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

佐藤永欣† 山本 崇† 西田 喜裕† 上原 稔† 森 秀樹†

†東洋大学工学部情報工学科

Webで検索を行うサーチエンジンを構築する場合、集中型サーチエンジンでは大量の文書の収集やインデックスの管理が問題になる。そこで、分散したサーチエンジンが協調して検索を行う協調サーチエンジン(Cooperative Search Engine,CSE)を提案した。あるサーチエンジンに情報が無くても、情報を持っている他のサーチエンジンを教えてもらうことで間接的に検索できる。CSEでは分散した多数のサーチエンジンや、何種類かのサーチエンジンを利用可能だが、正しい検索を行うためには検索時のスコアの扱いが重要になる。本稿ではCSEでのスコアの扱いについて述べる。

Tf·idf based Distributed Scoring in Cooperative Search Engine

Nobuyoshi Sato†, Takashi Yamamoto†, Yoshihito Nishida† Minoru Uehara†, Hideki Mori†

†Department of Information and Computer Sciences, Toyo University

A centeralized search engine has a few ploblems on collecting a large number of documents and managing index information. So, we suggest the system in which more than one distributed search engines cooperate. We call this system Cooperative Search Engine(CSE). Even if a search engine does not have index information, it can get the information by asking other search engines which has the infomation. In CSE, some kinds of distributed search engines are used, Therefore it is important to deal with different types of scores in order of correct search. In this paper, we describe how to deal with distributed scoring in CSE.

1 はじめに

協調サーチエンジン(Cooperative Search Engine, CSE)[8][9]は、既存の集中型検索エンジンの問題点を解決すべく提案された、一部の専門知識を有する複数の局所サーチエンジン(Local Search Engine, LSE)が互いに協調することで検索を行う分散型協調サーチエンジンである。通常のサーチエンジンでは、あるドメイン内の全ての文書を対象とした検索を1台のマシンで行う。このため、サーチエンジンとなるマシンではドメイン内の全ての文書に関するデータベースを作成し、所有することになる。このデータベースをインデックスと呼ぶが、これを1台のマシンに所有させるのは、インデックス作成用の文書の収集、インデックスの保管、検索時の負荷の面からも大きな負担となる。協調サーチエンジンでは、小規模なサーチエンジンが協調して動作するため、これらの問題から開放される。

CSEではこれらのサーチエンジンにNamazu[6]とSGSE[7]の2種類のサーチエンジンを用いることができるが、この2つのサーチエンジンではスコアの計算方法が違うので、何らかの対策が必要である。また、tf·idf法を使うためにも、スコア、全文書数、ヒットした文書数などを的確に把握しなければならない。この論文ではNamazuとSGSEのスコアの特性の違いと調整、分散tf·idf法、検索するサーチエンジンを絞り込む方法について述べる。

本論文の構成は以下の通りである。第2節では分散したサーチエンジンおよびtf·idf法についての関連研究、第3節ではCSEの概要と動作について述べる。第4節ではCSEにおいて使用されるプロトコルと分散tf·idf法と検索対象LSEの絞り込みのための拡張、第5節ではCSEにおける2種類のサーチエンジンのスコアの調整と分散tf·idf法と検索対象LSEの絞り込みについて述べる。最後に課題と結論を述べる。

2 関連研究

2.1 分散したサーチエンジン

サーチエンジンの構成は収集部分と検索部分の2つに分けることができる。このうち収集部分の分散化はJEIDA[3]等で行なわれている。この他に、収集部分に対する工夫として、商用サーチエンジンのInfoseek

で用いられている情報収集の技術がある。従来の商用サーチエンジンでは、ロボットと呼ばれるプログラムを動作させ、検索に必要なデータを収集するのが一般的な手法であったが、膨大に増加し、更新され続けるWWW ページに対応するのが困難になって来たために、各 WWW サイトで Infoseek が用意したスクリプトを動作させ、それぞれの WWW サイトから自発的に検索のデータを送信させるという技術を開発した。これにより、専門的にデータを集めたり、検索のためのデータベースの更新間隔を早めたり、データのサイズを必要に応じて小さくすることができる。

