索引篩法 - 大規模サーチエンジンのための高速なランキング検索法

原田 昌紀　佐藤 進也　風間 一洋

日本電信電話株式会社 NTT 未来ねっと研究所
〒180-8585 東京都武蔵野市緑町 3-9-11 NTT 武蔵野研究開発センタ

E-mail: {harada, sato, kazama}@ingrid.org

あらまし　今日の文書検索システムの多くは転置索引方式を採用して高速かつ効率のよい検索を実現している．しかし，Web サーチエンジンのように検索対象となる文書集合が大規模になると，転置索引から読み出される出現位置情報も増大し，検索速度が低下してしまう．本論文では，利用者が閲覧するのは検索結果の上位のみであることに着目して，索引から読み出される出現位置情報の量を削減し，検索を高速化する索引篩法を提案する．索引篩法では，あらかじめ適合度に大きく寄与する出現位置情報のみを格納した索引を用意し，適合度の高い文書を最初に検索する．大量の Web ページと実際の Web サーチエンジンで検索された質問を用いた実験により，提案手法の有効性を評価した．

キーワード　転置索引，情報検索，サーチエンジン，性能評価

Index Sieving – A Fast Ranking Search Method for Large-scale Search Engines

Masanori HARADA　Shin-ya SATO　Kazuhiro KAZAMA

NTT Network Innovation Laboratories, NTT Corporation
NTT Musashino R&D Center　3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan

E-mail: {harada, sato, kazama}@ingrid.org

Abstract　Most of today's document retrieval systems use inverted indices to implement fast and efficient search. However, as the targeted document collection is getting large, postings read from the inverted index also become large and search speed is sacrificed. In this paper, we propose *index sieving*, a method of reducing the size of postings read from the index to accelerate search. With index sieving, we first search highly relevant documents using an index which stores postings that will largely contribute to relevance scores. Effectiveness of the proposed method is evaluated by experiments using a large number of web pages and queries to a real web search engine.

Keyword　Inverted Index, Information Retrieval, Search Engine, Performance Evaluation

1. はじめに

今日の文書検索システムの多くは，索引の構成法として転置索引を採用している．転置索引とは索引語をキーとして，その索引語のすべての出現位置情報を格納する索引である．転置索引を用いると，文書ファイルを参照せずに検索語を含む文書を列挙できるため，大量の文書を高速に検索できる．

転置索引のもう一つの特長として，ランキング検索を効率的に実現できることが挙げられる．ランキング検索とは，検索質問の文書の適合度を計算し，適合度の降順に検索結果を表示する検索方式である．適合度の計算に用いられる主要なパラメータとして，検索語の文書内頻度と文書頻度(検索語が出現する文書の数)があるが，これらは転置索引から出現位置情報を読み出す過程で求められる．一方，索引の構成法としてシグネチャファイルや接尾辞配列を採用した場合，索引本体だけでは文書内頻度や文書頻度を求めることができない．このような理由から，転置索引はランキング検索をおこなう文書検索システムに適した索引構成法といえる．

しかし，Web サーチエンジンのように極めて大量の文書を対象とする場合には，たとえ転置索引を用いた

としても，索引から読み出される出現位置情報も大量になり，高速にランキング検索をおこなうことが難しくなる．そのため，実際の大規模な Web サーチエンジンでは，索引を分割し，多数の計算機で並列に検索をおこなっているが[1][2]，この方法には多数の計算機の導入・維持に要するコストの問題が伴う．

本論文では転置索引を用いたランキング検索を高速化する方法として**索引篩(ふるい)法**を提案する．ランキング検索がおこなわれる場合，どれほど大量の文書が検索されるとしても，実際に利用者が参照するのは適合度順で上位の文書のみである．その場合，仮にそれ以降の文書の検索を省略しても，利便性を損なうことはないと考えられる．そこで索引篩法では，あらかじめ適合度に大きく寄与する可能性のある出現位置情報のみを格納した小さな転置索引を用意し，それを用いて適合度の高い文書のみを検索する．これによって，ディスクから読み出される出現位置情報の量と，適合度の計算などに伴う CPU 処理コストが削減され，ランキング検索が高速化される．

ただし，任意の検索質問に対して，あらかじめ高い適合度を得る文書を特定しておくことはできない．そこで，本論文では Web サーチエンジンでよく用いられる一つの検索語のみからなる単純な検索質問に限って索引篩法を適用する．この場合は，検索語の文書内頻度と文書サイズから事前に適合度を知ることができる．

本論文の構成は次の通りである．2 節では一般的な転置索引の構成とランキング検索の手順を説明する．3 節では索引篩法の概略を述べ，高適合語索引の作成方法と高適合語索引を使ったランキング検索の手順を説明する．4 節では評価実験について報告する．5 節では関連する研究について述べる．最後に 6 節において，まとめと今後の課題を述べる．

