協調検索エンジンにおける分散型スコアリング

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

現在多くの集中型検索エンジンではスコアリングのために tf*idf 法を採用している。しかし、分散型検索エンジンでは tf*.idf 法に基づくスコアを局所的に計算することが困難である。そこで、分散 tf*idf 法が提案されている。本論文では、我々が提案する分散型検索エンジン CSE(Cooperative Search Engine)における分散 tf*idf 法の実現について述べる。

Distributed Scoring in Cooperative Search Engine

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

Current almost centralized search engines employ the tf*idf method for scoring. However, in distributed search engines, it is difficult to compute tf*idf based scores in local. The distributed tf*idf method was proposed in order to solve this problem. In this paper, we describe to realize distributed tf*idf method in CSE(Cooperative Search Engine), which is proposed by us.

1. はじめに

インターネットの爆発的な普及により目的の情報を迅速に入手する手段として検索サイトが利用されている。現在の多くの検索サイトは、単一サイトで文書を収集し、インデックス情報を抽出し、要求に応じて検索結果を返す集中型のシステムである。しかし、このような集中型では増加する文書量に対してインデックス情報の増加が問題となる。

また、一方でフリーな日本語全文検索エンジンの登場により大学や企業などの組織や個人の Web ページ検索が可能となった。しかし、独立した検索エンジンはインターフェースも千差万別で、複数の情報源を利用するにはメタ検索エンジンを利用する必要がある。

そこで、分散型の検索エンジンまたはインデックスシステムがいくつか提案されている[9][11][12][13][14][15][16]。しかし、これらのシステムは必ずしも普及していない。その理由として以下の原因が考えられる。

A) 性能面で集中型に及ばない
現在のインターネットでは、文書量やネットワークの転送速度などから集中型で十分であることが多い。しかし、ネットワーク技術の進歩により将来この前提が変わる可能性がある。

B) 導入コストが高い
多くのシステムは導入に際して多くの資源や管理者権限が必要である。また、導入後に拡張が困難である場合もある。

C) スタンドアロン検索システムとの非互換性

既にスタンドアロンの全文検索システムを導入している場合、新たに検索システムを導入するメリットが少ない。また、既存の検索システムを活用できる必要がある。

D) 大規模な検証が行われていない
分散型検索エンジンは普及しないと大規模な検証が困難であるが、検証ができないと普及も進まない。

E) 日本語が扱えないこと
海外の多くのシステムでは日本語の全文検索機能が困難ないし不充分である。

このうち A)はインターネットの規模の拡大と性能向上に依存する。D)は規模の増大に対して性能を維持できるかを検証する上で重要な問題であるが、現在は中規模の実験か理論的考察に頼らざるを得ない。E)は最近の日本語全文検索エンジンの進歩により解消されつつある。残る B)、C)は各システムに依存した問題である。

我々は、組織内の Web ページ検索を対象とした分散型検索エンジンとして CSE(Cooperative Search Engine)を開発した[1][2][3][4][5][6]。CSE は、少なくとも１つのサーバ LS(Location Server)と CGI で実現された局所的検索エンジン LSE(Local Search Engine)で構成される。LSE は広く普及している Namazu[7]や SGSE[8]などの全文検索エンジンを CSE で利用するためのメタエンジンである。このため CSE は導入が比較的容易であり、既存システムとの互換性を持つ。

分散型検索エンジンの問題点として、分散された個々の局所的検索エンジンは検索式の表現やスコアなどの共通知識を維持する必要がある。本文では、共通知識の中でも主としてスコアを取り上げ、CSE における分散スコアリングについて述べる。

本文の構成は以下の通りである。第２節では、関連研究として分散型検索エンジンにおけるスコアリング方式をまとめる。第３節では、CSE の概要を要約する。第４節では、CSE における分散スコアリングのアルゴリズムを提案する。第５節では、実装方法について述べる。第６節では、評価について述べ、最後に結論を述べる。

