Web コンテンツの自動選別システムの開発

03H022 樫尾豪人

目次

1 はじめに 1

2 現状と改善策 2
2.1 インターネットの現状 2
2.2 改善策 3

3 Web コンテンツの自動選別システム 4

4 使用方法 5
4.1 Web コンテンツ選別画面 5
4.2 キーワード編集画面 6
4.3 Google 検索 7
4.4 結果表示 8

5 開発環境 9

6 まとめ 10

付録 A プログラムのソース 11
A.1 Web コンテンツ選別のプログラム 11
A.2 キーワード登録、削除のプログラム 13
A.3 Google 検索のプログラム 15

1　はじめに

1960 年代に誕生したインターネットは、今日までに急激に進化してきた。現在利用されているインターネットの原案ともいえる、パケット交換方式[*1]による分散型通信ネットワーク概論が、RAND 社のポール・バラン (Paul Baran) 氏によって提唱され、この論理が現実の通信ネットワークとして登場したものが ARPANET[*2]である。最初は軍事的なものとして生まれてきたインターネットも、学術的なものへと姿を変え、現在では社会基盤にまで発展し、メイルや Web ページの閲覧だけでなく、電話や映像コンテンツの配信などのいろいろな分野で使われるようになっている。インターネットは私達が生活する上でなくてはならないものになり、現在では、そのネットワークは地球全土をカバーするまでになった。

近年のコンピュータ技術の進歩と共に、コンピュータの価格が下り、それによりコンピュータが各家庭に急速に普及し、それと同時に、通信インフラの整備[*3]も急速に進んできた。それらのことにより、インターネットを利用できる環境が拡大した。それに伴って、インターネットを利用する人口も増え、インターネット利用者が増るのに対応するようにインターネット上には多種多様な Web コンテンツが膨大に生まれ、日々新しい Web コンテンツが増え続けている。

インターネット上の膨大なコンテンツの中から自分の欲しい Web コンテンツを見付け易くするために、Google、Yahoo!などに代表されるような検索エンジンが出てきた。しかし、そのような検索エンジンを用いても、自分の欲しい Web コンテンツを見付けることは、容易ではない。

そこで、インターネット上から興味がある Web コンテンツをより早く簡単に見つけ出すことを可能にするシステムの開発を行い、インターネット利用者をサポートし負担を軽減することを目的とする。

[*1] パケット交換方式... 送信するディジタル・データをパケットという単位に分割して、通信する方式

[*2] ARPANET... Advanced Research Projects Agency Network:国防総省高等研究計画ネットワーク

[*3] 通信インフラの整備... 公共のために通信施設や通信網を作り上げること

2　現状と改善策

2.1　インターネットの現状

現在インターネットは世界中の人が利用している。膨大な人数が利用し、興味の対象もその数あるため、インターネット上には膨大な量の Web コンテンツがある。その中から、自分の興味のある Web コンテンツを新に捜し出すには大変な労力と時間が必要である。そのために、インターネット利用者には大きな負担がかかっている。ポータルサイトや検索エンジンでは Web コンテンツを探しやすくするためのさまざまな機能があるが、一人々々の興味の対象から Web コンテンツを、自動的に選別する機能がない。また、大規模なポータルサイトや検索エンジンのような大規模ネットワークでは、利用者が多すぎるため興味の対象が分散しすぎ、実現しにくい機能である。