検索部分の分散化は、サーチエンジンとなるマシンを複数用意して DNS のラウンドロビンを用いて負荷を分散している商用サーチエンジンの例がある。データベースも内部的に分散化していると思われるが、詳細をうかがい知ることはできない。

検索システム自体の分散化という観点からは、インターネット上に分散している情報を自律的に統合する機能をもった情報検索システムに NTT の "Ingrid"[4] が挙げられる。これは Ingrid サーバが Web サーバ内の情報のキーワードの索引付けと管理を自動的に行ない、他の Ingrid サーバと通信を行なって、類似した内容を持つ情報間でリンクをはりネットワークを形成し、検索できるようにしている。また、このネットワークの拡張は新たに Ingrid サーバを追加するだけですむという高い拡張性も備えている。

本研究で用いているような分散 tf·idf 法では、WHERE[5] がある。WHERE は CSE のように 2 段構成の検索ではなく、多数のコンポーネント間でユーザーの検索要求を転送して検索を行う全文検索システムである。また、WHERE では Whois++の検索サービスを利用することもできる。

3 CSE 構成と動作の概要

3.1 CSE の構成

CSE には二つの構成要素がある。一つは文書の検索、ユーザーとの対話を行う Local Search Engine(LSE)、もう一つは与えられたキーワードについて詳しい情報を持っている LSE を捜す Location Server(LS) である。CSE ではこれらの構成要素が協調して動作することにより分散した文書の検索を可能にしている。LSE はいくつ存在してもよく、通常は Web サーバー一つについて LSE が一つ存在する。しかし、複数の LSE が一つの Web サーバに同居したり、他の Web サーバの情報を持つことも可能である。これらの LSE は LSE 同士、または Location Server と通信しながら検索を行う。CSE のこのような動作モデルは以下のようなメリットを持つ。

- 集中型サーチエンジンではインデックスの更新を行う毎に各 Web サーバが持つ文書を転送する必要があるが、CSE では Location Server にキーワードとスコアの情報を転送すればよいだけなので更新時のトラフィックの軽減が期待できる。
- 各 Web サーバに検索機能を持たせることによりサーチエンジン専用のマシンが不要になる。つまり検索負荷を分散できる。
- 組織のある部門に関する情報はその部門が管理する、という情報の扱い方が理想的だが、各部門の Web サーバが自分の持つ情報について検索すれば、この理想に一歩近付くことができる。

3.2 LSE と Location Server が取り扱うデータ

上で述べたように CSE は 2 つの構成要素からなるが、LSE は全体的な情報を持たないものの、個々については非常に詳細な情報を持ち、Location Server は個々の詳しい情報は持たないものの全体の見通しが良いという特徴がある。したがって、LSE と Location Server が取り扱うデータの性質も使い方も違う。

LSE が扱うデータには、通常のサーチエンジンでも使うデータと、CSE の協調動作のために必要なデータの 2 種類に分けることができる。LSE が取り扱うデータには以下のようなものがある。

- 各文書の URL、タイトル、サマリー、スコア等、通常のサーチエンジンが扱う情報
- Location Server に登録するデータのうち、LSE が用意する必要がある LSE に登録されている文書数、文書の最大、最小スコア（キーワード毎に用意）、そのキーワードを含む文書数

これらのデータのうち、LSE の登録されている文書数、キーワードを含む文書数は後で述べる分散 tf·idf 法の計算のために必要なデータであり、本稿による拡張である。

この他にも静的な設定事項として、最低限 Location Server の URL、自分自身を起動するための URL を知っていなければならない。LSE での設定項目はインストールを用意にするため、なるべく少なくなるようにしている。Location Server に登録するデータは、更新の都度、LSE が生成するので Namazu か SGSE のインデックスをおくためのディスク容量があれば LSE をインストールできる。

Location Server が扱うデータは協調のためのデータだけで、抜粋的な内容がほとんどである。大きく分けると LSE 毎の抜粋的データと CSE 全体で共通なデータに分けられる。以下は Location Server が扱う LSE 毎の抜粋的データである。