2. 転置索引を用いたランキング検索

2.1. 転置索引の構成

転置索引は図 1に示すように，辞書ファイルと位置情報ファイルの二つから構成されるのが一般的である．

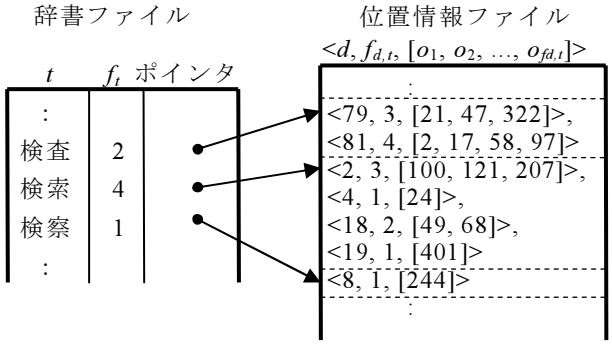

図1 転置索引の一般的な構成

辞書ファイルは文書集合中のすべての索引語を格納し，索引語をキーとして，その文書頻度，出現位置情報を格納したレコードへのポインタなどを対応づける．一方，位置情報ファイルは索引語 t ごとに出現位置情報のリストを格納したレコードの集まりである．出現位置情報は文書番号 d，文書内頻度 $f_{t,d}$，文書内出現位置（文書の先頭からのバイト数）o_i のリストの組である．通常，レコード中の文書番号と文書内出現位置は昇順に並べられ，数値の差分をとることにより圧縮された形で格納される[3][4].

ここで索引語とは本来は文書の内容を特徴づける単語やフレーズのことをいうが，今日では文書中のすべての単語を索引語とするのが一般的である．日本語では語の境界が明確ではないため，索引語の単位として形態素あるいは N グラムを用いることが多い．以下，本論文では N グラムを前提として議論を進めるが，索引篩法は形態素を索引語の単位とする場合にも適用できる．

N グラムを索引語の単位とする場合，文書内出現位置を用いることで，N よりも長い任意の検索語を検索できる．たとえば，“携帯電話”という検索語は，バイグラム“携帯”とバイグラム“電話”の文書番号が一致し，文書内出現位置の差が 2 になる位置を求めることで検索できる．本論文ではこれをフレーズ判定と呼ぶ．検索語の長さが N 未満の場合は，辞書ファイルで前方一致検索をおこない，検索語で始まるすべての N グラムの出現位置情報をマージする．

2.2. 検索質問と文書の適合度

多くの検索モデルでは，まず検索質問に含まれる検索語ごとに適合度を計算し，それらを線形結合することで検索質問と文書の適合度を求める．すなわち，検索質問 q と文書 d の適合度 $S_{q,d}$ は次式のように表わされる．

$$S_{q,d} = \sum_{t \in q} w_{t,q} \cdot S_{t,d}$$

ここで $w_{t,q}$ は検索質問 q における検索語 t の重み，$S_{t,d}$ は検索語 t と文書 d と適合度である．

検索語と文書の適合度は，索引語がどれだけその文書を特徴づけているかを表わすスコアと同様の考え方で計算できる．一般に索引語のスコアは次のような情報から計算される．

- 文書内頻度　　$f_{t,d}$
- 文書サイズ　　$|d|$
- 文書頻度　　f_t

文書内頻度は，索引語と文書の内容の関連の強さを表すと考えられる．ただし，大きな文書では文書内頻度も大きくなりやすいため，文書サイズを用いて正規化する．また，文書頻度の高いありふれた索引語は特定の文書を弁別する効果に乏しいと考えられる．したが

って，一般に検索語のスコアは，文書内頻度に対して単調増加し，文書頻度と文書サイズに対して単調減少するような関数によって計算される．

検索質問が一つの検索語のみからなる場合は，検索語の適合度がそのまま検索質問の適合度となる．さらに，この場合は検索語の文書頻度を定数とみなせるため，次式のように文書内頻度と文書サイズのみから検索語のスコアを計算し，適合度とすることができる．

$$S_{q,d} = S_{t,d} = \mathrm{score}(f_{t,d}, |d|)$$

文書サイズに加え，PageRank[1]や Web ページの更新時刻といった検索質問と関係なく決まるパラメータを正規化に用いる場合にも同じことがいえる．

2.3. 転置索引を用いたランキング検索の手順

図 2に転置索引 *I* を用いて検索語 *phrase* を検索し，適合度の高い文書上位 *k* 個を求める手順を示す．ただし，検索語は N より長いものとする．

まず，検索語を索引語（Nグラム）に分解し，それぞれの索引語を辞書ファイルで検索する．このとき，検索語を被覆するNグラムの組が複数通りある場合には，フレーズ判定のコストが最小となる組み合わせを選択するが[5]，詳細は省略する．続いて，それらの出現位置情報のリストを読み出し，フレーズ判定をおこなって検索語の出現位置情報を求める．