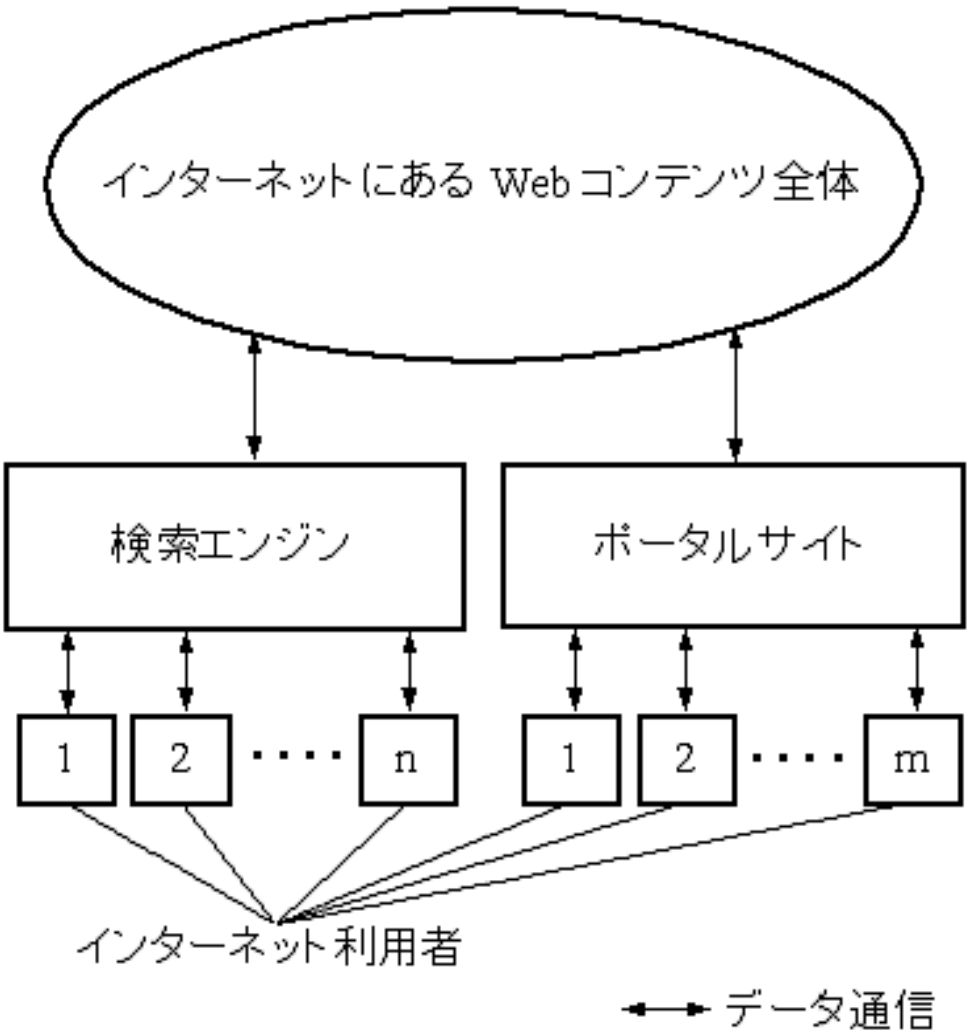

図 1　インターネットの現状

2.2 改善策

大規模ネットワークでは、利用者が多すぎ、興味の対象が分散しすぎるため実現しにくい機能だが、小規模ネットワークとして考えた場合、その中でインターネットを利用する人は同じグループに所属していたり、同じことを調べたりと興味の対象が似ている可能性が高いと思われる。そこで、小規模ネットワーク内から Web コンテンツを見たログと自分の興味のあるキーワードに対する Web コンテンツの URL とを比較し、合致したものを抽出することにより、検索エンジンでキーワードを用いて検索するよりも、利用価値の高い Web コンテンツを優先的に見つけ出すシステムの提案する。このような小規模でネットワークの中でのインターネット利用者、一人々々の興味の対象から Web コンテンツを自動的に選別するシステムの実現を行なう。

