結果とする*．
・OR演算子
　子ノードの検索結果の集合和を検索結果とする．
・ANDNOT演算子
　第1子ノードの検索結果と第2子ノードの集合差を検索結果とする．

ところが，前述の高速化手法はいずれも単一の検索語を対象としたものである．単一の検索語の処理が高速化されれば複合条件の処理も高速化される．しかし，複合条件の特性を考慮することができれば，一層の高速化が期待できるはずである．

複合条件では，条件に含まれる検索語に対する検索結果から最終的な検索結果が生成されるので，位置検査を行わなくても良い場合がある．本研究では，そのような無駄な位置検査を行わないことで検索処理を高速化する方法を提案する．さらに，検索場面で使用されることの多いAND演算子のなかにOR演算子が入れ子になった検索条件(以下，混合条件)の効率的な処理についても検討する．

本論文の構成は以下の通りである．つぎの章ではn-gram索引の登録・検索の基本的な処理方法，および単一検索語の高速化手法について述べる．3章では，AND・OR・ANDNOT演算子に応じて位置検査を削減する方法を提案するとともに，混合条件処理の高速化についても検討する．4章では，新聞記事を用いた評価結果および考察を示す．最後の5章はまとめである．

2. n-gram索引

2-1　登録処理

文書登録時には，対象文書から部分的に重複するものも含めて，全てのn-gramを抽出する．ただし，文書の末尾部分にあるn文字未満の単語を検索可能とするため，文書の末尾部分からはn文字未満のn-gramも抽出する．この処理を末尾処理と呼ぶ．

例えば，bi-gram(n=2のn-gramをbi-gramと呼ぶ)索引では，

* AND演算子・OR演算子を2項演算子とする場合と多項演算子とする場合がある．本研究では後者を採用する．

「携帯電話」という文書から「携帯」「帯電」「電話」「話」を抽出する．最後に「話」を1文字で抽出するのが末尾処理である．

2-2　検索処理

文書検索時には，検索語の長さ(以下m文字とする)に応じて処理方法が異なる．

(1)　m＜nの場合
検索語が索引単位のn-gramより長い場合に相当する．検索語から長さnのn-gramを抽出し，それら全てを含む文書を特定する．しかし，n-gramがバラバラに出現していたのでは検索語を含むことにならない．例えば，「携帯式電話機の帯電」という文書は，「携帯電話」から抽出される「携帯」「帯電」「電話」を全て含んでいるが，「携帯電話」自体は含んでいない．そこで，全てのn-gramを含む文書においてn-gramが連続した位置にあることを検査する必要がある．

(2)　m＝nの場合
検索語と索引単位のn-gramが等しい長さの場合に相当する．検索語と等しいn-gramが索引に登録されていれば，そのn-gramが出現していた文書が検索結果となる．

(3)　m＜nの場合
検索語が索引単位より短い場合に相当する．索引に登録されているn-gramで，先頭m文字が検索語に一致するもののいずれかを含む文書が検索結果となる．そこで，そのようなn-gramの全てをOR演算子で結合したものを検索条件として，検索を行なう．例えば，検索語が「話」であればOR(話,話あ,…,話遥)が検索条件となる．

n-gram索引では，nの値が検索性能に影響する．nが大きくなると，索引サイズは大きくなり，登録時間も増加する．一方，検索時間への影響は，検索語の長さに応じて異なる．検索語がnより長い場合，nが大きいほど検索語の分割数が少なくなるので検索時間は短縮される．ところが，検索語がnより短い場合，nが小さいほど検索語を展開するn-gram数が少なくなるので検索時間は短縮される．これら影響のバランスを考え，日本語を対象とした場合n = 1〜3が使用されることが多い[4)5)6)7)]．

2-3　単一検索語検索の高速化

単一検索語検索の3つのケースの中では，n-gramが文書中で検索語を構成しているかの位置検査が必要なので，nより長い検索語の処理に時間がかかり，その高速化の必要性が大きい．位置検査のためには，登録時にn-gramの文書中での出現位置を索引**に記録し，検索時にn-gramの出現位置の突き合わせを行えばよい[6)]．