後半では出現位置情報から文書毎に検索語のスコアを計算し，適合度の高い文書を求める．ここではヒープを優先順位つきキューとして用いることでメモリ上に保持する検索結果の候補となる文書を高々*k*+1 個に抑えている．また，7 行目で出現位置情報をすべて読み込むのではなく，11 行目で検索語の出現位置情報が必要になった時点で，各 N グラムの出現位置情報を位置情報ファイルから並行に読み出しながらフレーズ判定をおこなうようにすれば，出現頻度の高い索引語があっても，限られた大きさのバッファで検索することができる．

```
search(phrase, I, k)
 1. terms = tokenize phrase into N-grams;
 2. streams = new array of postings;
 3. for each term in terms
 4.     tentry = find the entry of term in I;
 5.     if (tentry == NOTFOUND)
 6.         return NOTFOUND;
 7.     tstream = retrieve postings of tentry from I;
 8.     append tstream to streams;
 9. pstream = join streams to create phrase postings;
10. heap = create empty heap;
11. for each posting entry p in pstream
12.     p.score = score(p.tf, docsize(p.docid));
13.     heap.push(p);
14.     if (heap.size() > k) heap.pop();
15. return entries in heap;
```

図2　一般的なランキング検索の手順

文書内頻度は出現位置情報に含まれているが，文書頻度はフレーズ判定を終えるまで知ることができないため，一般には出現位置リストを読み出しながら検索語の適合度を計算することはできないという問題がある[6]．しかし，前節で述べたように，適合度の計算に文書頻度を用いない場合にはこの問題は回避される．

なお，文書サイズは検索質問とは関係なく静的に決まるため，転置索引とは別に文書サイズ表を用意するのが一般的である．この表はさほど大きくはならないため，表全体をメモリ上に置くことで文書集合の大きさに関わらず一定時間で参照できる．適合度の計算においては，文書サイズは必ずしも正確である必要はないため，少ないビット数に圧縮することもできる[14]．

2.4. 転置索引を用いたランキング検索の速度

転置索引を用いたランキング検索は次のようなステップから構成される．

1. 辞書ファイルからの索引語エントリの検索
2. 位置情報ファイルからの圧縮された出現位置情報の読み出し
3. 圧縮された出現位置情報の展開
4. フレーズ判定
5. スコアの計算とヒープの操作

辞書ファイルは通常 B+木などで構成されるため，大量の索引語が登録されている場合でも，高々数回のディスクアクセスで検索できる．そのため，ステップ 1 の処理コストは，索引が大規模になっても大きく増えることはない．しかし，ステップ 2 以降の処理コストは，いずれも出現位置情報の量に比例する．つまり，ランキング検索に要する時間は検索語を構成する索引語の出現頻度の総和におおむね比例すると考えられる．

3. 索引篩法

3.1. 索引篩法の適用対象

本論文では索引篩法の適用対象として Web サーチエンジンを想定している．一般に Web サーチエンジンで用いられる検索質問は短く，利用者はランキング検索結果の上位しか閲覧しない傾向がある．英語圏の Web サーチエンジン Excite の場合，検索質問の長さは平均 2.21 語であり，利用者は 1 ページに 10 文書ずつ表示される検索結果を平均 2.35 ページしか閲覧していない[7]．日本語では複合語が用いられるため，検索語数はより小さくなる．表 1 に日本語 Web サイトを主な検索対象とする Web サーチエンジン ODIN において，

2002年1月に用いられた検索質問に含まれる検索語数の分布を示す．この期間の平均検索語数は1.40であり，検索質問の7割以上が一つの検索語のみから構成されていた．そこで，本論文では単一の検索語からなる検索質問に限定して議論を進める．

表1　ODINで検索された質問に含まれる検索語数

検索語数	割合
1	70.96%
2	21.03%
3	6.15%
4	1.36%
5	0.35%
6	0.09%
7	0.04%
8	0.007%
9	0.003%

3.2. 索引篩法の概略

ある索引語がある文書でとるスコアは文書内頻度と文書サイズから静的に計算できる．そこで，索引篩法では閾値Fを任意に定め，索引語ごとにF以上のスコアをとる文書を特定し，その文書内の出現位置情報のみを通常の転置索引とは別の転置索引に格納しておく．これを**高適合語転置索引**と呼ぶ．

索引語より長い検索語の文書内頻度はフレーズ判定をおこなうまでわからないため，事前に検索語のスコアを知ることはできない．しかし，そうした検索語の文書内頻度は，検索語に含まれる索引語の文書内頻度を超えないことから，高適合語転置索引に格納されなかった出現位置で，検索語のスコアがF以上になることはない．したがって，高適合語転置索引を用いて検索されたスコアF以上の文書はランキングの上位にあると保証できる．索引篩法では，たとえば適合度の上位10個の文書を検索する場合，まず高適合語転置索引を検索し，スコアF以上の文書が10個以上得られれば，その上位10個を検索結果として表示する．10個未満であれば失敗であり，通常の転置索引を用いてあらためてランキング検索をおこなう．

