Web ページの閲覧履歴の自動作成と共有の研究

97H025 金谷一馬

2001年 2月 23日

目 次

1 はじめに **1**

2 本研究の概要 **2**
2.1 オートブックマークを使用するための環境 2
2.2 オートブックマーク 3

3 ABM の研究 **5**

4 Everybody's Book Mark **7**
4.1 URL 登録 8
4.2 考察 9

5 謝辞 **9**

1　はじめに

近年、さまざまな情報が Web ブラウザーで閲覧することができる。いつも使用している端末からだけではなく、いろいろな端末から Web ブラウザーで閲覧する機会もある。その際に、過去に閲覧した Web ページの URL を参照したいときがある。Web ブラウザーには、過去に閲覧した Web ページの URL の一覧を表示する機能はあるが、大学の演習室のような多数のパソコンを、特定多数のユーザーが使用する環境 (ユーザーが前回利用したものと違うパソコンを利用する、または、特定の一台を利用していてもユーザーが利用していない時に他のユーザーが利用する環境) では、

WWW ブラウザの履歴データが保存されている先は、使用したクライアントマシンなので、次に履歴を表示させる場合には、同じクライアントマシンを使用しなければならない。

多数の人がクライアントマシンを使用するので、他人の履歴が混ざってしまうので自分専用の履歴を閲覧することが出来ない。

上記の問題解決のため 1999 年度の卒研生がオートブックマーク [1][2] を作成した。それを、より良いオートブックマークにするための改良及び URL の共有をすることを検討する。

2　本研究の概要

オートブックマーク (以下 ABM) は個人の過去の履歴を閲覧するものなので、情報資源の共有ができない。本研究ではオートブックマークの改良およびユーザーが良い情報を所有する Web ページや FTP 等の URL を登録しあうことにより情報の共有が出来るものの作成をする。URL を登録しあうことにより情報の共有が出来るものを EBM(Everybady's Book Mark) と呼ぶこととする。

2.1　オートブックマークを使用するための環境

ABM を利用するには、ユーザーはブラウザーの Proxy 設定を変更し、特定の Proxy サーバーを使うようにしなくてはならない。また、Proxy サーバなどに接続しているクライアントのユーザ名を返してくる ident デーモンが必要である。

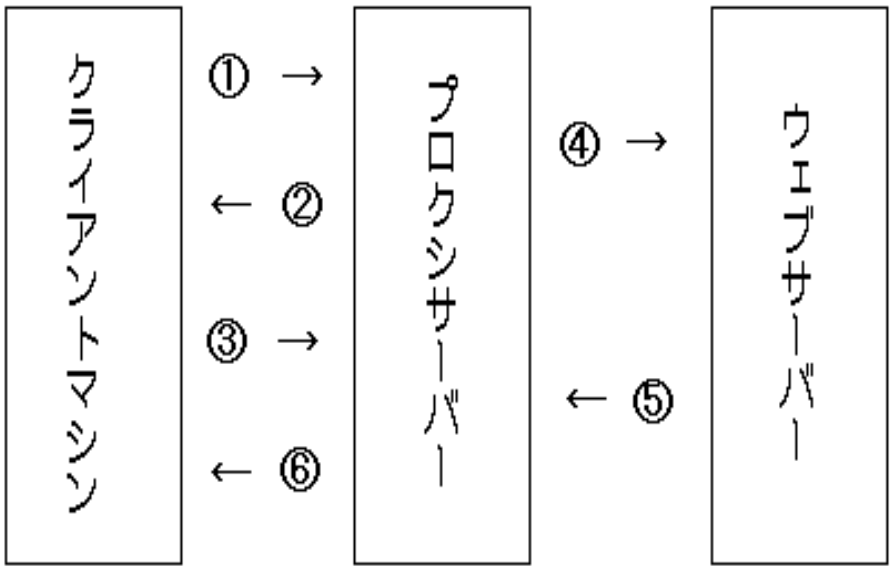

図 1: クライアントとプロクシ・ウェブサーバーの関係

クライアントとプロクシ・ウェブサーバーの関係を示す。
(図 1の表記去れている数字と対応させてある。)

1. クライアントから proxy サーバーに対して、Web ページのデータを請求する
2. proxy サーバーはクライアントの ident デーモンに対して、ユーザ名を問い合わせをする。
3. クライアントの ident デーモンは proxy サーバーにユーザー名を返す。
4. proxy サーバーは、Web サーバーに Web ページのデータを請求する。
5. proxy サーバーは、proxy サーバーに Web ページのデータをわたす。
6. proxy サーバーはクライアントに、Web ページのデータをわたす。