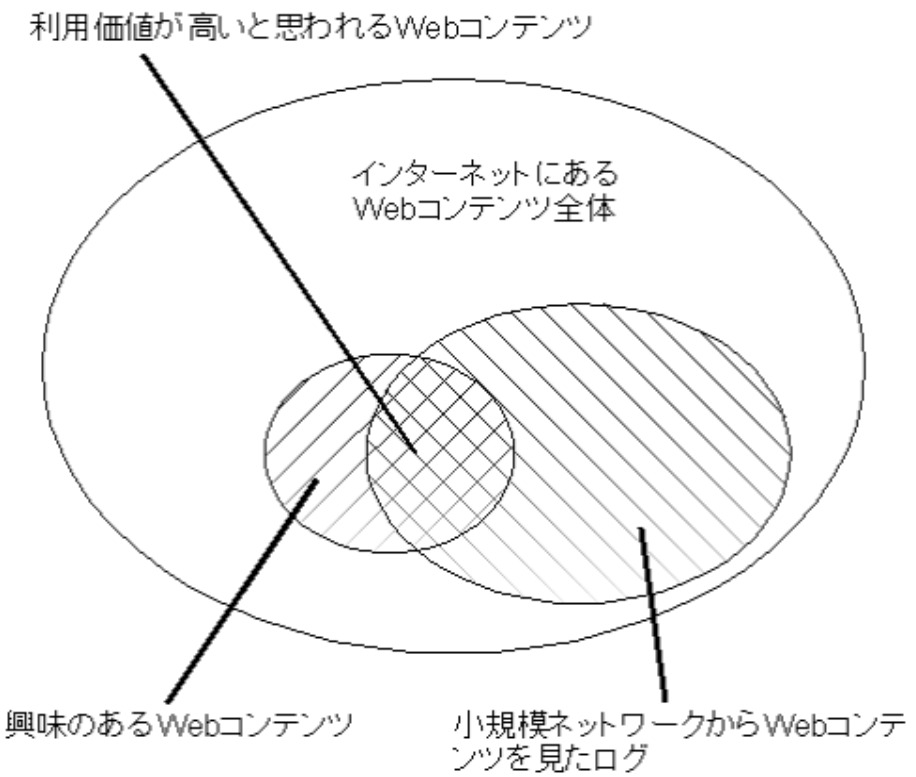

図 2　Web コンテンツの分布

3　Web コンテンツの自動選別システム

Web コンテンツの選別には、小規模ネットワーク内で閲覧された Web コンテンツに対しての URL のログ、興味のあるキーワードのデータベース、及び Google SOAP Search API を使用し、以下の通りにした。

1. 興味のあるキーワードをデータベースに登録する
2. 興味のあるキーワードのデータベースを用いて、Google SOAP Search API を使い検索を行う
3. Google SOAP Search API から帰ってきた結果と小規模ネットワーク内から Web コンテンツを見たログとを比較する
4. 合致するものと、合致しなかったものを別々に出力する