長さN未満の検索語の文書内頻度は，検索語で始まる索引語の文書内頻度の和になるので，それぞれの索引語のスコアがF未満でも，検索語のスコアはF以上になる可能性がある．そのため，検索語の長さがN未満の場合には，初めから通常の転置索引を用いてランキング検索をおこなう．

```
sieve_index(I, F, k_s)
1. create empty index Is;
2. for each term entry tentry in I
3.     if (tentry.df >= k_s)
4.         ps = create empty postings;
5.         pt = retrieve postings of tentry from I;
6.         for each posting entry p in pt
7.             if (score(p.tf, docsize(p.docid)) >= F)
8.                 append p to ps;
9.         if (ps.df >= k_s) save (tentry.term, ps) to Is;
```

図 3　高適合語転置索引の作成手順

3.3. 高適合語転置索引の作成

高適合語転置索引の作成手順を図 3に示す．通常の転置索引Iから，索引語ごとに出現位置リストを読み出し，スコアがF以上になる文書の出現位置情報を抽出して，高適合語転置索引Isに格納する．ただし，検索結果を出力する上で最低でも上位k_s個の文書が必要だとすると，スコアF以上の文書がk_s個未満の索引語は除外する．

3.4. 高適合語転置索引を用いたランキング検索

高適合語転置索引Isを用いて検索語$phrase$を検索し，スコアF以上の文書k個をランキング検索する手順を図 4に示す．

基本的に通常の転置索引を検索する場合の手順と大きくは変わらないが，辞書ファイルを検索した時点で文書頻度がk未満の索引語があれば，ただちに失敗とわかる(FAILURE1)．そうでない場合にはフレーズ検索をおこなって検索語の出現位置を求め，スコアがF以上の文書を検索結果候補として残し，それらがk個未満であれば失敗とする（FAILURE2）．

なお，通常は$k_s \le k$であるが，$k_s > k$としても不正確な検索結果が得られることはない．ただし，$k_s \le k$なら高適合語転置索引のみで検索できる検索語でも，$k_s > k$ではFAILURE1となることがある．

```
search(phrase, Is, F, k)
 1. terms = tokenize phrase into N-grams;
 2. streams = new array of postings;
 3. for each term in terms
 4.     tentry = find the entry of term in Is;
 5.     if (tentry == NOTFOUND || tentry.df < k)
 6.         return FAILURE1;
 7.     tstream = retrieve postings of tentry from Is;
 8.     append tstream to streams;
 9. pstream = join streams and create phrase postings;
10. heap = create empty heap;
11. for each posting entry p in pstream
12.     p.score = score(p.tf, docsize(p.docid));
13.     if (p.score >= F)
14.         heap.push(p);
15.         if (heap.size() > k) heap.pop();
16. if (heap.size() < k) return FAILURE2;
17. return entries in heap;
```

図 4　高適合語転置索引を用いたランキング検索

4. 評価実験

4.1. 検索システムの概要

Web サーチエンジン ODIN 用に開発した全文検索エンジン Jerky をベースに索引篩法を実装し，検索速度の評価をおこなった．Jerky は索引語の単位として N グラムを用いているが，N グラムの長さは字種によって異なり，平仮名では N=3，片仮名では N=4，漢字では N=2 である．字種が変化する境界では N=2 であり，英数字は空白文字で単語として区切る[8].

ODIN が検索対象とする一つの文書は，次のような 6 種類の領域のテキストから構成されている[9].

1. 本文
2. タイトル
3. META 要素の keywords プロパティ
4. META 要素の description プロパティ
5. その HTML ファイルにリンクするアンカーテキスト（別サーバ上にある場合）
6. その HTML ファイルにリンクするアンカーテキスト（同一サーバ上にある場合）

Jerky は出現位置情報として，文書番号，文書内の位置に加えて，領域番号を記録しており，文書内頻度の重み付けに用いている．実験で用いたスコアの計算式を次の通りである．

$$s_{t,d} = \frac{\log(\sum_r w_r f_{t,d,r} + 1)}{0.8 \times \sum_{k=1}^{N} \log(|d_k|)/N + 0.2 \times \log(|d|)}$$

ここで，$f_{t,d,r}$ は文書 d の領域 r における検索語 t の頻度である．w_r は領域 r の重みであり，ODIN では w = {1, 10, 5, 2, 12, 1}としている．たとえば，検索語がタイトル内に 1 回出現した場合のスコアへの寄与は，本文に 10 回出現した場合と同じ大きさになる．

分母は極端に大きい文書や，極端に小さな文書が検索されにくいようにするための補正項であり，ピボット正規化の考え方を参考にしている[10][11]. ピボット正規化では文書内のユニークな索引語数を用いるのが普通だが，N グラム索引では索引語の種類が増加しやすいことを考慮して，かわりに log(|d|)を用いている．ここで|d|は文書 d に含まれる索引語の数（文書内頻度の和）である．