2.2 オートブックマーク

ABM とは、Proxy サーバーのログファイルを用い個人専用の過去参照した URL を Web ページで表示させるものである。

ABM は、Proxy サーバのログファイルの情報を用いるので、ユーザーはブラウザーの Proxy 設定を変更し、特定の Proxy サーバーを使うようにしなくてはならない。また、URL を残すのと共にユーザー名をログファイルに残すようにするため、Proxy サーバーにユーザー名を渡すデーモンを端末で使用しなくてはならない。

これにより、誰がどの URL を閲覧したかの特定ができるようになる。

ABM の表示方法として URL の全体を表示する詳細版 (図 2)、ドメインまでを表示する簡易版 (図 3)、及びドメイン名と一つ目のパス名までを表示するようにした普通版 (図 4) を任意にユーザーが選択し表示できるようにした。[3]

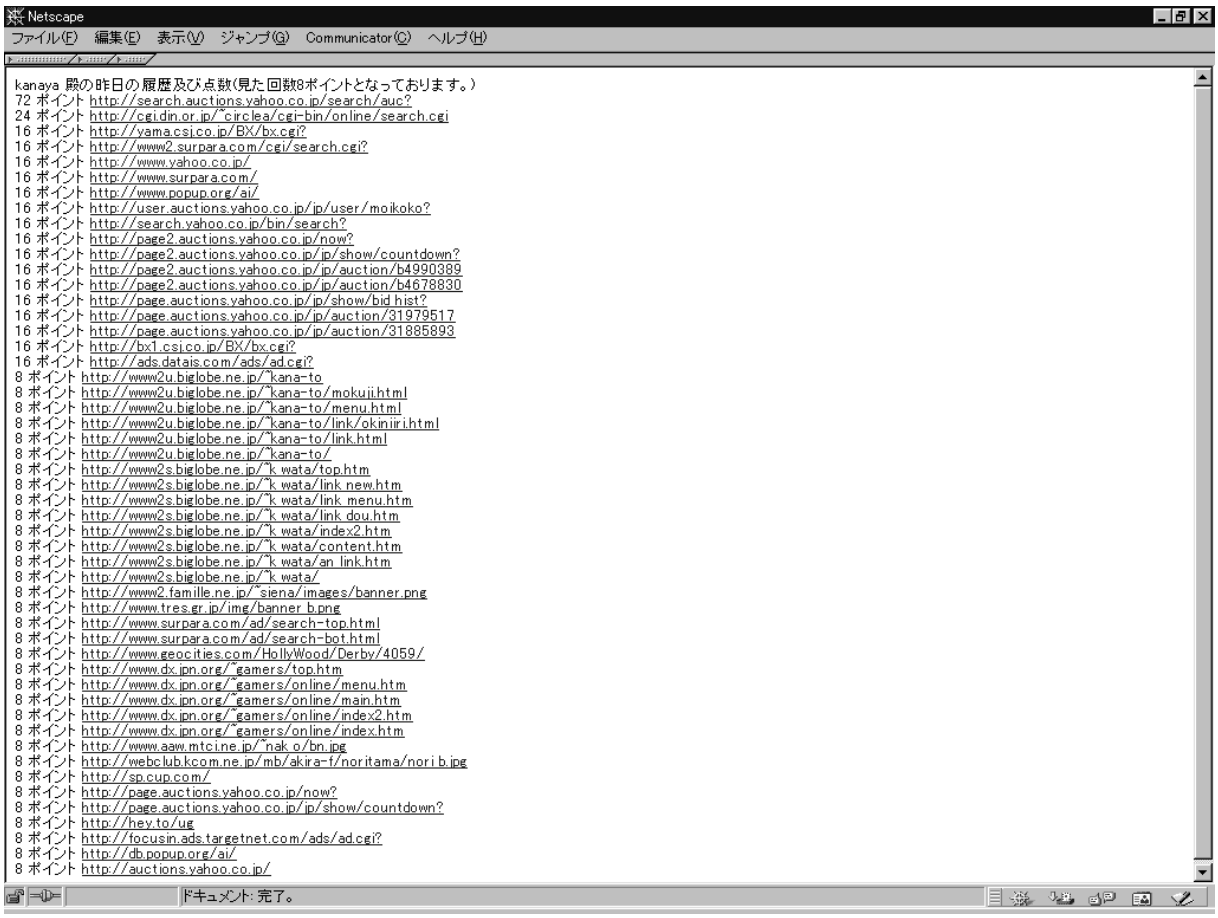

図 2: ABM:詳細版

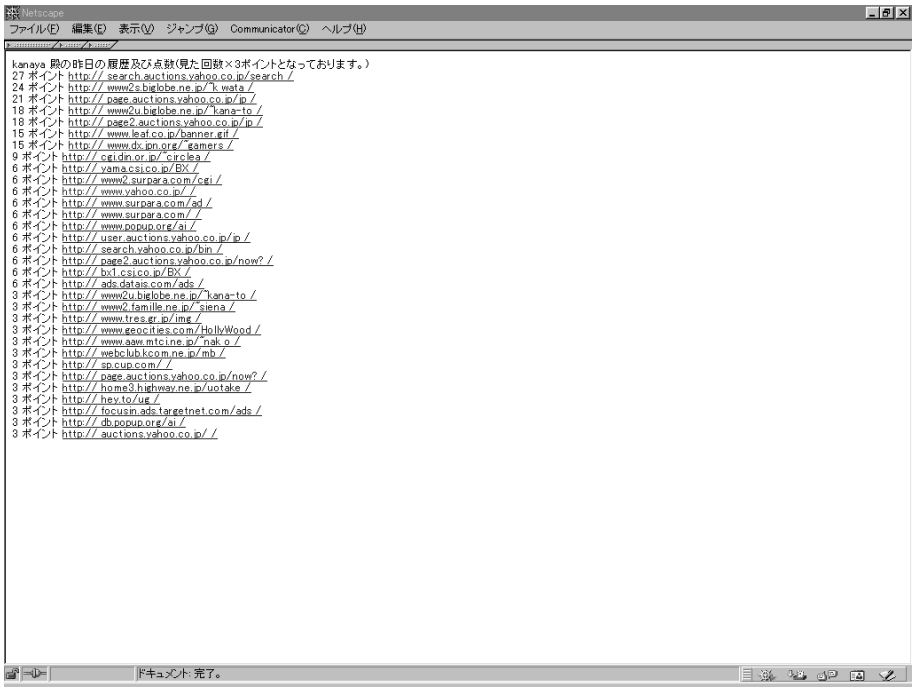

図 3: ABM:普通版

