協調検索エンジンにおけるクエリー対象の最小化

上原 稔、山本 崇、佐藤 永欣、西田 喜裕、森 秀樹
東洋大学工学部情報工学科
{uehara,yama,jju,nishi,mori}@ds.cs.toyo.ac.jp

インターネットの発展に伴い、集中型の検索エンジンでは、文書の収集やインデックスの管理が困難となる可能性がある。そこで、我々は分散した局所的な検索エンジンが協調して検索を行う協調検索エンジン(CSE)を提案した。CSE では、局所的検索エンジン(LSE)が他の LSE にクエリーを送信し、結果をまとめて表示する。そのため、クエリーの送信対象を最小化することで一定の最適化を実現できる。本論文では、CSE のそのアルゴリズムについて述べる。

Minimizing Query Targets in CSE

Minoru Uehara, Takashi Yamamoto, Nobuyoshi Sato, Yoshihito Nishida, Hideki Mori
Dept. of Information and Computer Sciences, Toyo University

The population of Internet is explosively increasing now. The documents in it will be too many to be collected in future. Then, we proposed Cooperative Search Engine (CSE) in which distributed Local Search Engines (LSE) communicate with each other to find the information. In CSE, an LSE sends a query to other LSEs, receives their replies and show the result merged them. Therefore, we can optimize the number of queries by minimizing the number of query targets. In this paper, we describe an algorithm in CSE.

1. はじめに

インターネットの文書が現在の増加率のまま増加していくと集中型の検索サイトで必要な文書を検索するのは困難となる。そこで、我々は分散された局所的検索エンジンが協調して検索を行う協調検索エンジン(Cooperative Search Engine, CSE)を提案した[1,2,3,4]。

CSE には、局所的な文書を管理する複数の局所検索エンジン(Local Search Engine, LSE) と１つの位置サーバ(Location Server, LS)が存在する。ユーザからの要求を受けたLSEはLSに「どのLSEが知っているか」を問い合わせ、返されたLSEの集団に対して問い合わせ、その結果をまとめて表示する。

このような方式では、CSE に参加する LSE が増加するにつれ、(並列に検索したとしても) 通信遅延が問題となる。そこで、LS が返す LSE のサイト集合を最小化することが重要となる。本論文では、論理型検索においてサイト集合を最小化し、クエリーを最適化するアルゴリズムを提案する。

本論文の構成は以下の通りである。第２節では、既存の論理型検索について検討する。第３節では、

CSE における論理型検索の原理を述べる。第4節では、前節の原理に基づきアルゴリズムを提案し、検索例を示す。第5節では、CSE の検索効率について検討する。第6節では、関連研究について述べる。最後に結論を述べる。

2. 論理型検索

論理型検索あるいはブーリアン検索は、検索条件をブーリアン演算子により組み合わせた検索論理式を用いる検索方式である。基本的な検索論理式 E は、以下の通りである。

E: *keyword* | (E) | E AND E | E OR E | E NOT E

ここで、*keyword* は文中にそのキーワードを含む URL の集合である。また、NOT は二項演算とする。

次に検索の効率について考える。検索の効率は集合の要素数が大きくなるほど処理時間がかかり、また検索アルゴリズムによっても異なる。例えば、AND または NOT 検索では、全要素に対してキーワード A の全文検索が行われ、その結果にキーワード B の全文検索が行われる。OR 検索では全要素に対して A と B の両方で全文検索を行う。このようなアルゴリズムでは、検索論理式 E の処理時間 T(E)は以下の関係がある。

T(a)∝S

T(E AND a)∝T(E)+#E

T(E AND F)∝T(E)+T(F)+2#E+#F

T(E NOT a)∝T(E)+#E

T(E NOT F)∝T(E)+T(F)+2#E+#F

T(E OR F)∝T(E)+T(F)　　(逐次検索)

T(E OR F)∝max(T(E),T(F))　　(並列検索)

ここで、a はキーワード式、E、F は論理式とし、#a、#E はそれぞれに該当する集合の要素数、S は全要素数である。これより OR 検索は並列で行い、AND および NOT 検索では該当文書の少ない順にキーワードを左から右へ並べることが望ましい。

3. CSE における論理型検索

CSE における検索の特徴は、"To know who knows it is to know it."である。つまり、大量の知識を持つ代わりにそれらへの参照を持つ。この参照は、誰が情報源かを示す言明である。以下に CSE に関する用語を定義する。

定義1　URL 集合

サイト集合 S の検索論理式 E にマッチする文書の URL 集合を U(S,E)と表す。

定義より $U(S,E)=\cup_{s\in S}U(\{s\},E)$ は自明である。また、キーワード a, b と、それぞれを含む文書を持つサイト集合 S_a, S_b があるとき以下の関係が成り立つ。