実験では文書集合として JP ドメインを中心に収集した Web ページ約 4300 万ページを用いた．一文書の大きさの平均は約 1500 文字である．また，検索質問の集合として ODIN において 2002 年 1 月に検索された単一の検索語のみからなる検索質問を無作為に 1,000 個抽出して用いた．検索語の多くは固有名詞あるいは複合語であり，国語辞典の見出し語にあるような一般的な言葉はほとんどない．

評価に利用したハードウェアは Pentium III 1GHz, 主記憶 2GB, SCSI HDD5 台のディスクアレイを搭載している．ソフトウェアは Solaris 8，JDK1.3.1 である．

4.2. 出現位置情報量と検索速度の関係

まず，通常の転置索引 I を用いた場合の検索速度を調べた．図 5 に検索語を構成する索引語の出現位置情報の量と検索時間の関係を示す．2.4節で述べた通り，転置索引から読み出される出現位置数に比例して検索時間が長くなることがわかる．平均検索時間は 818 ミリ秒であった．

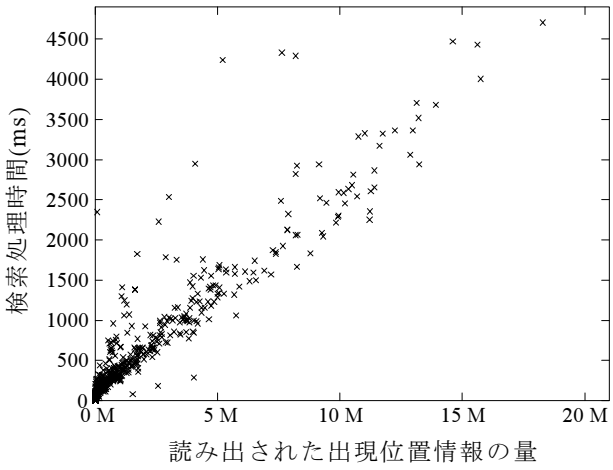

図 5 通常の転置索引を用いた場合の検索時間

4.3. 高適合語転置索引のサイズ

6 つの異なる閾値 F で高適合語転置索引 I_2，I_4，I_7，I_{11}，I_{16}，I_{22} を作成した．I_2 は log(|d|)が平均に等しくなるような文書 d において，重み付けされた文書内頻度の和，すなわち $\Sigma_r f_{t,d,r}$ が 2 となる場合のスコアを閾値 F としている．直感的に言うと，平均以下のサイズの Web ページの本文に 2 回以上登場する索引語の出現位置情報を抽出したものが I_2 である．残りの 5 つも同様である．いずれも k_s=10 とした．

表 2にこれらの高適合語転置索引と通常の転置索引の大きさを示す．ここで|L|は辞書ファイルに含まれる索引語数であり，|P|は位置情報ファイルのサイズ（単位は MB）である．|L|は通常の転置索引の数%にとどまるが，位置情報ファイルはそれほど小さくはならな

表 2 高適合語転置索引のサイズ

	F	|L|	|P|
I	N/A	52,679,269 (100%)	146,288 (100%)
I_2	0.1647	2,329,119 (4.42%)	59,317 (40.5%)
I_4	0.2413	1,546,807 (2.94%)	37,924 (25.9%)
I_7	0.3117	1,220,856 (2.31%)	26,367 (18.0%)
I_{11}	0.3725	979,621 (1.86%)	19,274 (13.2%)
I_{16}	0.4247	491,529 (0.93%)	13,038 (8.9%)
I_{22}	0.4701	320,748 (0.61%)	9,448 (6.5%)

表 3　高適合語転置索引の平均検索時間（単位 ms）

	ALL	FAILURE1		FAILURE2		SUCCESS	
I	818.4	N/A		N/A		818.4	100%
I_2	338.6	6.7	14.7%	531.7	27.0%	332.2	58.3%
I_4	233.4	7.4	19.1%	386.8	31.3%	223.2	49.6%
I_7	173.9	9.3	22.9%	300.6	31.3%	169.1	45.8%
I_{11}	145.7	10.6	26.3%	260.6	30.8%	145.5	42.9%
I_{16}	90.0	8.8	38.8%	186.4	28.6%	101.4	32.6%
I_{22}	63.0	9.9	47.4%	141.1	27.4%	77.1	25.2%

い．これは，一部の索引語の出現位置情報が転置索引で大きな割合を占めているためと思われる．

4.4. 高適合語転置索引の検索時間

3.4 節で述べた手順で 6 種類の高適合語転置索引を検索するのに要した時間を表 3に示す．作成時と同じく k=10 とし，k 個以上の文書が検索された場合のみを成功とした．失敗した場合は図 4に合わせて，原因別に FAILURE1 と FAILURE2 に分けて集計している．