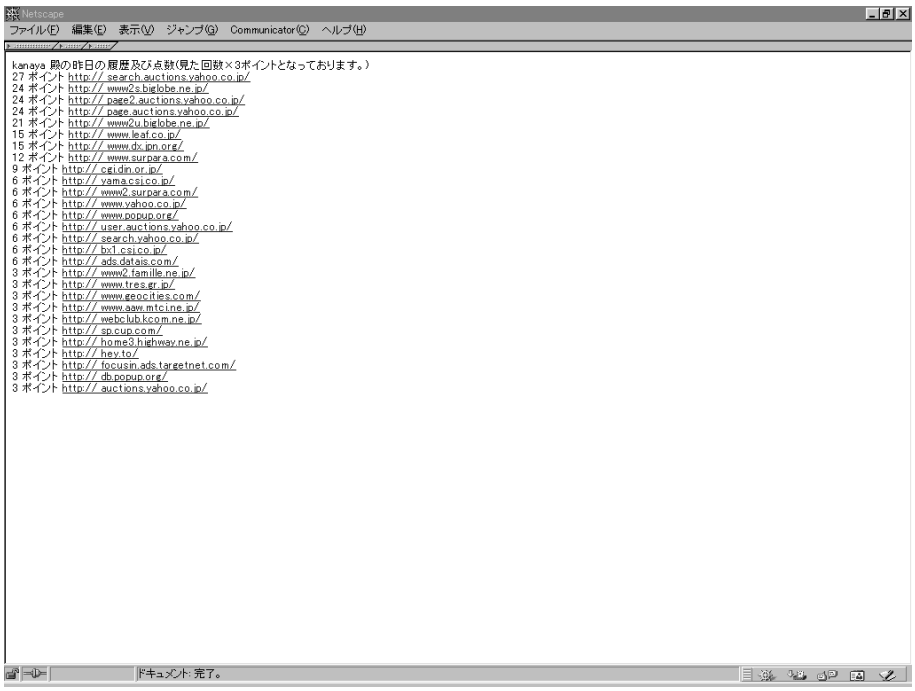

図 4: ABM:簡易版

3　ABM の研究

現在の ABM は、過去 5 日間閲覧した URL を点数化した合計により、点数の高いものから表示されるようにした。

URL の閲覧時期による重みづけ× 閲覧回数で表示される点数を計算している。藤井研が使用している Proxy サーバーのログをユーザーと URL をセットにして、いつ参照されたかを調べた結果 (図 5)、よく使用する URL は毎日のように使用される、このようなものばかりだけなら URL の閲覧時期による重みづけは必要とされないが、その日だけ必要であり使用しているであろう URL は、何回かその日のうちに見直される確立が非常に高い、そして、後に見直される確立が低いので、URL の参照した時期が古いものは点数を低く、新しいもの程点数を高くすることにより、必要である確立が高いもの程、点数を高くするようにするため、重みづけをした。また、再度 URL を参照されるであろう確立が高いのは 5 日までなので、過去 5 日間の表示にした (図 6)。