- LSE に登録されている文書数、キーワードとキーワード毎の文書の最大最小スコア、そのキーワードを含む LSE 毎の文書数
- LSE を起動するための URL

また、CSE 全体で共通なデータには CSE 全体での文書数がある。Location Server が扱うデータのうち、本稿による拡張は、LSE に登録されている文書数、そのキーワードを含む LSE 毎の文書数、CSE 全体での文書数、キーワード毎の文書の最大最小スコアである。

これらのデータは全て動的なデータであり、管理者が設定する必要はない。後で述べるように、位置情報更新時に LSE が自動的に登録するデータである。管理者が設定する必要があるのはデータベースをおくディレクトリ等だけである。

3.3 CSE の動作

3.3.1 CSE の検索時の動作

Fig.1 に CSE の検索時の動作の概略図を示す。ユーザーの検索要求は入力フォーム付のページに対する入力としてブラウザ等から LSE に渡される。

以下では、ブラウザ等から検索すべき文字列を LSE が受け取った後、検索が行われる様子を見ていく。LSE_1 と LSE_2、Location Server がある環境を想定し、LSE_1 がユーザーから検索要求を受け取ったものとする。

1. LSE_1 はユーザーの検索要求を受け取る。受け取った検索論理式は内部で最適化する。
2. LSE_1 は Location Server に検索すべき文字列を送信する。

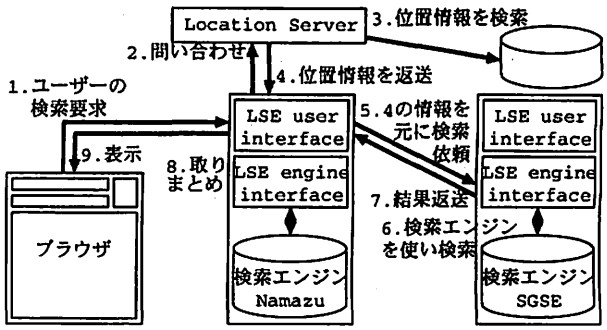

Fig. 1. CSE の全体的な動作の概念図

3. Location Server は位置情報を検索する。
4. Location Server は得られた位置情報のうちスコアが高いものをいくつか返送する。ここでは説明のために LSE_2 だけを返送するものとする。
5. LSE_1 は Location Server から送り返された位置情報をもとに LSE_2 に検索を依頼する。
6. LSE_2 では LSE_2 が持っているデータを既存のサーチエンジンを使って検索する。
7. LSE_2 は検索結果を LSE_1 に送り返す。
8. LSE_1 は LSE_2 の検索結果を取りまとめ、検索結果をブラウザなどに送り返す。

3.3.2 位置情報更新時の動作

CSE では Location Server があるキーワードに関してどこを捜せばよいかという位置情報を持つため、位置情報を定期的に更新する必要がある。また、CSE 全体として見たときに、インデックスの更新は一度に行えたほうが都合が良い。そこで、位置情報の更新に合わせて各 LSE が持つ文書のインデックスも更新する仕組みを用意した。

CSE の更新時の動作の概略図を Fig.2 に示す。

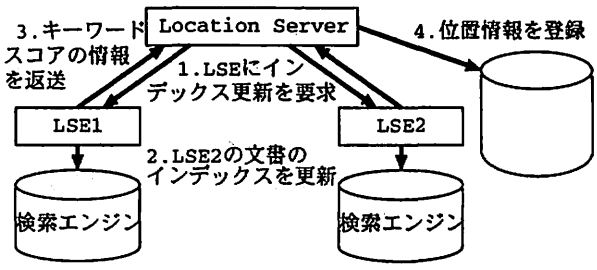

Fig. 2. CSE の位置情報更新時の動作

位置情報更新時には、Location Server が主体となって動作する。これは全体を制御するのに都合が良いからである。新しい LSE を Location Server に追加登録することを主な目的として、LSE が主体となって位置情報を更新することもできる。Location Server は LSE の自己申告にしたがって LSE を登録する。現在のところ、認証機能はない。