閾値 F が小さく，高適合語転置索引が大きくなるほど，成功率は高くなるが，検索処理時間も長くなるというトレードオフがある．ただ，出現位置ファイルの大きさが通常の半分以上になる I_2 でも成功率は 58.3%にとどまっており，高適合語転置索引のみでは検索できない検索質問がかなり多いことがわかる．実際，評価に用いた検索質問のうち，通常の転置索引を使った場合の検索結果が 10 件以上になるものは 81.4%であり，成功率がこれを超えることはない．その一方で，I_{22} には通常の転置索引の 0.61%の索引語しか格納されていないにもかかわらず，成功率は 25%以上もあり，検索速度も通常の 10 倍以上になっていることから，索引篩法が非常に有効に働く検索質問も多いといえる．

失敗時には通常の転置索引で検索しなおす必要があるため，高適合語転置索引の検索時間はオーバーヘッドになってしまう．FAILURE1 の場合は辞書ファイルの検索のみで終了するためにオーバーヘッドは小さいが，FAILURE2 の場合は成功時以上に検索処理時間が長くなる傾向が見られる．これは 1 つの N グラムから構成される単純な検索語は FAILURE1 か成功のいずれかになり，2 つ以上の N グラムに分割される検索語だけが FAILURE2 になるためと考えられる．

実際に失敗時に通常の転置索引で検索しなおした場合の平均検索時間を表 4に示す．失敗時のオーバーヘッドを含めても，I_2 以外では通常の転置索引のみを用いる場合より平均検索時間が短くなっている．I_{16} を用いた場合が最速であり，11%強の高速化となった．ただし，成功時には高適合語転置索引だけをアクセスするにもかかわらず，表 3よりも少し平均検索時間が長くなっている．これは失敗時には高適合語転置索引と通常の転置索引の両方をアクセスするためにディスクキャッシュが効きにくくなるためと考えられる．

表 4　ランキング検索全体の処理時間（単位 ms）

	ALL	FAILURE1		FAILURE2		SUCCESS	
I	818.4	N/A		N/A		818.4	100%
I_2	841.6	1225.8	14.7%	1628.4	27.0%	379.8	58.3%
I_4	805.3	1051.1	19.1%	1506.9	31.3%	267.3	49.6%
I_7	768.9	985.9	22.9%	1435.3	31.3%	204.3	45.8%
I_{11}	734.0	904.4	26.3%	1377.0	30.8%	167.2	42.9%
I_{16}	727.9	790.6	38.8%	1330.8	28.6%	123.3	32.6%
I_{22}	762.8	864.0	47.4%	1196.3	27.4%	99.8	25.2%

FAILURE1 で高適合語転置索引が大きい場合にオーバーヘッドが大きくなるのも同様の理由と思われる．

4.5. 索引篩法の有効な検索語

表 5 に検索語の文書頻度（通常の転置索引を用いて検索した場合の検索結果数）と I_{16} を用いた場合の成功率の関係を示す．理論上は文書頻度が 10 以上であれば成功する可能性があるが，成功率が高くなり，索引篩法による高速化が期待できるようになるのは文書頻度が 1,000 を超える検索語であるとわかる．そこで高適合語転置索引を作成する際に，文書頻度の大きい索引語の出現位置情報を抽出することにすれば，成功率を大きく下げることなく，高適合語転置索引のサイズを小さく抑え，オーバーヘッドを削減できる可能性がある．

また，一般に検索語が長くなるほど，検索語の文書頻度が小さくなることから，検索語が少数の N グラムに分割される場合にのみ高適合語転置索引で検索するという方法も考えられる．これらの方法の効果の検証は今後の課題としたい．

表 5　検索語の文書頻度と成功頻度 (I_{16})

f_t	ALL	FAILURE1	FAILURE2	SUCCESS
0	33	17 (51.5%)	16 (48.5%)	0 (0%)
1-9	153	88 (57.5%)	65 (42.5%)	0 (0%)
10-49	156	98 (62.8%)	58 (37.2%)	0 (0%)
50-99	66	36 (54.5%)	29 (43.9%)	1 (1.5%)
100-499	195	103 (52.8%)	81 (41.5%)	11 (5.6%)
500-999	62	22 (35.5%)	16 (25.8%)	24 (38.7%)
1000-4999	143	16 (11.2%)	20 (14.0%)	107 (74.8%)
5000-	192	7 (3.6%)	2 (1.0%)	183 (95.3%)
total	1000	387 (38.7%)	287 (28.7%)	326 (32.6%)

5. 関連研究

転置索引を用いたランキング検索の効率化に関して，多くの研究がおこなわれている．

ディスクから読み出されるデータ量を削減し，検索を高速化するための一つの方法として，転置索引の圧縮がある．通常，出現位置情報は文書番号の昇順に並べられるため，番号の差分をとり，小さい数値ほど少ないビット数で表現するように符号化することで出現位置リストを圧縮できる[3]．圧縮率の高さよりも，検索速度の向上に主眼が置く場合には，CPU 処理コストの少ない単純な符号化方法が有効である[4]．