長い検索語の検索処理を高速化するには，位置検査の処理コストを削減すればよい．その手法として，筆者らは2重最適化法を提案した[7)]．2重最適化法では，検索処理を，検索語から抽出されたn-gramを全て含む文書を特定する候補文書特定と，候補文書においてn-gramが連続した位置に出現し検索語を構成するかを調べる位置検査の2段階に分けて処理する．そして，2つの処理において使用するn-gram群を以下のように使い分けることで個別に最適化し，位置検査の処理コストを削減する．

(1)　候補文書を少なくするため，候補文書特定では，検索語から抽出される全n-gramを使用する．ただし，前後に位置する位置検査に使用するn-gramよりも文書頻度(n-gramを含む文書数)の大きいものは候補文書を削減するのに有効でないので，そうしたn-gramは除外する．

(2)　個々の位置検査のコストを削減するため，位置検査では，検索語を被覆し，かつ各n-gramの文書頻度の和が最小となるn-gram群を使用する．

なお，検索処理手順には，文書順(document-order)と索引単位順(term-order)の二つがある[8)9)]．文書順では，一つずつ文書を取り上げ，その文書が検索語を含んでいるか検査して，検索結果を決定していく．これに対し，索引単位順では，まず最初の索引単位に該当する文書集合を中間結果とし，それ以降は残りの索引単位を一つずつ取り上げ，その索引単位に該当する文書集合と前段の中間結果との合成結果を新たな中間結果とするという処理を繰り返すことで検索結果を決定する．n-gram索引における検索語処理では，中間結果としてn-gramの位置情報を保持しておく必要のない文書順の方が効率的であり[8)9)]，2重最適化法でも文書順を採用した．

** 索引の代表的なファイル形式には転置ファイルとシグネチャーファイルの 2つがあるが，出現位置を記録するためには前者を用いる必要がある．

2重最適化法の処理手順を，検索語を「携帯電話」として説明する．この検索語からは「携帯」「帯電」「電話」が抽出され，候補文書特定に使用される．一方，位置検査に使用されるのは，検索語を被覆する最小個数のn-gramである「携帯」「電話」である．Fig.1に処理手順を示す．ここで，n1,n2,n3が「携帯」「帯電」「電話」に対応し，djが文書，斜体の数字が処理順序を表している．検索処理は，n1,n2,n3の全てが出現している文書を見つけ，その文書が見つかったらn1,n3を用いて位置検査を行うという手順で進む．この例では，d3が最初の，d6が2番目の候補文書となる．なお，d5は位置検査に使用されるn1,n3が存在しているが，候補文書特定だけに使用されるn2が存在していないので，候補文書にならない．

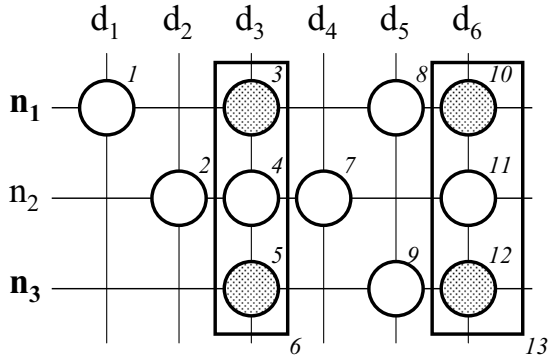

Fig.1　Evaluation process example for a single keyword.

3. 複合条件検索の高速化

本章では，まずAND・OR・ANDNOTの各演算子の高速化方法を示す．さらに，複数の演算子が混合した混合条件の高速化の方法も提案する．

3-1　AND演算子

AND演算子の処理方法としては，中間結果を保持しておく必要のない文書順が効率的であることが知られている[9]．文書順による処理では，全ての索引語が出現する文書を文書IDの小さい順に見つければ良く，アルゴリズムはFig.2のようになる．

(1) 検索語 k を最初の検索語に、targetDocId = 1 とする。
(2) k について targetDocId 以上で最小の文書を検索する。検索できない場合、検索処理を終了する。
(3) 検索文書（matchDocId とする）が targetDocId より大きければ、targetDocId = matchDocId とする。
(4) k が最初の検索語であれば k を次の検索語に、そうでなければ k を最初の検索語にし、ステップ(2)に戻る。
(5) k が最後の検索語であれば、targetDocId が該当文書であるので、targetDocId を検索結果に加える。targetDocId をインクリメントし、k を最初の検索語にしてステップ(2)に戻る。
(6) k を現在の targetDocId では検索していない次の検索語にしてステップ(2)に戻る。