$U(S_b,a)\subseteq U(S_a,a)$

$U(S_a-S_b,b)=\{\}$

$U(S_a-S_b,a\ AND\ b)=\{\}$

ここで、それぞれの関係を明確にするためベン図として表1を示す。

次に単純な論理型検索について考える。

定理1　単純 OR 検索

キーワード a, b を含む文書を持つサイト集合 S_a, $S_b\subseteq S$ あるとき、$U(S, a\ OR\ b)=U(S_a\cup S_b, a\ OR\ b)$ が成り立つ。

証明　自明。

表1　キーワード a, b における URL 集合

		S_a		
		S_b		
		Sa-Sb	Sa∩Sb	Sb-Sa
A		U(Sa-Sb, a)	U(Sa∩Sb, a) - U(Sa∩Sb, a AND b)	{}
	B	{}	U(Sa∩Sb, a AND b)	
			U(Sa∩Sb, b) - U(Sa∩Sb, a AND b)	U(Sb-Sa, b)

定理 2　単純 AND 検索

キーワード a, b を含む文書を持つサイト集合 S_a, $S_b \subseteq S$ があるとき、U(S, a AND b) = U($S_a \cap S_b$, a AND b)が成り立つ。

証明　表 1 より成立。

NOT 検索では、U($S_a \cap S_b$, a NOT b)≠{}であるため、U(S, a NOT b) = U(S_a-S_b, a NOT b)は成立しない。

定理 3　単純 NOT 検索

キーワード a, b を含む文書を持つサイト集合 S_a, $S_b \subseteq S$ があるとき、U(S, a NOT b) = U(S_a, a NOT b) が成り立つ。

証明　U(S, a NOT b) = U(S, a) - U(S, a AND b)で、U(S, a) = U(S_a, a)かつ U(S, a AND b) = U($S_a \cap S_b$, a AND b)より、U(S, a NOT b) = U(S_a, a) - U($S_a \cap S_b$, a AND b)。また、U(S_a, a NOT b) = U(S_a, a) - U(S_a, b)で、U(S_a, b)= U($S_a \cap S_b$, a AND b)であるから成立。

次に一般的な検索式について考える。以下の式が成立する。

U(S,A)=U(S_A,A)

U(S,A OR B) = U(S,A)∪U(S,B)

U(S,A AND B) = U(S,A)∩U(S,B)

U(S,A NOT B) = U(S,A) – U(S,B)

これらにより以下の定理が成立する。

定理 4　OR 検索

検索式 A, B に合致する文書を持つサイト集合 S_A, $S_B \subseteq S$ あるとき、U(S, A OR B)=U($S_A \cup S_B$, A OR B)が成り立つ。

証明　自明。

定理 5　AND 検索

検索式 A, B に合致する文書を持つサイト集合 S_A, $S_B \subseteq S$ があるとき、U(S, A AND B) = U($S_A \cap S_B$, A AND B)が成り立つ。

証明　U(S,A)=U(S_A,A)=U(S_A-$S_B \cup S_A \cap S_B$,A) = U(S_A-S_B,A)∪U($S_A \cap S_B$,A)と U(S,B) = U(S_B-S_A,B)∪U($S_A \cap S_B$,B)より、U(S,A AND B) = U(S,A)∩U(S,B)= U(S_A-S_B,A)∩U(S_B-S_A,B) ∪ U(S_A-S_B,A)∩U($S_A \cap S_B$,B) ∪ U(S_B-S_A,B)∩U($S_A \cap S_B$,A) ∪ U($S_A \cap S_B$,A)∩U($S_A \cap S_B$,B)= U($S_A \cap S_B$,A)∩U($S_A \cap S_B$,B) = U($S_A \cap S_B$, A AND B)が成立。

定理 6　NOT 検索

検索式 A, B に合致する文書を持つサイト集合 S_A, $S_B \subseteq S$ があるとき、U(S, A NOT B) = U(S_A, A NOT B) が成り立つ。

証明　U(S, A) = U(S_A, A)かつ U(S, A AND B) = U($S_A \cap S_B$, A AND B)より、U(S, A NOT B) = U(S, A) - U(S, B) = U(S, A)-U(S, A AND B) = U(S_A, A) - U($S_A \cap S_B$, A AND B)。また、U(S_A, A NOT B) = U(S_A, A) - U(S_A, A AND B)で、U(S_A, A AND B) = U($S_A \cap S_B$, A AND B)であるから成立。

定理 4, 5, 6 は CSE における論理型検索のアルゴリズムを与える。CSE では、サイト集合を検索式に基づいて限定さえすれば、各 LSE からの結果を特に後処理をせずに和すればよい。