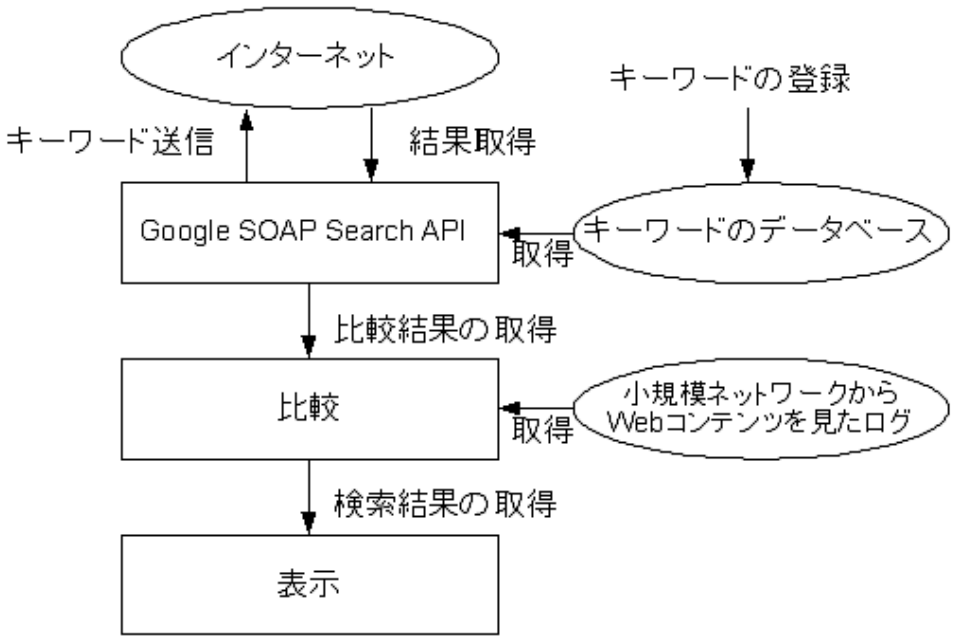

図 3　Web コンテンツの選別

4　使用方法

4.1　Web コンテンツ選別画面

Web コンテンツの自動選別システムの使用方法

コンテンツ選別画面では、フレームで 2 つに分割された画面が表示され、Web コンテンツ選別とキーワード編集と Google 検索のリンクがあるウインドウと、小規模ネットワーク内から Web コンテンツを見たログと合致したものと合致しなかったものを表示するウインドウが表示される。Web コンテンツ選別のリンクをクリックすることにより Web コンテンツの選別を行うことができ、キーワード編集のリンクをクリックすることによりキーワード編集画面へ移ることができる。また、Google 検索のリンクをクリックすることにより、データベースに登録されている興味のあるキーワードを用いて検索を行うことができる。

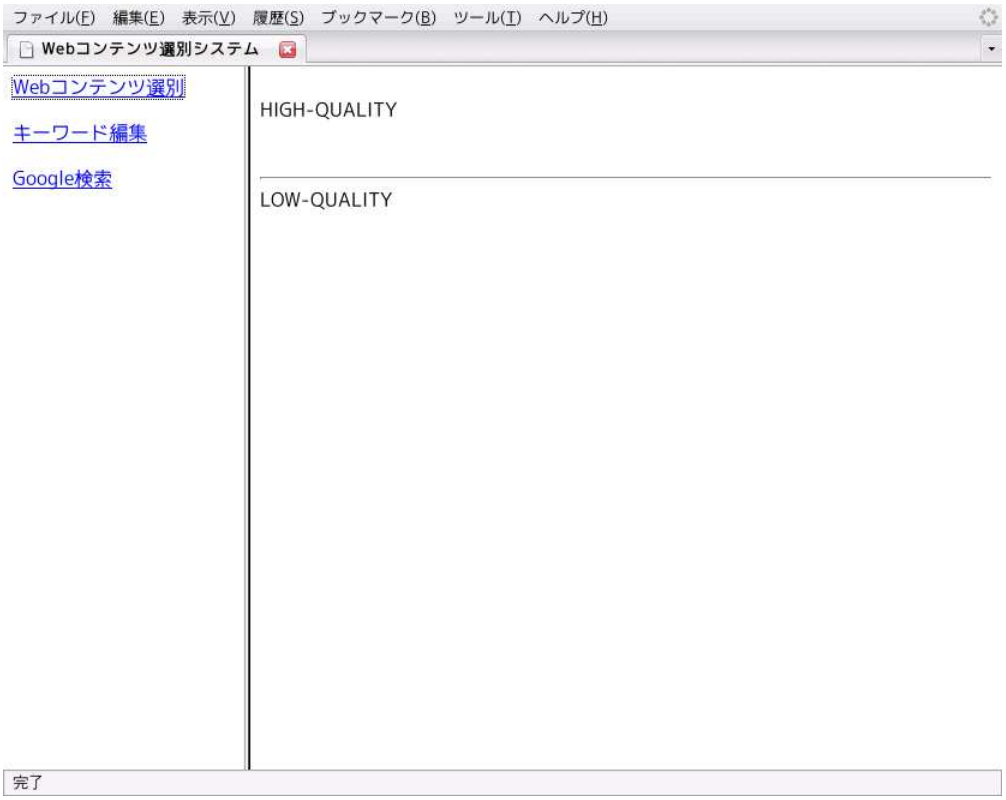

図 4　Web コンテンツ選別画面

4.2　キーワード編集画面

　キーワード編集画面では、テキストボックスにキーワードを入力し、登録ボタンを押すことによってキーワードを登録することができる。また、テキストボックスにキーワードを入力し削除ボタンを押すことによってキーワードを削除できる。