Fig.2　Basic algorithm for AND operator.

n-gram索引では，検索語の長さに応じて処理方法が異なることを考慮しなければならない．検索語の長さがnに等しい場合は，単語索引と状況は同じであり，上記アルゴリズムをそのまま用いればよい．短い場合は，短い検索語は複数のn-gramでOR展開されるためANDとORの入れ子条件になる．入れ子条件については3-4節であらためて検討する．長い場合は，位置検査が必要になるため，それをいかに効率的に処理するかが検索時間に大きく影響する．そこで，以下では，長い検索語を含むAND条件について考察する．

AND条件では，全ての検索語を含む文書が検索結果になる．すなわち，検索結果に含まれる文書は，少なくとも全ての検索語について候補文書でなければならない．したがって，全ての検索語について候補文書となっていない文書に関する位置検査を行う必要はない．ところが，基本アルゴリズムでは，ステップ(2)において検索語を確実に含む文書を検索しているので，その際に見つかる候補文書全てについて位置検査が行なわれ，せっかく実施した位置検査が無駄になる場合がある．

3つの検索語から成るAND演算子の処理の様子(Fig.3)を使ってこのことを説明する．Fig.3で，diは文書，kjは検索語，(di,kj)の交点の白丸はdiがkjの候補文書であること，網掛けの丸はdiがkjの該当文書(実際にkjを含む文書)であることを示す．基本アルゴリズムでは，d1,d2のように全ての検索語について候補文書となっていない文書についても位置検査が行なわれるが，位置検査を行う必要があるのはすべての検索語が候補文書となるd3,d6に限られる．したがって，Fig.3の

例では，太線で囲まれていないケースについては位置検査を省略可能である．

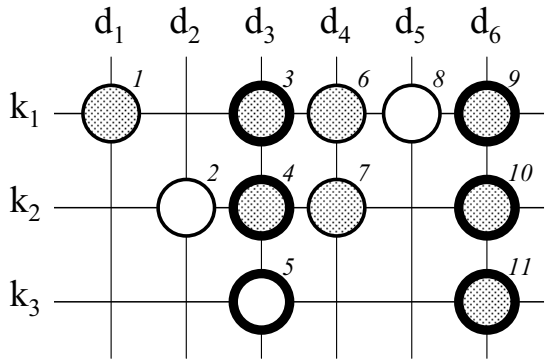

Fig.3　Evaluation process example for AND operator.

不要な位置検査を省略するためには，基本アルゴリズムのステップ(2)(5)を以下のように改良すればよい．

(2)　kについてtargetDocId以上で最小の候補文書を検索する．検索できない場合，検索処理を終了する．

(5)　kが最後の検索語であれば，targetDocIdに対して全ての検索語について位置検査を行なう．全ての検索語が出現していればtargetDocIdが該当文書であるので，targetDocIdを検索結果に加え，そうでなければ加えない．targetDocIdをインクリメントし，kを最初の検索語にしてステップ(2)に戻る．

この修正により全ての検索語について候補文書となる文書以外では位置検査が行われなくなるので，位置検査回数は大幅に減り，検索時間の短縮が期待できる．

3-2　OR演算子

OR演算子では，AND演算子とは異なり，文書順よりも検索単位順の方が効率的である[10]．これは，OR演算子を文書順で実現するためには，ある時点での全ての検索語に該当する文書のなかで最小の文書IDを持つ文書を見つけなければならない．しかし，最小の文書を見つけるには，各検索語に対する最小文書IDをヒープ構造などを用いて常に管理しておく必要があるが，その処理の負荷が大きいからである．一方，索引単位順であれば，処理途中では常に1つの検索語だけが処理対象となるので，こうした問題は発生しない．索引単位順によるアルゴリズムをFig.4に示す．

(1) 最初の検索語について該当する文書をすべて検索し、検索結果とする。
(2) k を次の検索語とする。次の検索語がない場合、検索処理を終了する。
(3) targetDocId = 1 とする。
(4) k について targetDocId 以上で最小の文書を検索する。検索できない場合、ステップ(2)に戻る。
(5) 検索文書（matchDocId とする）が検索結果に含まれていなければ、matchDocId を検索結果に追加する。
(6) targetDocId = matchDocId + 1 とし、ステップ(4)に戻る。

Fig.4　Basic algorithm for OR operator.