4. CSE における論理型検索のアルゴリズム

ここで、CSE における論理型検索のアルゴリズムを示す。はじめに LSE のクライアントとしての動作を表す LSE Client を示す。

```
Algorithm LSE Client
Input E: Query
Output U: Set of URL
Variables
  LS: Location Server
  S: Set of Site
  function σ(E: Query): Query
  begin
    case E in
    x NOT y: return x
    x AND y: return σ(x) AND σ(y)
    x OR y: return σ(x) OR σ(y)
```

```
      otherwise: return E
      esac
    end
  begin
    Send σ(E) to LS.
    Receive S from LS.
    U = {}
    parallel for each s_i in S
      begin
        Send E to s_i.
        Receive U_i from s_i.
        U = U ∪U_i.
      end
    Return U.
  end.
```

ここで、parallel は並列処理を意味する。Query は検索式の構造体で、関数σは定理 6 に基づき検索式を、サイト集合を求める式に変換する関数である。

次に、LS の動作を示す。

```
  Algorithm LS
  Variables
    E: Query
    S: Set of Site
    function Know(E: Query): Set of Site
    begin
      case E in
      x AND y: return Know(x) AND Know(y)
      x OR y: return Know(x) OR Know(y)
      otherwise: return {s| s:Site knows url
                 matched E}
      esac
    end
  begin
    Receive E from s.
    S = Know(E).
    Send S to s.
    end.
```

ここで、関数 Know は検索式 E にマッチする URL を持つサイトの集合を返す関数である。

最後に LSE のサーバー側の動作として LSE Server を示す。

```
  Algorithm LSE Server
  Variables
    E: Query
    U: Set of URL
  begin
    Receive E from s.
    U = {d | d is document's url matched to E}.
    Send U to s.
  end.
```

表 2　例 1

Site	Document's URL
s_1	u_1="A B C", u_2="A", u_3="B", u_4="C", u_5="A B", u_6="B C", u_7="C A", u_8=""
s_2	u_9="A", u_{10}="B", u_{11}="A B", u_{12}=""
s_3	u_{13}="A", u_{14}="B", u_{15}="A C", u_{16}=""
s_4	u_{17}="B", u_{18}="C", u_{19}="B C", u_{20}=""
s_5	u_{21}="A", u_{22}=""
s_6	u_{23}="B", u_{24}=""
s_7	u_{25}="C", u_{26}=""

次に本アルゴリズムによる論理型検索の例を LSE Client を中心に示す。各サイトの URL は表 2 の通りとする。検索式として E = a NOT b AND c OR a AND b NOT c を与えたとき、以下のように進行する。

1. Send σ(E) to LS.
   σ(E)="a AND c OR a AND b"として LS に問い合わせる。
2. Receive S from LS.
   LS はσ(E)に基づき S={s_1, s_2, s_3}を返す。
3. For each s in {s_1, s_2, s_3}, Send E to s_i
   {s_1, s_2, s_3}のそれぞれに"A not B and C or A and B not C"を送信する。このとき可能であれば結果を得るまで非同期もしくは並列に送信してもよい。

4. Receive U_i from s_i. $U = U \cup U_i$.
   s_1 から $U_1=\{u_5, u_7\}$を受信し、s_2 から $U_2=\{u_{11}\}$ を受信し、s_3 から $U_3=\{u_{15}\}$を受信する。なお、並列実行している場合は、受信次第 U にマージする。
5. Return U.
   結果として$\{u_5,u_7,u_{11},u_{15}\}$を返す。

この結果はアルゴリズムの正しさを傍証するものである。

5. CSE の検索コスト

ここで、CSE の検索コストについて考察する。CSE では、検索に要する時間として以下の要素が考えられる。

(1) LS との通信時間
   ある LSE が LS へ問い合わせ、LS がサイト集合を計算し、それを返すまでの時間である。
(2) 各 LSE との通信時間
   ある LSE が(1)のサイト集合に含まれる全 LSE に対して、検索を依頼し、その LSE が検索を行った結果を受信して、すでに受信していた URL 集合とマージするまでの時間である。
(3) URL 集合を表示する時間
   URL 集合をスコア順に整列し、表示するまでの時間である。

このうち(1)は LS を分散配置したりキャッシュを用いたりして応答を高速化することはできるが、本質的に必要な処理である。しかし、(2)は本文で述べたように並列検索時のサイト集合 $\sigma(E)$を絞り込むことで縮小できる。また、(3)は検索結果に依存する。そこで、ここではサイト集合の絞込みの有効性をシミュレーションで検証する。