以下は更新時の動作の概略である。

1. Location Server は各 LSE にインデックスの更新とキーワード、スコア情報の返送を要求する。
2. 各 LSE はインデックスを更新し、キーワードとスコアに関する情報を抽出する。
3. 各 LSE はインデックスの更新が終ったらキーワードとスコアの情報を LS に返送する。
4. Location Server はキーワード、スコアを位置情報として登録し、LSE の問い合わせに答えられるようにする。

4 CSE で用いるプロトコル

CSE で用いている通信プロトコル (Cooerative Search Protocol、CSP) は、GMTP(Generic Message Transfer Protocol)[9] の上に構築されている。このため、本稿では特に CSP/GMTP と記述する。GMTP は 1 本のコネクション型の通信路において双方向に汎用メッセージの送受信を可能とする。引数は任意の文字列で、サイズに制限はなく、一つのメッセージに任意の個数の引数を含めることができる。GMTP は目的に応じてメソッドとその応答 (すなわち API) を定義することで特定用途に特化することができる。GMTP にはファイアーウォールを越えたり、CGI として実装されている LSE を Location Server が起動するときのために HTTP の上にピギーバック形式で送られる、GMTP over HTTP も定義されている。

CSP/GMTP は、実際には GMTP 上に定義されたメソッドとその応答である。以下では、GMTP 上の CSE 用 API を定義する。本稿での定義は分散 tf-idf 法によるスコアの計算を実現することを主な目的として、文献 [9] での定義を更新するものである。なお、GMTP の定義には変更はない。便宜上、下記のような疑似 RPC 形式で表す。

$$(y_1, y_2, .., y_m) = Method(x_1, x_2, .., x_n)$$

以下に Location Server が受け付けるメソッドの一覧を示す。void は引数、帰り値がないことを表す。

AskMe : (void) = AskMe(void)

Callback を実現するための何もしないメソッド。

Ask : (whole_num_docs, Hosts, idfs) = Ask(Expr)

検索論理式 Expr に応答できると期待される LSE の集合 Hosts を返す。whole_num_docs は CSE 全体の登録済文書の数である。

Update : (void) = Update(lse, lse_num_docs, weightKeys)

URL lse で表される LSE から、重み付けキーワード集合 weightKeys を受け取り、サイト集合を更新する。lse_num_docs は LSE に登録されている文書の数である。

以下に LSE が受け付けるメソッドの一覧を示す。

Search : (URLs) = Search(Expr, idfs)

検索論理式 Expr を検索した結果 URLs を返す。

UpdateMe : ( ) = UpdateMe(server)

server が示す URL に対して Update 要求を送る。

上のメソッドの説明に出て来る引数、帰り値の形式を以下に示す。まずは検索論理式 Expr の形式である。

```
Expr = Keyword
     | Phrase
     | "(" Expr ")"
     | "link:" TEXT
     | "title:" TEXT
     | "url:" TEXT
     | "site:" TEXT
     | Expr SP "AND" SP Expr    ; AND
     | Expr SP "NOT" SP Expr    ; NOT
     | Expr SP "OR" SP Expr     ; OR
Phrase = <"> TEXT <">
Keyword = 1*(ALPHA | DIGIT | EUC)
EUC = <any EUC encoded character
       (octets 128 - 255)>
```

サイト集合は、以下のように表す。

```
Hosts = *(URL CRLF)
```

URL 集合は以下のように表す。

```
URLs = "totalmatch:" 1*DIGIT CRLF CRLF
   *("title:" TEXT CRLF
     "score:" 1*DIGIT CRLF
     "url:" URI CRLF
     "summary:" TEXT CRLF)
   CRLF
```

ここで、URIはRFC1945で定義されたURIである。tolalmatchの行は全体のヒット数を表している。また、titleの行はタイトルを表し、scoreの行はスコアを表し、urlの行はヒットしたURLを表している。そして、summaryの行は概要を示している。

重み付けキーワード集合weightKeysは、キーワードとそのスコアのペアからなる集合である。

```
weightKeys = *(Phrase SP TfMax SP
   TfMin SP NumDocs CRLF)
   TfMax = 1*DIGIT
   TfMin = 1*DIGIT
   NumDocs = 1*DIGIT
```