基本アルゴリズムをベースに，n-gram索引に対する検索処理の高速化方法を検討する．OR条件では，どれか1つの検索語を含む文書は検索結果に含まれるので，すでに検索結果に含まれることが確定した文書について別の検索語が含まれるかを調べる必要はない．ところが基本アルゴリズムでは，ステップ(4)において常に位置検査を行うので，位置検査が無駄になることがある．Fig.5の状況では，例えば，d3に対してk2,k3の位置検査を行なうが，d3はk1を処理した時点で結果に含まれることが確定しているので，これらの位置検査は不要である．同様に，太線で囲まれていないケースについては位置検査を省略可能である．

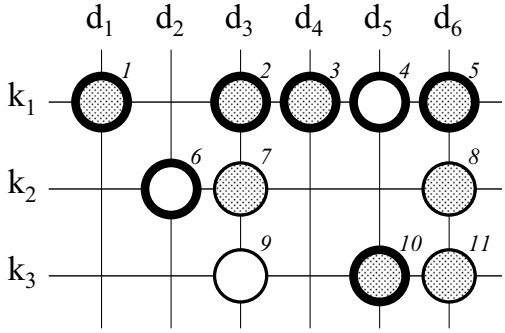

Fig.5　Evaluation process example for OR operator.

不要な位置検査を省略するためには，基本アルゴリズムのステップ(4)(5)を以下のように改良すればよい．

(4)　kについてtargetDocId以上で最小の候補文書を検索する．検索できない場合，ステップ(2)に戻る．

(5)　検索文書(matchDocId)が検索結果に含まれていなければ，kについて位置検査を行なう．kが出現していればtargetDocIDを検索結果に追加し，そうでなければ加えない．

3-3　ANDNOT演算子

ANDNOT演算子は，第2子ノードが補集合を求めるNOT演算子であるAND演算子と捉えることも可能であり，処理手順としてはANDと同様に文書順が効率的である．ANDNOT演算子の基本アルゴリズムはFig.6のようになる．なお，ステップ(5)で使用されるmaxDocIdは文書IDの最大値であり，maxDocId+1は，全ての登録文書の文書IDよりも大きい値になる．

(1) targetDocId = 1, matchDocId2 = 0 とする。
(2) 検索語 1 について targetDocId 以上で最小の文書を検索する。検索できない場合、検索処理を終了する。
(3) 検索文書（matchDocId1 とする）が matchDocId2 より小さければ、matchDocId1 が該当文書であるので、matchDocId1 を検索結果に加える。targetDocId = matchDocId1 + 1 とし、ステップ(2)に戻る。
(4) matchDocId1 が matchDocId2 に等しければ、matchDocId1 は該当文書ではない。targetDocId = matchDocId1 + 1 とし、ステップ(2)に戻る。
(5) matchDocId1 が matchDocId2 より大きければ、検索語 2 について matchDocId1 以上で最小の文書を検索する。検索文書を matchDocId2 とするが、検索できなければ matchDocId2 = maxDocId + 1 とする。ステップ(3)に戻る。

Fig.6　Basic algorithm for ANDNOT operator.

ANDNOT条件では，検索語2のみを含む文書は検索結果に含まれない．したがって，検索語1を含まない文書に対しては検索語2の候補文書であっても，位置検査を行う必要がない．ところが，基本アルゴリズムでは，ステップ(5)において常に位置検査を行うことになるので，位置検査が無駄になることがある．Fig.7の状況では，d2,d3に対してk2の位置検査を行なっているが，d2,d3はk1を含んでいないので，これらの位置検査は省略可能である．

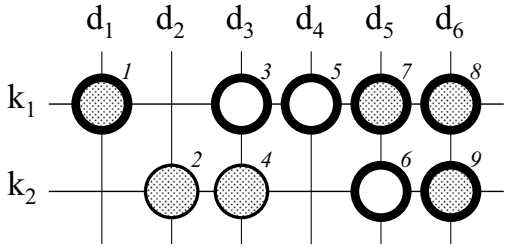

Fig.7　Evaluation process example for ANDNOT operator.