シミュレーションは N 個のキーワードと N-1 個の演算子で構成されるすべての検索式を生成し、その検索式で求められるサイト集合が全サイトに占める割合を平均として求めた。また、データは N 個のキーワードのすべての組み合わせをサイトごとに分散した。この結果を表 3 に示す。実際の検索ではキーワードや文書の偏りにより異なるものの約 40%の高速化が期待できる。ただし、インデックスの最適配置などの工夫が必要である。ここでいう高速化とは、並列検索の前後で行われるプロセスの fork や結果のマージなどの逐次処理に要する時間の削減である。

また、2 節で述べたように、クエリー中のキーワードは、検索式の左側から、AND 検索の具体的なもの（文字数の長いもの）、AND 検索の抽象的なもの（文字数の短いもの）、NOT 検索の抽象的なもの、NOT 検索の具体的なものの順で配すとさらに検索時間を縮小できる可能性がある。

表 3　N 個のキーワードを含む式の評価

N	割合[%]
2	60
3	54.7
4	57.5
5	61.0
6	64.5
平均	59.4

6. 関連研究

複数の検索エンジンを用いた検索方式にはメタ検索エンジンがある。メタ検索エンジンでは、基本となる検索エンジン同士が協調していないため、資源の節約とならない。また、スコアに関して統一された評価がないため、結果のマージが困難である。

分散型検索エンジンは自らも検索エンジンである複数サイトの協調で実現される。メタ検索エンジンの機能に加え、インデックス管理が問題となる。分散型検索エンジンとしては、Infoseek Distributed Search[9]や CHIC[7]などがある。[9]ではスコアリングの問題を解消しているが、オープンな協調を目指すものではない。[7]では標準プロトコルを採用している。中でも WHOIS++[10]

の forward knowledge は LS に似た役割を持つ。他には、HTTP に SEARCH メソッドを取り込む方式[8]も提案されている。これらの方式はいずれも多くの資源を必要とするため組織内の文書管理として採用するには導入コストが大きい。また、粗い分散化を前提にしているため、CSE のようにサイト集合を限定するものは少ない。

論理型検索は多くの検索サイトで実現されている。しかし、それぞれ論理式の機能や表現が異なる。例えば、AltaVista は NOT を単項演算として扱う。また、CSE では Namazu や SGSE をサブシステムとして用いるが、それぞれの検索式は仕様が異なる。統一的な仕様としては Z39.50[5]や STARTS[6]などが提案されている。現在の CSE は独自プロトコル CSP/GMTP[3]を用いているが、今後はこれらのプロトコルをサポートし、相互運用の可能性を検証していく必要がある。

7. まとめ

本文では、CSE における論理型検索のアルゴリズムと最適化の手法について述べた。CSE では、「誰が情報を知っているか」を知識として持つことでクエリーの送信先を限定し高速な分散検索を可能とする。また、並列検索可能で、結果のマージも効率がよい。今後は、共通プロトコルをサポートし、大規模な運用を可能とする。

参考文献

[1] 佐藤永欣、山本崇、西田喜裕、上原稔、森秀樹: "協調サーチエンジンの研究", 第 45 回東洋大学工業技術研究所講演会(1999.3)

[2] 山本崇、佐藤永欣、西田喜裕、上原稔、森秀樹: "協調検索エンジンの研究", DICOMO'99, p169-174(1999.6)

[3] 西田喜裕、山本崇、佐藤永欣、上原稔、森秀樹: "分散サーチエンジンにおける協調型検索", SWoPP'99

[4] 佐藤永欣、山本崇、西田喜裕、上原稔、森秀樹: "協調検索エンジンにおけるスコアリング", 情報処理学会第 59 回全国大会

[5] "The Z39.50 Maintenance Agency", http://lcweb.loc.gov/z3950/agency/

[6] Luis Gravano, Kevin Chang, Hector Garcia-Molina, Carl Lagoze, Andreas Paepcke: "Stanford Protocol Proposal for Internet Search and Retrieval", http://www-db.stanford.edu/ ~gravano/starts.html

[7] Peter Valkenburg, Dave Beckett, Martin Hamilton, Simon Wilkinson: "Standards in the CHIC-Pilot Distributed Indexing Architecture", http://www.terena.nl/projects/chic-pilot/tnc/paper.html

[8] Martin Hamilton: "Experimental HTTP methods to support indexing and searching", draft-hamilton-indexing-00.txt

[9] "Infoseek Distributed Search Patent", http://software.infoseek.com/patents/dist_search/

[10] C. Weider, J. Fullton, S. Spero: "Architecture of the Whois++ Index Service", RFC1913, ftp://ftp.ripe.net/rfc/rfc1913.txt