2. 関連研究

Web ページを対象とした分散型検索エンジンに関する研究はいくつか行われている。Infoseek[13]では、sitelist.txt を用いた効率的な文書収集法を提案している。また、国内でもロボットによる分散収集の試みも行われている[16]。これらは主として文書収集の分散化である。一方、検索自体を分散システムで行うものには[9][11][12][14][15]などがある。これらは基本的に forward knowledge に基づくものである。CSE も同様の原理に基づいている。

多くの検索エンジンでは、スコアリングに tf*idf 法を採用している。Tf*idf 法では、単にキーワードの文中における出現頻度が高いだけではなく、キーワードの重要さを考慮する。すなわち、どの文書にも多く含まれる語はスコアを低く押さえ、珍しい語のスコアを高くする。

Tf*idf 法におけるスコア w は以下の式で与えられる。

$$w = K \cdot tf \cdot idf$$

$$idf = \log(N/n)$$

ここで、K は定数、tf は文中の語数、n は語を含む文書数、N は全文書数である。

しかし、分散型検索エンジンでは n および N を局所的に決定することはできない。そこで、サイト i の $n=\Sigma n_i$、$N=\Sigma N_i$ を用いてスコアを求める方法が分散 tf*idf 法である。分散 tf*idf 法ではスコアを以下の式で与える。

$$w = K \cdot tf \cdot idf'$$

$$idf' = \log(\Sigma N_i / \Sigma n_i)$$

ここで、K は定数、tf は文中の語数、n_i はサイト i の語を含む文書数、サイト i の N_i は全文書数である。

分散 tf*idf 法を採用しているシステムには

WHERE[9]、Infoseek[13]などがある。

分散 tf*idf 法に基づいて n_i および N_i を収集する方法には２相方式[10]と１相方式がある。２相方式では、まず各サイトに検索要求を送信して各サイトから n_i および N_i を得て、その結果を再び各サイトへ送り、各サイトは検索結果とスコアを正しく計算して返す。しかし、２相方式では通信遅延が増加してしまうという問題がある。それに対して、１相方式では、n_i および N_i をあらかじめ収集しておき、検索要求時にそれらをサイトへ送り、各サイトは検索結果とスコアを正しく計算して返す。このため、１相方式では検索時の通信遅延を低減することができる。

3. CSE

ここでは、後続の節の議論を円滑にするため CSE の概要について述べる。

CSE には、局所的検索エンジン LSE(Local Search Engine)と、それらの位置(URL)を認識する位置サーバ LS(Location Server)がある。LSE は、原則として各 Web サーバーに配され、そのサイトの全文検索エンジンを利用するメタエンジンである。LS は、LSE の位置を把握し、与えられた検索式に返答できる文書を持つ LSE の集合を返す。

CSE の動作は以下の通りである（Fig.1 参照）。

1. ユーザは LSE に検索要求を送る。
2. LSE はユーザの検索要求を受け、LS に問い合わせる。
3. LS は検索条件を満たす文書を持つサイト集合を求める。
4. LS はそのサイト集合を返す。
5. LSE はサイト集合に含まれる各 LSE に検索要求を送信する
6. 各 LSE はメタエンジンとして全文検索エンジンを起動する。
7. 検索結果を検索要求の送信元へ返送する。
8. LSE は検索結果を取りまとめ、表示する。

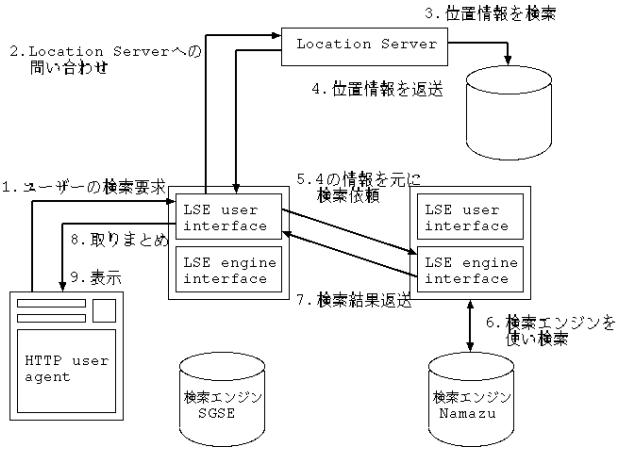

Fig.1　CSE の動作

CSE は forward knowledge を用いて分散した知識をまとめる。同様の WHOIS++や WHERE などの Query Routing では forward する度に遅延が増加する。これらの方式では、forward の深さ d に比例した遅延が生じる。しかし、CSE では LS が返すサイトに、クライアントが直接並列検索を行うため原理的な遅延は O(2)と少ない。ただし、結果のマージにはサイト数 N に応じて O(log N)の時間を要する。