不要な位置検査を省略するためには，基本アルゴリズムのステップ(4)(5)を以下のように改良すればよい．

(4)　matchDocId1がmatchDocId2に等しければ，検索語2について位置検査を行なう．検索語2が出現していなければ該当文書であるので，matchDocId1を検索結果に加え，そうでなければ加えない．TargetDocId= matchDocId1+1とし，ステップ(2)に戻る．

(5)　matchDocId1がmatchDocId2より大きければ，検索語2についてmatchDocId1以上で最小の候補文書を検索する．検索文書をmatchDocId2とするが，検索できなければmatchDocId2=maxDocId+1とする．ステップ(3)に戻る．

3-4　混合条件

前節まででは，演算子がひとつの場合を扱った．ところが，現実の検索場面では，複数の演算子が組み合わせられることもある．特に，ひとつの概念を複数の検索語のOR演算子で表現し，それらをAND演算子で結合した条件が多く用いられる[1]．さらにn-gram索引では，3-1節で述べたように，AND演算子のなかに短い検索語が含まれる場合，短い検索語はOR演算子で展開されるので，ユーザがAND演算子しか用いていなくとも，OR演算子とAND演算子が入れ子になった検索条件の処理が必要になる．そこで，この節ではOR演算子とAND演算子が入れ子になった混合条件の高速化について検討する．

AND演算子の子ノードとしてOR演算子が存在する場合，AND演算子の処理は文書順で行われるため，その子ノードであるOR演算子の処理も文書順で行われる．ところが，前述のようにOR演算子の文書順処理は非効率であるので，混合条件の処理も非効率となる．一方，混合条件は，ド＝モルガンの法則を使って等価なOR標準形(OR演算子の子ノードとしてAND演算子が存在する形式)に変換することができる[1]．OR標準形であれば，OR演算子は索引単位順，その子ノードであるAND演算子は文書順と，それぞれに適した処理手順を用いることができるので，検索処理も効率化できる．

ただし，n-gram索引では，AND演算子の子ノードとして短い検索語が含まれる場合，非常に多くの子ノードを持つOR演算子がAND演算子の子ノードになることを考慮しなければならない．こうした条件では，OR標準形への変換コス

トが大きいので，変換により検索時間が増大する可能性がある．したがって，混合条件を常にOR標準形に変形することが望ましいとは限らない．そこで，OR標準形に変換した場合の子ノードの数を見積もり，それがある値(OR標準形変換閾値)以下であるときに限ってOR標準形に変換することとする．なお，OR標準形変換閾値の適切な値については，4-5節で実験的に検討する．

4. 評価実験

4-1 対象データ

本研究で提案したn-gram索引に対する複合条件検索の効率的な処理方法を評価した．

評価には新聞記事5年分(毎日新聞CD-ROM91〜95年版)を使用した．このCD-ROMでは記事ごとにキーワードなども付与されているが，見出しと本文を合わせて1文書として登録を行なった．5年分の合計で，登録件数は496997件，テキストサイズは441.5MB(1件当たり932B)である．実験ではbi-gramを索引単位としてこれら文書を索引付けした．

検索条件としては，以下の4セットを用意した．なお，いずれの検索条件も検索結果が0件となることはない．

(1) AND条件
3文字以上の検索語を少なくとも1つ含む2文字以上の検索語をAND結合したもの40個
例：AND(複写機,消費電力,発熱,低減)

(2) OR条件
3文字以上の検索語を少なくとも1つ含む2文字以上の検索語をOR結合したもの40個
例：OR(内線,構内回線,PBX)

(3) ANDNOT条件
3文字以上の検索語を少なくとも1つ含む2文字以上の検索語をANDNOT結合したもの30個
例：ANDNOT(液晶,ディスプレイ)

(4) 混合条件
3文字以上の検索語を少なくとも1つ含む検索語をAND,ORで結合したもの30個
例：AND(OR(感光体,ドラム),OR(冷却,風))

検索時間は，SUN Ultra30 (CPU:UltraSPARC II 296MHz)を使用し，索引ファイルはローカルディスクに置いて測定した．索引ファイルが全くメモリ上に読み込まれていないcold start条件と，データがキャッシュされているhot start条件の2つの状況について測定した．各検索条件について5回ずつ測定し，最大値と最小値を除いた3回の平均値を求め，さらに各セットごとに平均を取った．