図 5　キーワード編集画面

4.3　Google 検索

Google 検索のリンクをクリックすることにより、データベースに登録されている興味のあるキーワードを用いて Google のデータベースから検索を行い、その結果を出力する。

図 6　データベースを用いた検索結果

4.4　結果表示

コンテンツ選別画面で Web コンテンツ選別のリンクをクリックすると、コンテンツが選別されて表示される。HIGH-QUALITY と書かれた下に表示されているものが小規模ネットワーク内から Web コンテンツを見たログと合致したものである。LOW-QUALITY と書かれた下に表示されているのが、小規模ネットワーク内から Web コンテンツを見たログと合致しなかったものである。

図 7　選別結果

5 開発環境

システムの開発に使用したプログラミング言語やモジュールなどの環境

- Apache
  - 1995 年に NCSA httpd 1.3 をベースに、開発が始まった Web サーバで、数多くある Web サーバの中で、最大のシェアを誇る Web サーバである。NCSA httpd に patch[*4]を当てたものが、Apache であり、「A patch」(パッチを当てた) をもじり、アメリカの先住民族の部族名である Apache とダブらせて名づけられた。
- HTML(Hyper Text Making Language)
  - WWW 上に HyperText[*5]を情報発信するための Markup Language[*6]で、HyperText Markup Language の頭文字をとった略称である。
- Perl
  - Larry Wall 氏が開発したプログラミング言語で、テキストの検索や抽出に向いた言語である。インタプリタ型であるため、プログラムを作成したら、コンパイルなどの処理を行なうことなく、すぐに実行することができる。
- Google SOAP Search API
  - Google の検索エンジンをプログラミング言語から 利用できるようにした Web サービスで、Web 検索、スペルミス指摘、などの取り出しが可能である。SOAP[*7]と呼ばれる Web サービスの仕様に添って作られている。また、この API には WSDL[*8]ファイルが用意されているため、様々なプログラミング言語から使用することができる。
- SOAP::Lite モジュール
  - SOAP を利用するための Perl モジュールのコレクションで、ここでは Google SOAP Search API を Perl から使用するために用いた。
- Squid
  - HTTP などの通信を中継・キャッシュするプロキシサーバ[*9]。

[*4] patch... 完成したプログラムの間違い部分を修正するファイル
[*5] HyperText... 文書の任意の場所に、他の文書の位置情報を埋めこみ、複数の文書を相互に連結する仕組
[*6] Markup Language... 文章の構造や、修飾情報を、文章中に記述することができる記述言語
[*7] SOAP... XML と HTTP などをベースとした、他のコンピュータにあるデータやサービスを呼び出すためのプロトコル (通信規約)
[*8] WSDL... Web サービスを記述するための、XML をベースとした言語仕様
[*9] プロキシサーバ... 内部ネットワークからインターネット接続を行う際に、セキュリティ確保と高速アクセスを実現するために設置されるサーバ

6　まとめ

Web コンテンツ自動選別システムを用いることで、より早く簡単に興味があるコンテンツを見つけ出すことができる。それにより無駄な時間や労力を削減することができるため、Web コンテンツを見つけ出すために本来使うはずであった時間や労力を、他の生産的な活動に転換することができる。

謝辞

本研究を進めていく上で、御指導及び御協力を頂いた大垣 斉講師・藤井 信夫教授・fken.a4w メイルリストのメンバーの方々、そして情報安全工学研究室の学生の方々に深く感謝の意を表します。

参考文献

[1] Tara Calishain, Real Dornfest 著 山名早人 監訳 田中裕子訳. GoogleHacks. 株式会社オライリージャパン. 2004 年. ISBN4-87311-136-6