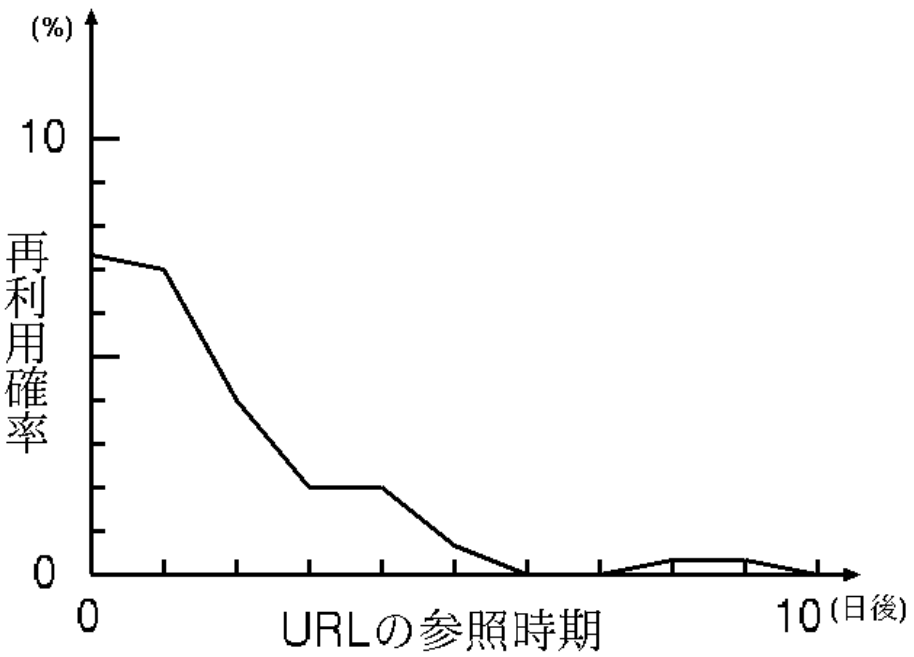

図 5: 藤井研究室での一度利用した URL の再利用率

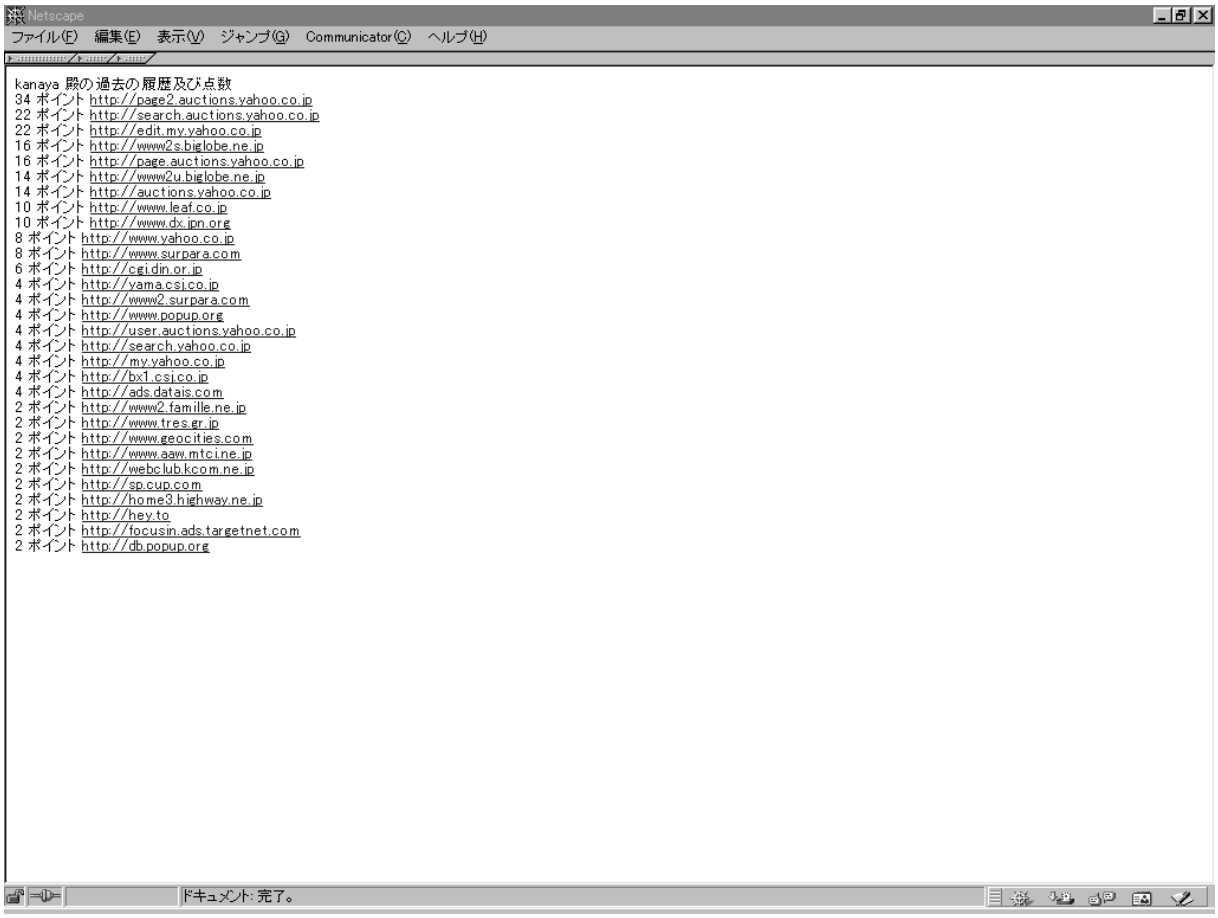

図 6: ABM:過去 5 日間の表示

4 Everybody's Book Mark

ABM では個人が過去閲覧した URL を利用するものなのでみんなで利用することが出来ない。よい情報は、たくさんのユーザーと共有するべきであるので、私は EBM を提唱する。他のユーザーから推奨してもらえると言うことは、登録してくれたユーザーには、認められるような良いページである。登録してもらった URL は、登録した以外ののユーザーにとって必要な情報をもちえているとは限らないが、たくさんの URL をユーザーより登録してもらうことにより必要とする情報を所有する Web ページの URL が利用できると思われる。

だから、EBM はいろいろな情報をみんなで共有するために、ユーザーから情報を提供してもらうことにある。ユーザーから提供してもらった URL を Web ページで他のユーザーに提供する一連のものが EBM である。(図 7は、登録してもらった URL を表示するの例である)