4-2 AND条件の結果

AND条件に対する検索時間をTable 1のANDの行に示す．ここで，基本が基本アルゴリズム，改良が本研究で提案した改良アルゴリズムによる検索時間(秒単位)であり，改良欄の括弧内の数字は基本アルゴリズムに対する改良アルゴリズムの増減比を表している．

Table 1 Response time for single operator queries.

	cold start			hot start		
	基本	改良		基本	改良	
AND	0.667	0.639	(-4.2%)	0.037	0.021	(-43.2%)
OR	0.761	0.740	(-2.8%)	0.108	0.097	(-10.2%)
NOT	0.752	0.682	(-9.2%)	0.198	0.149	(-24.7%)

Cold start・hot start条件ともに改良アルゴリズムによって検索時間は短縮されており，提案方法の有効性が確認できる．ただし，増減比を比較すると，hot startでは大きく減少しているに，cold startでは小さな値となっている．これはhot startでは必要なデータがメモリ上に読み込まれているので，位置検査に伴う突き合わせ処理の減少分が直接的に検索時間に反映されるのに対して，cold startではディスクアクセスの占める割合が高いが，ディスクアクセスは突き合わせの処理の減少に比例して削減するわけではないからである[7]．

4-3 OR条件の結果

OR条件に対する測定結果をTable 1のORの行に示す．ここでも，改良アルゴリズムにより，検索時間が短縮されている．ただし，AND条件と比較すると短縮の割合は小さい．この差異が生じたのは，位置検査を省略できる状況がANDとORで異なるからである．AND演算子では，全検索語について候補文書と判断された文書以外では，各検索語の候補文書において位置検査を省略可能なので，実際に省略されるこ

とが多い．一方OR演算子では，各検索語の候補文書がすでに処理した検索語のいずれかで該当文書と判断された場合においてのみ位置検査を省略できるが，これに該当するのは複数の検索語がもともと同じ文書に出現している場合に限られるため，省略されることが少ないのである．

4-4　ANDNOT条件の結果

ANDNOT条件に対する評価結果をTable 1のNOTの行に示す．ここでも，検索時間も短縮されており，改良アルゴリズムの有効性が確認できた．短縮効果の傾向は，OR条件よりもAND条件に近い．これは，ANDNOT演算子はAND演算子の一種と捉えることができ，位置検査が省略できる場合が多いからと考えられる．

4-5　混合条件の結果

混合条件に対しては，OR標準形変換閾値を1,2,5,10,20,50,100,200,500,1000,2000と変化させ，OR標準形への変換が有効であるか調べた．検索時間は，基本・改良の検索アルゴリズムそれぞれについて測定した．測定結果をTable 2に示す．

Table 2　Response time for mixed queries.

	cold start			hot start		
	基本	改良		基本	改良	
1	1.284	1.310	(+2.0%)	0.286	0.302	(+5.6%)
2	1.286	1.310	(+1.9%)	0.289	0.303	(+4.8%)
5	1.252	1.235	(-1.4%)	0.296	0.284	(-4.1%)
10	1.226	1.191	(-2.9%)	0.293	0.270	(-7.8%)
20	1.246	1.203	(-3.5%)	0.318	0.291	(-8.5%)
50	1.246	1.201	(-3.6%)	0.319	0.289	(-9.4%)
100	1.246	1.187	(-3.8%)	0.313	0.283	(-9.6%)
200	1.236	1.187	(-3.8%)	0.314	0.283	(-9.9%)
500	1.235	1.188	(-3.8%)	0.314	0.283	(-9.9%)
1000	1.245	1.200	(-3.8%)	0.325	0.295	(-9.2%)
2000	1.314	1.266	(-3.9%)	0.395	0.361	(-8.6%)

変換閾値の影響は，3-4節で予想した通りの傾向が観測された．すなわち，改良アルゴリズムでは，閾値が1の場合よりも2以上の方が検索時間は短く，OR標準形への変換が有効であることがわかる．しかし，閾値が1000以上では明らかに検索時間は増大しており，閾値は10〜500程度であることが望ましい．なお，1000以上で検索時間を増大させているのは検索条件中に1文字検索語が混ざっている検索条件(例えばAND(OR(感光体,ドラム),OR(冷却,風)))であった．