TfMaxとTfMinはキーワードに対する各文書のスコアの最大値と最小値、NumDocsはそのキーワードを含む文書の数である。

idfsはキーワードとキーワード毎に計算したidfの値の組の集合である。

```
idfs = *(Phrase SP idfValue)
   idfValue = 1*DIGIT "." 1*DIGIT
```

5 CSEにおけるスコアの扱い

5.1 NamazuとSGSEのスコアの違い

CSEでは現在の所、NamazuとSGSEの2種類のサーチエンジンを使っている。NamazuとSGSEではスコアの計算方法が違うので本来はスコアの調整が必要である。Fig.3に示すようにNamazuのスコアに対するSGSEのスコアの比を計算すると、Namazuのスコアが20ポイント付近で小さくなったり、スコア毎の変動が大きいなど、若干ばらついているもののほぼ1前後に収まっている。また、各文書に対して双方のサーチエンジンが付けた文書に対するキーワード毎のスコアの違いは、概ね1から3程度の差で済んでいる。実際の検索ではこのスコアの違いはほとんど影響しないと考えて、現在のCSEではスコアの調整は行っていない。しかし、将来的にFreya等、他のサーチエンジンに対応したり、CSE専用のサーチエンジンを作成する場合には調整が必要になると思われる。

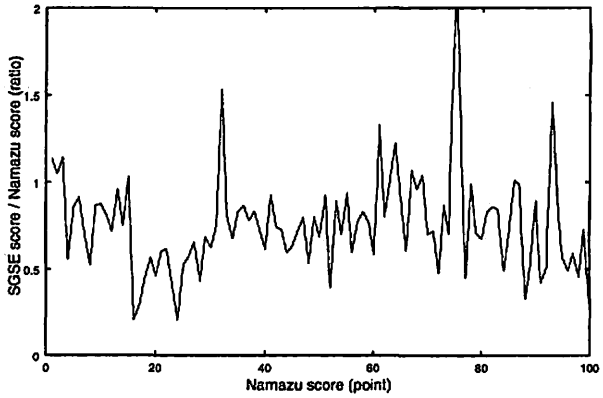

Fig. 3. NamazuとSGSEのスコアの比
Namazuのスコアが同じキーワード・文書に対してSGSEが付けたスコアをNamazuのスコア毎に平均した値の比。
http://www.toyonet.toyo.ac.jp/以下の文書を使用した。

5.2 LSEにおける分散tf·idf法

tf·idf法はキーワードが文書中に出現する頻度だけでなく、キーワードの『珍しさ』も考慮するスコアの計算方法で、論理型検索の時に効果的である。tfはキーワードの単純な出現回数、idfはキーワードが一部の文書に集中している度合である。tf·idf法を用いる際のスコアの計算式を以下に示す。

$$score = tf \cdot idf \quad (1)$$

$$idf = \log \frac{N}{n} \quad (2)$$

ただし、Nは全文書数、nはヒットした文書数である。CSEにおいて分散tf·idfを用いる際のidf_{dist}の計算は以下のようになる。

$$idf_{dist} = \log \frac{N_{total}}{\sum n_{keyword,i}} \quad (3)$$

ただし、N_{total}はCSE全体の文書数、$n_{keyword,i}$はキーワード毎に各LSEでヒットした文書数である。

分散tf·idfは既にWHERE[5]等で用いられている。CSEでは、idfの計算用にLSのデータベース中のN_{total}、$n_{keyword,i}$を、tfの値として各文書のスコアを用いてユーザーに返送するスコアを計算している。ユーザーのブラウザに表示するtf·idf法によるスコアの計算は各LSEで分散して行うため、Location Serverにtf·idf法の計算に必要なデータを記憶しておく必要がある。最終的なスコアの計算を分散して行う事で、ユーザーから検索要求を受け取ったLSEは最後に他のLSEの検索結果をまとめてソートするだけで済む。

Namazu はデフォルトの状態では tf·idf 法が既に適用されたスコアを出力するので、tf·idf 法適用前のスコアを出力するように設定して使用する。SGSE が出力するスコアは tf·idf 法を使っていないので、そのまま使用している。