4. CSE における分散スコアリング

CSE では、検索時の実行効率に優れる１相方式を分散 tf*idf 法として採用する。

LS は Update プロトコル[3]によりサイト s_i の LSE から以下の情報を収集する。

N_i

*(key tf_{max} tf_{min} n_i)

ここで、N_i はサイト s_i の全文書数、key はキーワード、tf_{max} および tf_{min} はサイト s_i のキーワード key に関する tf の最大値と最小値、n_i は語を含む文書数である。LS は効率良く検索できるようにこれらを key ごとにまとめて整理・保存する。

Namazu や SGSE などの検索エンジンでは、検索論理式のスコアは単純なキーワードのスコアの組み合わせで計算される。例えば、Namazu では文書 d に対する検索式 E のスコア w(d,E)は以下の

式で計算される。

w(d,k) = tf(d,k) * idf(k)

w(d,A and B) = min(w(d,A), w(d,B))

w(d,A or B) = max(w(d,A), w(d,B))

w(d,A not B) = w(d,A)

ここで、tf(d,k)は文書 d におけるキーワード k の出現頻度を表す。ただし、後述のタグによる重み付けが行われているものと仮定する。また、idf(k)はキーワード k の idf、A および B は検索論理式を表す。また、SGSE では以下の式で計算される。

w(d,k) = tf(d,k)

w(d,A and B) = w(d,A) * w(d,B)

w(d,A or B) = w(d,A) + w(d,B)

w(d,A not B) = w(d,A)

SGSE は tf*idf ではないため、検索の質が低い。また、SGSE の計算法では AND 式のスコアが高くなる傾向がある。CSE では、Namazu の計算法を採用している。

ここで、あるサイトが持つ全文書の検索論理式 E に関する tf の値の領域を range(E)で表すことにする。このとき、k はキーワード、A および B は検索論理式とすると以下の式が成り立つ。

range(k) = [tf_{min}(k), tf_{max}(k)]

range(A and B) = [

min(range(A).tf_{min}, range(B).tf_{min}),

min(range(A).tf_{max}, range(B).tf_{max})]

range(A or B) = [

max(range(A).tf_{min}, range(B).tf_{min}),

max(range(A).tf_{max}, range(B).tf_{max})]

range(A not B) = range(A)

ここで、w(d,E)の算出法を Namazu と同様とし、tf_{min}(k)はそのサイトの全文書中の最小 tf 値とし、tf_{max}(k)は最大 tf 値である。なお、range(E).tf_{min} および range(E).tf_{max} はそれぞれ最小値、最大値である。また、range(E)は可能性の範囲を表すもので、上下限に該当する文書が存在するとは限らない。

サイト s_i が検索式 E に該当する文書を持つとき、そのスコアは $range_i$(E)で限定される。例えば、s_i は E に対する tf の範囲が $range_i$(E)に含まれる。

分散 tf*idf 法を用いた場合の検索アルゴリズムは以下の通りである。

1. クライアント(LSE_{init})は、LS に検索式 E と必要な文書数を順位の範囲[n_{min},n_{max}]をパラメータとして検索要求を送信する。すなわち、スコアの降順による順位で n_{min} 番から n_{max} 番までの候補を要求する。
2. LS はサイト集合 S を求め、その各要素 s_i∈S を $range_i$(E)の tf_{max} の降順にソートする。さらに、その結果を LSE_{init} に送信する。このとき以下の情報が返される。

   N

   *(k idf_k)

   *(s_i $tf_{max,i}$ $tf_{min,i}$ n_i)