基本と改良の両検索アルゴリズムを比較すると，変換閾値が1,2の場合を除いては，改良の方が高速である．前段で述べたように，変換閾値が1,2はそれ以上の場合よりも検索時間がかかっているため，変換閾値としては不適切である．したがって，混合条件を処理する場合においても改良アルゴリズムの方が優れていると結論付けることができる．変換閾値が1,2の場合に改良アルゴリズムの性能が悪いのは，現在のOR演算子の実装では，与えられたtargetDocId以上で最小の候補文書の検索が，ORの子ノードに対して位置検査を伴う検索を呼び出すようにコーディングされていることに原因があると考えられる．すなわち，OR標準形への変換を行わずORがANDの子ノードにある場合，ANDが改良アルゴリズムで動作すると，ORの子ノードに位置する検索語についてはANDの検索アルゴリズムのステップ(3)とステップ(6)の2ヵ所で位置検査が行われてしまうのである．OR演算子の最小候補文書の検索の実装を，子ノードに対して位置検査の伴わない候補文書検索を行うように改良すれば，この問題は回避できると思われる．

なお，先ほど変換閾値としては10〜500程度の値が適切であると述べたが，この範囲であれば，検索時間はほぼ一定である．これはパラメータ調整にあまり気を遣う必要がないことを意味しており，検索システムにとって望ましい特性である．

5. 今後の展開

本研究では，n-gram索引のための複合条件の効率的な処理手法を提案した．基本的な考えは，長い検索語処理に伴う位置検査を可能な限り省略することで検索を高速化するというものである．AND,OR,ANDNOT演算子の処理アルゴリズムを見直し，位置検査の削減する改良アルゴリズムを提案した．さらに，AND,ORが入れ子になった混合条件に対しては，OR標準形に変換した場合の子ノードの数に応じて選択的にOR標準形に変換するという手法を提案した．新聞記事5年分を用いて評価した結果，提案手法の有効性が確認できた．

本研究の高速化手法は，ソフトウェア研究所で開発中の

全文検索サーバーに実装されている．全文検索サーバーは，データベース管理システムG-BASE V3.5と連携して動作し，大量文書の高速検索を実現する．G-BASE V3.5は，現在，筑波大学などの電子図書館システムで使用されており，99年下期からは社内特許システムNS2でも使用される予定である．また，全文サーバー単体では，オフィス文書管理システムLIFISAの次期バージョンの開発に使用されている．

謝辞

本研究を推進するにあたりご協力をいただいたソフトウェア研究所第1研究室，および(株)リコーシステム開発の方々に感謝いたします．

参考文献

1) I. Witten, A. Moffat, T. Bell: Managing Gigabyes: Compressing and Indexing Documents and Images, Van Nostrand Reinhold (1994).

2) H. Fujii, B. Croft: A Comparison of Indexing Techniques for Japanese Text Retrieval, Proc. Of 16th ACM SIGIR Conf. (1993) pp. 237--246.

3) A. Robertson, P. Willett: Application of n-grams in Textual Information Systems, Journal of Documentation, 54, 1 (1998) pp. 48-69.

4) 赤峰,福島:高速全文検索のためのフレキシブル文字列インバージョン法,情報処理学会アドバンストデータベースシステムシンポジウム予稿集,(1996) pp. 35-42.

5) 松井,難波,井形:大容量情報全文検索エンジンTerass, FUJITSU, 48, 3 (1997) pp. 240-243.

6) 菊池:日本語文書用高速全文検索の一手法,電子情報通信学会論文誌,J75-D-I, 9 (1992) pp. 836-846.

7) 小川,松田:n-gram索引を用いた効率的な文書検索法,電子情報通信学会論文誌,J82-D-I, 1 (1999) pp. 121-129.

8) H. Turtle, J. Flood: Query Evaluation: Strategies and Optimizations, Information Processing & Management, 31, 6 (1995) pp. 831-850.

9) E. Brown: Fast Evaluation of Structured Queries for Information Retrieval, Proc. Of 18th ACM SIGIR Conf., (1995) pp. 30-38.

10) M. Kaszkiel, J. Zobel: Term-ordered Query Evaluation versus Document-ordered Query Evaluation for Large Document Databases, Proc. Of 21st ACM SIGIR Conf., (1998) pp. 343-344.