N_{total} は Location Server に対して Ask メソッドでキーワードに関する情報を知っていると思われる LSE を問い合わせた際に返される、whole_num_docs を用いる。whole_num_docs は位置情報更新時に各 LSE が Location Server に送った LSE が保有する文書数を合計したもので、Location Server が保管している数値である。$\sum n_{keyword,i}$ は Location Server のデータベースにある各 LSE におけるキーワード毎のヒット数を使う。N_{total} と $\sum n_{keyword,i}$ は Location Server のデータベース更新時に得られる値で、次回の更新まで変化しないため、検索動作を単純化・高速化する目的で Location Server に値を置いている。これにより、スコアの計算も完全に分散して行われる。

5.2.1 Location Server における検索対象 LSE の絞り込み

検索対象の LSE を絞り込まなかった時、検索時の負荷が非常に高くなることが考えられ、検索する LSE を絞り込むことが必要になる。しかし、各 LSE の文書の詳細な情報が Location Server には無いので、単純に LSE の各文書のスコアの合計を使った絞り込みではスコアの高い文書を発見できない事が考えられる。

絞り込を目的として、Location Server は各 LSE の文書の最大スコアと最小スコアを保持している。LSE を絞り込む手順は以下の通りである。

1. 最も高い最大スコアを持つ LSE を探す。この LSE を LSE_1 とする。
2. LSE_1 の最小スコアより大きい最大スコアを持つ LSE をリストアップする。
3. LSE_1 とこれらリストアップされた LSE を検索対象とする。これらの LSE のリストは最大スコアが小さくなる順にソートされる。

Location Server から検索すべき LSE のリストを受け取った LSE は並列的に検索を行うので、このリストは本来ならばソートされている必要は無いが、LSE が何らかの都合によりこのリストの一部だけを検索する時に備えている。

また、これだけでは CSE のシステム全体を検索できないので、リストアップされなかった LSE も最大スコアの順にソートして返している。これは、上の手順で検索対象とした LSE の検索結果だけではユーザーが満足しない場合に備えての事である。

6 まとめと今後の課題

本稿では、協調サーチエンジンにおいて、tf·idf 法を使用したスコアの計算方法と検索対象をスコアが高い文書を持っていると思われる LSE に絞り込む方法について述べた。この結果、協調サーチエンジン全体で統一的なスコアの取扱が実現したほか、重要なキーワードの弁別能力が増し、キーワードスパミング等への耐性も高くなったと考えられる。また、検索対象 LSE の絞り込みにより検索時の負荷の軽減もはかられる。

これからの課題として、現在の協調サーチエンジンが抱える問題点を挙げる。

- LS からの要求だけではなく、LSE 側からも自発的に Location Server のデータの更新を行えるが Location Server は LSE を認証しないため、不正な処理を行うクライアントを防げない。
- Location Server は一つしか無いため、大規模分散環境に適さない。Location Server の分散化は今後の課題である。

参考文献

[1] 馬場 肇『日本語全文検索エンジンソフトウェアのリスト』 http://www.kusastro.kyoto-u.ac.jp/~baba/wais/other-system.html

[2] 山名早人『*Trends of WWW Search Engines*』 http://www.etl.go.jp/~yamana/Research/WWW/survey.html (1998)

[3] 『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 (1997)

[4] NTT ソフトウェア研究所 Ingrid 研究プロジェクト『*Ingrid: 広域情報検索システム*』 http://www.ingrid.org/

[5] Miguel Rio, Joaquim Macedo, and Vasco Freitas *"A Distributed Weighted Centroid-based Indexing System"*

[6] 高林 哲 『全文検索システム *Namazu*』 http://openlab.ring.gr.jp/Nnamazu/

[7] 『*SonyDrive Search Engine*』 http://www.sony.co.jp/sd/Search/SGSE-DL.html

[8] 山本崇、佐藤永欣、西田喜裕、上原稔、森秀樹『協調サーチエンジンの研究』,DICOMO'99,p169-174(1999)

[9] 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