文書を特徴づける効果の大きい索引語のみを索引に

登録することで索引のサイズを抑える文書検索システムも多い．たとえば，あらかじめ用意されたストップワードを無視する方法や，Latent Semantic Indexing(LSI)を用いて索引語を選別する方法がこれにあたる[11]．しかし，こうした方法を用いると検索できない語句も多くあるため，Web サーチエンジンのように検索質問が短い場合や，N グラムを索引単位とする場合には採用しづらい．

本論文では単一の日本語フレーズの検索に限定して議論をおこなったが，先行研究の多くは複数の英単語からなる検索質問を対象としている．複数の検索語を検索する場合のランキング検索アルゴリズムは文書順と検索語順の二種類に大別される．

文書順では，複数の検索語の出現位置リストを並行に読み出しながら，文書番号の小さい文書から順に適合度を計算する．ヒープを用いて適合度の高い文書のみを保持することで，文書集合が大規模になっても，一定の量のメモリで正確な適合度を計算しながら検索できるという特長がある[12]．2.3節で述べたフレーズ検索の手順は文書順の一種である．

一方，検索語順では，検索語ごとに適合度の高い文書を求めていき，最後にそれらをマージすることで検索質問の適合文書として出力する．このとき，文書頻度の低い検索語から順に検索し，適合度への寄与の小さい検索語の検索は省略することで，ランキング検索の精度を大きく低下させることなく，適合度を保持するメモリの量を抑えることができる[13][14]．これをさらに効率化する方法として，出現位置情報を文書内頻度の降順にソートしておき，適合度への寄与が大きい出現位置情報だけを索引から読み出して検索する方法も提案されている[15]．

適合性フィードバックによる質問拡大をおこなう場合のように，検索語の数が大きいときには，検索語順の方が高速になる[16]．しかし，文書集合が大規模な場合には，検索語順で AND 条件・NOT 条件のブーリアン検索やフレーズの検索をおこなうのは難しい．

Brown らは文書順を採用した場合にも，あらかじめ適合度への寄与が大きい出現位置情報を転置索引のヘッダーページに格納しておくことで，検索結果候補となる文書数を減らし，ランキング検索を高速化できることを示した[17]．彼らのシステムは AND 検索やフレーズ検索にも対応しているが，フレーズ検索では通常通り，フレーズを構成する索引語の出現位置情報すべてを読み出して近接性を判定している．

Carmel らは，適合度への寄与の大きい出現位置情報のみを格納した転置索引を使ったランキング検索方法を提案している[18]．彼らは索引語ごとに異なる閾値を用いることで，ランキング検索結果の上位が，通常の索引を用いた場合とほとんど変わらないようにできることを証明している．Carmel らの方法は索引篩法における高適合語転置索引だけを使って検索するのと似ている．しかし，彼らが小規模な索引を用いて，英単語の OR 条件による検索をおこなうことを想定しているのに対し，本論文では大規模な索引を用いて日本語のフレーズを正確に検索することを想定している．

検索語が索引語と一致する場合は，あらかじめすべての索引語についてスコアの高い文書上位 k 個を求めておくことで，瞬時にランキング検索結果を求めることができる[2]．Web サーチエンジンでは複合語が検索されることが多いため，この方法が有効な検索質問は少ないと思われるが，索引篩法と併用することで，さらなる高速化を期待できる．

6. まとめと今後の課題

本論文では大規模な文書検索システムにおけるランキング検索を高速化する方法として索引篩法を提案した．実際の Web サーチエンジンのデータを用いた実験により，一部の検索語の検索速度が大幅に向上し，全体として 10%前後の高速化が達成されることを確認した．

なお，Web サーチエンジンでは文書の追加や削除が頻繁におこなわれるが，索引篩法は索引の更新にも対応できる．基本的に転置索引に出現位置情報が追加・削除されたときは，高適合語転置索引でも該当する出現位置情報を追加・削除すればよい．スコアの計算に文書集合のサイズを用いないので，文書数が増減しても高適合語転置索引全体を再構成する必要はない．

索引篩法には多くの点で改良の余地が残されている．

まず，失敗時に通常の転置索引で検索しなおすのは，同じ出現位置情報を二度読み出すことになり無駄が多い．たとえば，出現位置情報を高適合語転置索引と低適合度転置索引に分けて格納し，失敗時にはマージしながら検索するようにすれば，ディスクからの読み出しを減らすことができる．また，多段階に分割する方法や，出現位置情報をスコア順にソートして格納する方法も検討に値する．このような工夫によって失敗時のオーバーヘッドを小さくできれば，小規模な検索システムにも適用可能になる可能性がある．

本論文ではすべての索引語で一定の閾値 F を用いる方法を述べたが，索引語ごとに異なる閾値 F_t で出現位置情報を選別することもできる．この場合，検索語を構成する索引語の閾値の最大値を超えるスコアを得る文書が k 個以上検索されれば成功となる．しかし，閾値 F_t をどのように決めれば高い成功率が得られるのかは明らかではない．