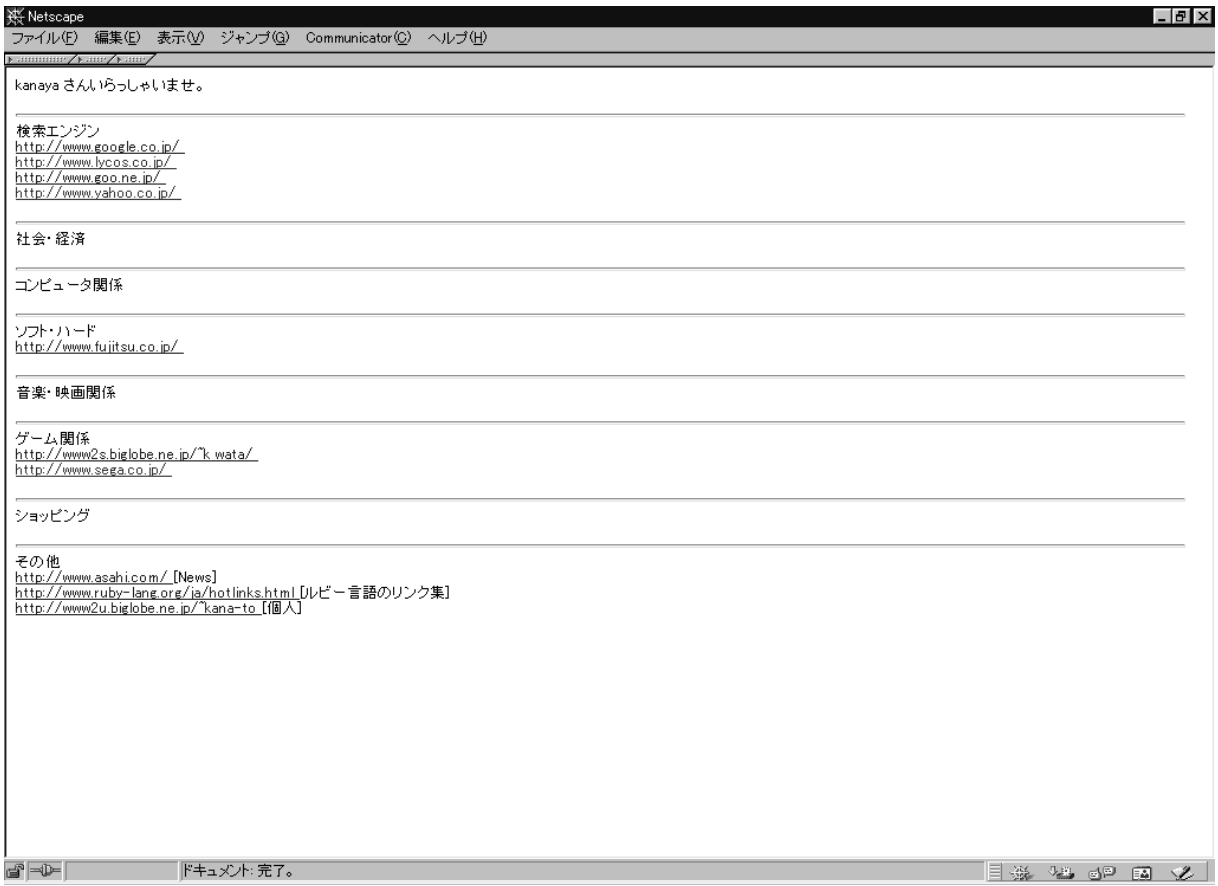

図 7: Everybody's Book Mark

4.1 URL 登録

URL 登録については、yahoo 等で既に採り入れられている。yahoo と EBM の相違点は、登録してもらう際に、ユーザーに登録をしてもらうという負担がかかる、登録してもらう URL を必ずしも覚えているとは限らない。そこで、わたしは ABM を用いたということ、登録してもらえるということは一度自分でその URL を見に行ったことがあると考えそれならば、ABM に記録が残っている確率が高いはずである、そこで、ABM に登録ボタンをつけて容易に登録できるようにする。

これにより、ユーザーへの登録の負担は減るものと考えた。(図 8は、ABM を用いて URL 登録するの例である)

しかし、必ずしも ABM に記録があるとはかきらず、ABM なしに任意に登録できるようにもしたがこれでは、こころともない一部のユーザにより悪戯される可能性がある。その対処方として ident デーモンが動作してないときは、書き込みが出来ないようにしました。

図 8: ABM を用いての EBM の登録

4.2 考察

本研究で ABM は、閲覧回数・閲覧時期による点数の重みづけにより URL の利用状況がわかり、自分の利用したい URL が良く利用されるものか利用されないものかで、検索しやすくなった。

EBM は、ABM の機能を利用することにより登録する URL を忘れても登録出来るようになった。

しかし、ユーザーが URL の新規ジャンル登録ができず、現段階では管理側が登録しなくてはない。ユーザーが URL の新規ジャンル登録ができれば、管理者側の負担が減る。