   ここで s_i はサイトの LSE を示す URL で、$range_i$(E)=[$tf_{max,i}$,$tf_{min,i}$]、idf_k は E に含まれるすべてのキーワード k の idf、N は全文書数である。一般的に N には S に含まれないサイトの文書数も含まれるため $N \geqq \Sigma n_i$ である。
3. LSE_{init} は、U を空集合とし、tf_{max} および tf_{min} を初期化する。
4. LSE_{init} は、各 LSE s_i∈S に以下のパラメータを伴う検索要求を送信する。

   E

   n_{min} n_{max}

   *(k idf_k)

   このとき、4～6 までは並列に実行され、$|U| > n_{max}$ かつ $tf_{min} > tf_{max,i}$ なる s_i への送信は中止される。
5. LSE s_i は該当する文書の URL 集合 U_i を求め、その各要素 $u_j \in U_i (j=1..n_{max})$ を tf の降順にソートする。その結果を LSE_{init} に送信する。このとき以下の情報が返される。

   n_i

*n_{max} (u_j w_j)

ここで n_i は s_i 中の該当全文書数、u_j は文書の URL、w_j の計算を前述の Namazu 式 w(d,E)に基づき行う。その際、idf_k を用いる。また、n_{max} が無指定の場合、すべての該当文書を返す。

6. LSE_{init} は、各 LSE $s \in S$ の結果 U_i を tf_{max} および tf_{min} を更新しながら U にマージソートする。LSE_{init} は、U から$[n_{min},n_{max}]$の範囲の要素を表示する。

このアルゴリズムでは、$[n_{min},n_{max}]$の範囲の文書を収集するのに$[1,n_{max}]$の文書を収集する必要がある。実際に収集するまでは本当の順位がわからないため、無駄がある。将来の実装ではキャッシュ機能を搭載することで効率の改善を試みる。

ここで、検索の例を示す。４つのサイト s_1, s_2, s_3, s_4 があり、それぞれ検索式 E=k に合致する URL 集合$\{u_{11},u_{12}\}$, $\{u_{21}, u_{22}\}$, $\{u_{31}, u_{32}, u_{33}, u_{34}\}$, $\{u_{41}, u_{42}\}$を持つとき、n_1=2、n_2=2、n_3=4、n_4=2、N_1=8、N_2=8、N_3=16、N_4=32 とし、各 u_{ij} の tf は Table 1 の通りであったとする。

Table 1. スコアの例

URL	tf	tf.idf	分散 tf.idf
u_{11}	8	4.8	6.4
u_{12}	3	1.8	2.4
u_{21}	10	6.0	8.1
u_{22}	5	3.0	4.0
u_{31}	7	4.2	5.6
u_{32}	6	3.6	4.8
u_{33}	4	2.4	3.2
u_{34}	3	1.8	2.4
u_{41}	2	2.4	1.6
u_{42}	1	1.2	0.8

ここで、従来の tf*idf では tf に比例したスコアとならないが、分散 tf*idf では tf に比例したスコアとなることが確認できる。

検索は以下のように行われる。

1. LS に検索要求を送信する。
2. LS は以下の情報を返す。

   k log(64/10)

   s_2 10 5 2

   s_1 8 3 2

   s_3 7 3 4

   s_4 2 1 2
3. U = { }とする。
4. s_2, s_1, s_3 に以下を送信する。

   E

   3 5

   k log(64/10)
5. s_2 は以下の情報を返す。

   2

   u_{21} 8.1

   u_{22} 4.0
6. U=$\{u_{21},u_{22}\}$とし tf_{max}=10、tf_{min}=5 とする。
7. s_1 は以下の情報を返す。

   2

   u_{11} 6.4

   u_{12} 2.4
8. U=$\{u_{21},u_{11},u_{22},u_{12}\}$とし tf_{max}=10、tf_{min}=3 とする。
9. s_3 は以下の情報を返す。

   4

   u_{31} 5.6

   u_{32} 4.8

   u_{33} 3.2

   u_{34} 2.4
10. U = $\{u_{21}, u_{11}, u_{31}, u_{32}, u_{22}, u_{33}, u_{12}, u_{34}\}$ とし tf_{max}=10、tf_{min}=3 とする。
11. s_4 には送信しない。
12. u_{31}, u_{32}, u_{22} を表示する。