付録 A　プログラムのソース

A.1　Web コンテンツ選別のプログラム

```
#!/usr/bin/perl

open(FH,"goo_URL");
@goo = <FH>;
close(FH);
open(FH2,"all_URL");
@log = <FH2>;
close(FH2);
open(FH3,"goo_link");
@link = <FH3>;
close(FH3);

$a=0;
$b=0;
$c=0;

while($log[$a])
{
$b=0;
while($goo[$b])
{
if($log[$a] eq $goo[$b])
{
$hit_url[$c] = $goo[$b];
$hit_link[$c] = $link[$b];
$c++;
}
$b++;
}
$a++;
}


$a=0;
while($hit_url[$a])
{
$b=0;
while($goo[$b])
```

```
{
if($hit_url[$a] eq $goo[$b])
{
$link[$b] = "";
}
$b++;
}
$a++;
}


print <<END;
Content-type: text/html\n\n
<meta http-equiv="CONTENT-TYPE" content="text/html; CHARSET=UTF8">
<html>
<head>
<title>選別</title>
</head>
<body>
<br>
HIGH-QUALITY<br>
<br>
@hit_link<br>
<hr>
LOW-QUALITY<br>
<br>
@link<br>
<br>
</body>
</html>
END
```

A.2 キーワード登録、削除のプログラム

```
#!/usr/bin/perl

if($ENV{'REQUEST_METHOD'} eq 'POST')
{
read(STDIN,$query, $ENV{'CONTENT_LENGTH'});
}
else
{
$query = $ENV{'QUERY_STRING'};
}

if($query =~ /tasu/)
{
$query =~ s/tasu=//;
open(FH,">>keyword");
print FH $query;
print FH "\n";
close(FH);
}
elsif($query =~ /kesu/)
{
open(FH,"keyword");
@key = <FH>;
close(FH);

$query =~ s/kesu=//;
$a=0;
while($key[$a])
{
$keys = $key[$a];
chomp $keys;
if($keys eq $query)
{
$key[$a] =~ s/$key[$a]//;
}
$a++;
}
open(FH,">keyword");
print FH @key;
close(FH);
```

```
}

open(FH,"keyword");
@keyword = <FH>;
close(FH);

print <<END;
Content-type: text/html\n\n
<meta http-equiv="CONTENT-TYPE" content="text/html; CHARSET=EUC-JP">
<html>
<head>
<title>キーワード編集</title>
</head>

<body>
キーワード編集<br>
<hr>
<form action="key.cgi" method="POST">
キーワード<input type="text" name="tasu" size="40"><br>
<input type="submit" value="登録">
</form>
<hr>
<form action="key.cgi" method="POST">
キーワード<input type="text" name="kesu" size="40"><br>
<input type="submit" value="削除">
</form>
<hr>
・キーワードリスト<br>
@keyword
<hr>
</body>
</html>
END
```

A.3 Google 検索のプログラム

```
#!/usr/bin/perl

use strict;
use encoding "utf8";
use SOAP::Lite;

open(FH,"keyword");
my @key = <FH>;
close(FH);
open(FH2,">>goo_URL");
open(FH3,">>goo_link");

my $count=0;

while($key[$count])
{
my $WSDL = 'file:GoogleSearch.wsdl';
     my $APIKEY = '';
     my $KEYWORD = $key[$count];

     my $query = [
         $APIKEY,
         $KEYWORD,
         0,
         10,
         'false',
         'countryJP',
         'false',
         'lang_ja',
         'utf8',
         'utf8'
     ];

    my $service = SOAP::Lite->service( $WSDL );
     my $result = $service->doGoogleSearch( @$query );

     return unless ref $result;
     return unless ref $result->{resultElements};

     foreach my $item ( @{$result->{resultElements}} )
```

```
{

        print FH3 "<a href=\"";
        print FH3 $item->{URL};
        print FH3 "\">";
        print FH3 $item->{title};
        print FH3 "</a><br>";
        print FH3 "\n";

        print FH2 $item->{URL};
        print FH2 "\n";
    }
$count++;
}
close(FH2);
close(FH3);

open(FH,"goo_link");
my @link = <FH>;
close(FH);

print <<END;
Content-type: text/html\n\n
<meta http-equiv="CONTENT-TYPE" content="text/html; CHARSET=UTF8">
<html>
<head>
<title>検索結果</title>
</head>

<body>
検索結果<br>
<hr>
@link
</body>
</html>
END
```