さらに、登録された URL で利用されていない URL 等を自動で消去することにより、EBM をよりスムーズに利用することが出来る。

以上のようにユーザーや管理者の負担を更に減らせることが分かる。今後は、これらの改善が課題である。

5 謝辞

本研究を進めていく上で御指導頂いた大垣 斉講師、藤井 信教授, 中村 孝講師ならびに各研究室の方々に深く感謝の意を表します。

参考文献

[1] 1999 年度卒業 宮北 哲博: 卒業論文,
http://www-fken.ise.osaka-sandai.ac.jp/fken99g/42/H/

[2] 1999 年度卒業 高徳 哲也: 卒業論文,
http://www-fken.ise.osaka-sandai.ac.jp/fken99g/41/H/

[3] 大垣 斉: ABM,
http://www-fken.ise.osaka-sandai.ac.jp/ABM/index.html

[4] AWK を 256 倍使うための本：株式会社アスキー,
志村拓
1993 年
ISBN 4-7561-01-62-3

[5] Ruby Home Page まつもと ゆきひろ
http://helium.ruby-lang.org/ja/index.html

付録

EBM の登録プログラムフローチャート

EBM の ABM 機能を利用して登録には touroku.cgi・kanryou.cgi を使用する。touroku.cgi のプログラムのフローチャートをここに示す。

touroku.cgi のフローチャート

機能 ： ABM を利用して URL を登録してもらう、EBM の登録フォーム

EBM の登録プログラム

EBM の ABM 機能を利用して登録には touroku.cgi・kanryou.cgi を使用する。touroku.cgi・kanryou.cgi のプログラムをここに示す

touroku.cgi のプログラム

言語 ： awk
機能 ： ABM を利用して URL を登録してもらう、EBM の登録フォーム

```
#!/bin/awk -f
BEGIN{
usr=ENVIRON["REMOTE_IDENT"]
day=ENVIRON["QUERY_STRING"]

#ログ指定
if (day==0){
  alog="../log.dat"
    dday="今日"
    aday=1
}


if (day==1){
  alog="../log1.dat"
    dday="昨日"
    aday=1
}


if (day==2){
  alog="../log2.dat"
    dday="2 日前"
    aday=1
}


if (day==3){
  alog="../log3.dat"
    dday="3 日前"
    aday=1
}


if (day==4){
```

```
  alog="../log4.dat"
    dday="4 日前"
    aday=1
}

if (day==5){
  alog="../log5.dat"
    dday="5 日前"
    aday=1
}

#HTML 部分
print "Content-type: text/html\n"
print "<HTML>"
print "<head><title>登録フォーム</title>"
print "</head>"
print "<body bgcolor=lightcyan>"
print usr"さん EBM へ登録ありがとうございます。<BR>"
print dday"の履歴です。他の日の閲覧した URL を登録したい場合は<BR>"
print "こちらよりお選びください。<BR>"

if(day!=0){
  print "<a href=touroku01.cgi?0>今日 </a>"
}

if(day!=1){
  print "<a href=touroku01.cgi?1>昨日 </a>"
}

if(day!=2){
  print "<a href=touroku01.cgi?2>2 日前 </a>"
}

if(day!=3){
  print "<a href=touroku01.cgi?3>3 日前 </a>"
}

if(day!=4){
  print "<a href=touroku01.cgi?4>4 日前 </a>"
}

if(day!=5){
  print "<a href=touroku01.cgi?5>5 日前 </a>"
```

```
}

print "<BR>"

#ログの読み込み
while(getline < alog){
  if ($1==usr){
    if($2 !~ ".gif")
      acc[$2] += aday
}
}

#HTML 部分
print "<form action=kanryou01.cgi method="post">"
print "<input type=hidden name=usr value="usr"><br>"
print "<p><input type=submit value=登録>"
print "<select name=menu>"
print  "<option value=a>ジャンルを選んでください</option>"
print  "<option value=1>検索エンジン</option>"
print  "<option value=2>社会・経済</option>"
print  "<option value=3>コンピュータ関係</option>"
print  "<option value=4>ソフト・ハード</option>"
print  "<option value=5>音楽・映画関係</option>"
print  "<option value=6>ゲーム関係</option>"
print  "<option value=7>ショッピング</option>"
print  "<option value=1024>その他</option>"
print "</select><br>"
print "<input type=text name=txt size=25 value=その他の場合はご記入ください><br>"
print "登録の際には下記より登録したい URL 上のボタンをクリックしてから<br>上記の登録ボタンをお
してください。</p>"

#usr のログを表示する
print "</p>閲覧回数<br>"
for (na in acc){
  print acc[na],"回  ","<input type=radio name=URL value="na"><br><a href="na" target=_blank>",na "</A><B
}
#HTML 部分
print "</form>"
print "</HTML>"
}
```