5. 分散スコアリングの実装

CSE では局所的検索エンジンとして Namazu と SGSE をサポートしている。Namazu は tf*idf 法を採用しているが、分散スコアリングには対応していない。SGSE は tf のみの単純なスコアを採用し

ている。本節では、Namazu および SGSE を用いて分散スコアリングを実現する方式について述べる。

5.1 タグの重み付けの共通化

Namazu および SGSE では tf を計算する場合、タグによる重み付けが行われている（Table 2, 3 参照)。

Table 2. Namazu におけるタグの重み付け

重み	タグの種類
32	Keywords, Description
16	TITLE
8	H1
7	H2
6	H3
5	H4
4	H5
3	H6
2	STRONG, EM, KBD, SAMP, VAR, CODE, CITE, ABBR, ACRONYM, DFN
1	その他

Table 3. SGSE によるタグの重み付け

重み	タグの種類
100	Keywords
10	Description
2	TITLE
1	その他

スコアを共通化するにはこの重みを共通にしなければならない。どちらの検索エンジンでも重みはインデックス作成時に tf に含めて計算する。そこで、我々は CSE のためのインデックスを生成するプログラムを開発した。この手法では、インデックスの再作成を必要とするが、CSE 独自の重み付けを与えることができる。ただし、現在は Namazu の基準をそのまま採用しているため Namazu を採用しているサイトでは特に変更は行う必要はない。SGSE を採用しているサイトではインデックスの再構築が必要となる。

5.2 メタエンジンによるスコアの計算

スコアの分散計算はメタエンジンである LSE が Namazu や SGSE を呼び出して行う。このとき Namazu や SGSE の検索プログラムには一切変更を加えない。このためこれら既存の検索エンジンと両立が可能となる。

LSE で論理型検索を実現するには、Namazu や SGSE の論理型検索を直接使うことなく、単独のキーワード検索を繰り返し LSE で論理型検索をエミュレーションする。これにより論理型検索をサポートしていない検索エンジンでも分散 tf*idf 法を実現できる。しかし、性能面では劣る。将来、キーワードの idf を外部から与えることができる検索エンジンが登場すれば効率を改善できる。

次に、分散 tf*idf で必要とされるパラメータを算出する方法を Namazu と SGSE のそれぞれについて述べる。まず、Namazu の場合について述べる。

Namazu に依存するのは tf、n_i、N_i の各要素である。5.1 節で述べたように tf はインデクサーによって統一されている。検索時にその値を取り出すには namazu.conf でスコアの設定を SIMPLE とする。次に、n_i は NMZ.i 中に各キーワードに続き記録されているバイナリデータ(n_i*2)を用いる。また、N_i は NMZ.total より取得する。

次に SGSE の場合について述べる。SGSE でも tf、n_i、N_i の各要素である。Tf の算出法は Namazu と等しい。次に、n_i は HTML.inv で対応するキーワードのエントリに登録されている URL 番号とスコアのペアの数を数えることで得られる。N_i は HTML.idx のファイルサイズを 4 で割ることで得られる。

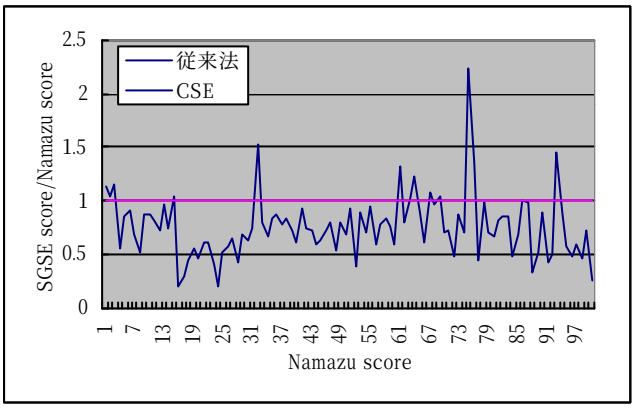

Fig.2　Namazu と SGSE のスコアの比

Table 4. CSE の性能比較(1)

	応答時間（秒）
CSE	1.267
Namazu	0.046
SGSE	0.448

Table 5. CSE の性能比較(2)