また，スコアではなく文書内頻度によって出現位置情報を選別してもよい．この場合は文書内頻度が閾値

と等しいとき，最小サイズの文書がとり得るスコアの最大値を超える文書が k 個以上検索されれば成功とする．ただし，この方法では成功率が大きく低下する恐れがある．

最大の課題は，複数の検索語を含む検索質問への対応である．特に Web サーチエンジンの場合，2，3 の検索語を含む検索質問の AND 検索の高速化は重要である．部分的な出現位置情報による AND 検索で上位 k 件を正確に検索することは，英単語であっても容易ではなく，日本語フレーズではなおさら困難だと思われる．しかし，多少の精度の低下が許されるのであれば実現できる可能性はある．

なお，本論文では文書頻度が検索語のスコアの計算に必要ないことを利用したが，実用上は検索結果文書数の表示があったほうがよい．そこで擬似頻度法[6]のような方法によって文書頻度を推定することが望ましい．

文　献

[1] S. Brin and L. Page, “The Anatomy of Large-Scale Hypertextual Web Search Engine,” Computer Networks and ISDN Systems, vol.30, no.1-7, pp.107-117, 1998.

[2] 速水賢史,竹野浩,永瀬智哉,藤本典幸,萩原兼一, “スケーラビリティのある WWW 並列全文検索システム構築法の提案と評価”,情報処理学会データベース研究会研究報告, vol.123, no.7, pp.45-52,2001.

[3] I. H. Witten, A. Moffat, and T. C. Bell, Managing Gigabytes: Compressing and Indexing Documents and Images, Morgan Kaufmann, San Francisco, California, second edition, 1999.

[4] F. Scholer, H.E. Williams, J. Yiannis and J. Zobel, “Compression of Inverted Indexes For Fast Query Evaluation,” Proc. 25th ACM-SIGIR International Conference on Research and Development in Information Retrieval, pp.222-229, Tampere, Finland, 2002.

[5] 小川泰嗣, 松田透, “N-gram 索引を用いた効率的な文書検索法,” 信学論(D-I), vol.J82-D-I, no.1, pp.121-129, Jan. 1999.

[6] 小川泰嗣, “擬似頻度法：n-gram 索引のための高速な日本語文書のランキング検索法,” 信学論(D-I), vol.J83-D-I, no.10, pp.1043-1054, Oct. 2000.

[7] B. J. Jansen, A. Spink, and T. Saracevic, “Real Life, Real Users, and Real Needs: A Study and Analysis of User Queries on the Web,” Information Processing and Management, vol.36, no.2, pp.207-227, 2000.

[8] 原田昌紀,風間一洋,佐藤進也, “ユニコードを用いた N-gram 索引の一実現方式とその評価,” 研究会報告 NL136, 情報処理学会, Mar. 2000.

[9] 風間一洋,原田昌紀,佐藤進也, “ハイパーリンクとアンカーテキストを利用した情報検索とランキングの一手法,” 研究会報告 FI59/DD24, 情報処理学会, Jul. 2000.

[10] A. Singhal, C. Buckley, and M. Mitra, “Pivoted Document Length Normalization,” Proc. 19th ACM-SIGIR International Conference on Research and Development in Information Retrieval, pp.21-29, Zurich, Switzerland, Aug. 1996.

[11] 北研二,津田和彦,獅々掘正幹, 情報検索アルゴリズム, 共立出版, 東京, 2002.

[12] D.R.Cutting and J.O.Pedersen, “Space Optimizations for Total Ranking,” Proc. RIAO’97, Computer-Assisted Information Searching on Internet, Quebec, Canada, pp.401-412, Jun. 1997.

[13] M. Persin, “Document Filtering for Fast Ranking,” Proc. 17th ACM-SIGIR International Conference on Research and Development in Information Retrieval, pp.339-348, Dublin, Ireland, 1994.

[14] A. Moffat, J. Zobel, and R. Sacks-Davis, “Memory Efficient Ranking,” Information Processing and Management, vol.30, no.6, pp.733-744, 1994.

[15] M. Persin, J. Zobel, and R. Sacks-Davis, “Filtered Document Retrieval with Frequency-Sorted Indexes”, Journal of the American Society of Information Science, vol.47, no.10, pp. 749-764, 1996.

[16] M. Kaszkiel and J. Zobel, “Term-ordered Query Evaluation versus Document-ordered Query Evaluation for Large Document Databases,” Proc. 21st ACM-SIGIR International Conference on Research and Development in Information Retrieval, pp.343-344, Melbourne, Australia, Aug. 1998.

[17] E. Brown, “Fast Evaluation of Structured Queries for Information Retrieval,” Proc. 18th ACM-SIGIR International Conference on Research and Development in Information Retrieval, pp.30-38, Seattle, Washington, 1995.

[18] D. Carmel, D. Cohen, R. Fagin, E. Farchi, M. Herscovici, Y. Maarek, and A. Soffer, “Static Index Pruning for Information Retrieval Systems, ” Proc. of the 24th ACM SIGIR Conference on Research and Development in Information Retrieval, pp.43-50, 2001.