kanryou.cgi のプログラム

言語 ： Ruby
機能 ： touroku.cgi で登録してもらった URL を書き込み保存する。

```
#!/home/kanaya/bin/ruby
print "Content-type: text/html\n\n"
require "cgi-lib"
input = CGI.new
require "kconv"

#初期設定
CR = "\015"
LF = "\012"
EOL = CR + LF
BR = "<br>"
FILENAME = "/tmp/EBM01.log"
LOCKFILENAME = "/tmp/EBM01.log.lock"
MAXLOG = 30
$log = []
a = 0

#ファイルロッククラス作成
class FileLockDir
  def lock(lockdir)
    begin
      Dir.mkdir(lockdir)
      lockflag = true
    rescue Errno::EEXIST
      lockflag = false
    end
    return(lockflag)
  end

  def unlock(lockdir)
    begin
      Dir.rmdir(lockdir)
      lockflag = true
    rescue
      lockflag = false
    end
    return(lockflag)
  end
end
```

```
#ログファイルの読みこみ関数
def read_log_file
  c = 0
  begin
    fh = open(FILENAME)
  rescue Errno::ENOENT
    return
  end

  fh.each{|l|
    $log[c] = l
    c += 1
    if c >= MAXLOG
      break
    end
  }
  fh.close
end

#ログファイルの書き込み関数
def write_log_file(writedata,a,b,c)
  $log.unshift("#{writedata} #{a} #{b} #{c}\n")
  c = 0
  fh = open(FILENAME, "w")
  $log.each{|l|
    fh.print l
    c += 1
    if c >= MAXLOG
      break
    end
  }
  fh.close
end

#ファイルロッククラスの初期化
fillock = FileLockDir.new

#フォームからのデータ読みこみと日本語変換と改行の変換
if input["URL"]
  URL = Kconv.toeuc(input["URL"])
  URL.gsub!(/#{EOL}/, BR)
  URL.gsub!(/#{CR}/, BR)
  URL.gsub!(/#{LF}/, BR)
```

```
end

if input["menu"]
  menu = Kconv.toeuc(input["menu"])
  menu.gsub!(/#{EOL}/, BR)
  menu.gsub!(/#{CR}/, BR)
  menu.gsub!(/#{LF}/, BR)
end

if input["txt"]
  txt = Kconv.toeuc(input["txt"])
  txt.gsub!(/#{EOL}/, BR)
  txt.gsub!(/#{CR}/, BR)
  txt.gsub!(/#{LF}/, BR)
end

if input["usr"]
  usr = Kconv.toeuc(input["usr"])
  usr.gsub!(/#{EOL}/, BR)
  usr.gsub!(/#{CR}/, BR)
  usr.gsub!(/#{LF}/, BR)
end

#ログファイルの読みこみ関数を呼び出す
read_log_file

#書き込みがあったら
if URL
  begin

#ファイルロック設定
    lockd = fillock.lock(LOCKFILENAME)

    unless lockd
      print <<EOF;
<html><body>
他の人が書き込み中です。
しばらく待ってからもう一度書いて下さい。
</body></html>
EOF
    end

#ログファイルの書き込み関数を呼び出し
```

```
    write_log_file(URL,menu,usr,txt)


  ensure
#ファイルロック解除
    fillock.unlock(LOCKFILENAME) if lockd
  end
end


#HTML の表示
if menu == "a"
print <<EOF;
<html><body>
<font color="red" size="8">ジャンルを選択されていないので、登録されませんでした。
<br>もう一度登録しなおしてください。</font><br>
<br>
<br>
<p><a href="http://www.fken/~kanaya/html/touroku01.html">
EBM 登録ページへ</a></p>
</body></html>
EOF

elsif
print <<EOF;
<html><body>
<font color="red" size="6">登録ありがとうございます。</font><br>
ですが、まだ試作段階なので登録させた URL は
後日には消えております。<br>
</body></html>
EOF
end



#HTML の表示
print <<EOF;
<html>
<head>
<meta http-equiv="Content-Type" content="text/html">
<title>登録終了です</title>
</head>
<body bgcolor="skyblue">
```

```
<br>
EOF

$log.each{|l|
  print "<a href=#{l}>#{l}</a>\n<hr>\n"
}

print <<EOF;
<br>
<a href="http://www.fken/%7Ekanaya/html/ebm.html">EBM トップページへ</a><br>
</body>
</html>
EOF
```