		応答時間（秒）
And 検索	CSE	1.5
	Namazu	5.6
Or 検索	CSE	5.0
	Namazu	5.6

6. 評価

分散スコアリングの必要性を検証するために Namazu と SGSE のスコアの差を比較する（Fig.2 参照）。ここでは、http://www.toyonet.toyo.ac.jp/ 以下の文書をサンプルとして、Namazu と SGSE で個別にインデックスを作成し、共通の全キーワードごとに各文書のスコアを求め、その比を積算した。なお、図中の「従来法」ではオリジナルの Namazu と SGSE を用い、「CSE」では本論文で述べた変更を施した。この結果より、従来法ではスコアのばらつきが 0.2〜2 の間に散らばっていることがわかる。これは極端な差ではないものの、適切な候補を選択する基準としてスコアを利用することが困難であることを示している。それに対して CSE の方式では Namazu と SGSE のどちらを用いても同じ結果となることが確認された。

また、CSE の性能を評価するために、キーワード検索の平均応答時間を Table 4,5 に示す。Table 4 は CSE と Namazu および SGSE を単一サイトにて単一キーワードで検索した結果である。この結果から CSE は Namazu や SGSE などの検索エンジンに比べて約４倍の遅延があるものの十分実用的な範囲であることがわかった。遅延の原因は、CSE がこれらの検索エンジンを内部的に呼び出していることと通信の遅延によるものである。また、Table 5 は、出現頻度 100〜120 のキーワード 10 組を無作為に抽出し、24 サイトに対して CSE と Namazu それぞれの And 検索、Or 検索を行った結果である。ただし、CSE は 24 サイトに並列検索を行うのに対し、Namazu は手動で 24 サイトを検索する場合を想定し、逐次検索を行った。結果として CSE の並列検索が有効に働いていることがわかった。特に、対象を絞り込める And 検索では、同じ並列検索でも Or 検索より有効である。なお、Namazu の遅延の原因はネットワークによると考えられる。このように、既存検索エンジンを通常の利用法で逐次的に用いるなら、CSE で一度に並列検索した方が性能的にもスコアの質的にも優れていることと言える。

7. まとめ

本論文では CSE における分散スコアリングについて述べた。CSE では、メタ検索エンジンの働きにより Namazu や SGSE といった検索エンジンを変更することなく分散 tf*idf 法に基づくスコア計算を実現できる。実装では Namazu のスコアリング方式を基に統一したが、本方式の原理は Namazu に依存していない。CSE は比較的小規模な組織内の孤立した全文検索エンジンを統合し、あたかも組織全体に渡る一つの大きな検索エンジンがあるかのようにみせることができる。

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(1999.8)

[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(1999.9)

[5] 上原稔、山本崇、佐藤永欣、西田喜裕、森秀樹: “協調検索エンジンにおけるクエリー対象の最小化”, DPSWS'99 (1999.12)

[6] 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における tf*idf 法に基づく分散スコアリング”, DPSWS'99 (1999.12)

[7] 馬場肇: “日本語全文検索システムの構築と活用”, ソフトバンク, ISBN4-7973-0691-2 (1998.9)

[8] “SonyDrive Search Engine”, http://www.sony.co.jp/sd/Search/

[9] Miguel Rio, Joaquim Macedo, Vasco Freitas: “A Distributed Weighted Centroid-based Indexing System”, In Proceedings of the 8th European Networking Conference (JENC8), 1997, http://www.terena.nl/conf/jenc8/papers/322.ps.

[10] 國頭吾郎、相澤清晴: “モバイルエージェントを用いた WWW 検索システムの協調に関する検討”, DICOMO'99, pp145-150,(1999.6)

[11] 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

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

[13] Steve Kirsch: "Infoseek's approach to distributed search", Report of the Distributed Indexing/Searching Workshop, http://www.w3.org/Search/9605-Indexing-Workshop/Papers/Kirsch@Infoseek.html

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

[15] "Ingrid: 広域情報検索システム", http://www.ingrid.org/index-j.html

[16] “次世代分散型情報検索システムに関する調査研究報告書”, http://www.jeida.or.jp/committee/jisedai/top.html