

ソフトウェアサイエンス実験 S11 最終レポート

200911434 青木大祐

2013年3月6日

この実験に際して、情報科学類の Web サーバである www.coins.tsukuba.ac.jp を用いて実習を行った。PHP はサーバにインストールされていた 5.2.17 を、同じく Perl についてもサーバにインストールされていた 5.8.9 を使用した。

1 課題 1 WWW 上におけるメディア情報検索インターフェイスの作成

Perl で実装されたサンプルをもとに、PHP を利用して検索エンジンの実装を行った。

1.1 WWW 検索エンジンの作成、データ登録機能の追加

登録されているデータの表示を行う index.php の実装は以下のとおり。

ソースコード 1 index.php

```
<!doctype html public "-//W3C//DTD HTML 4.01 Transitional//EN">
<html lang="JA">
<head>
<meta http_equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>メディア情報検索の基礎 課題1-1 sample</title>
</head>
<body bgcolor="white" text="black">

<h1 align="center">メディア情報検索の基礎 課題1-1</h1>
<hr align="center" noshade width="80%"/>
<div align="center">

<script type="text/javascript">
    function search() {
        var query = window.document.getElementById("form").value;
        window.open("./search.php?query=" + query);
    }
</script>

<form>
<input type="text" name="KEYWORD" size="15" id="form" />
</form>
<button type="button" name="button" value="button" onClick="search()">search</button>

</div>
<hr align="center" noshade width="80%"/>

<caption>検索対象データ</caption>

<?php
print("<table border=\"1\" cellPadding=\"5\" align=\"center\">");
$fileName = "target.txt";
$file = fopen($fileName, "r");

while ($line = fgetcsv($file)) {
    $output = "";
    $flag = false;
    $output .= "<tr>";

    foreach($line as $data) {
        $output .= "<td>";
        $output .= $data;
        $output .= "</td>";
    }

    $output .= "</tr>";
    print($output);
}

fclose($file);
?>

</table>
<form action="add.php" method="post">
    追加: <input type="text" name="content" />
    <input type="submit" />
</form>
<hr align="center" noshade width="80%"/>
<div align="center">
```

```

<a href="../text.html">戻る</a>
</div>
</body>
</html>

```

fgetcsv() 関数を使って登録されているデータを読み出し、テーブルとしてページ内に表示している。

図 1 index.php

The screenshot shows a web page titled "メディア情報検索の基礎 課題1-1". At the top is a search bar with a "search" button. Below it is a section labeled "検索対象データ" containing a 3x4 grid table:

ほげ	fuga	puyo	ふが
aa	a		aa
あいうえお	hoge	fuga	びよ

Below the table is a "追加:" input field and a "送信" button. At the bottom right is a "戻る" link.

また、データの新規登録には add.php を、データの検索には search.php をそれぞれ呼び出している。各部分の実装は以下のとおり。

ソースコード 2 add.php

```

<?php
$fh = fopen("target.txt", "a");
fwrite($fh, htmlspecialchars($_POST['content'] . "\n"));
fclose($fh);
header("HTTP/1.1 301 Moved Permanently");
header("Location: ./index.php");
?>

```

add.php では、与えられたクエリを target.txt に書き込み、もとの index.php にリダイレクトするという処理を行なっている。

ソースコード 3 search.php

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title></title>
  </head><body>
<style type="text/css">
table, td, th { border: 1px #1b1b1b solid ;
                border-collapse: collapse ; }
</style>

<?php
  $fileName = "target.txt";
$file = fopen($fileName, "r");
print("query is ");
print($_GET['query']);
print("<table border=\"1\" cellPadding=\"5\">");
while ($line = fgetcsv($file)) {
  $output = "";
  $flag = false;
  $output .= "<tr>";
  foreach($line as $data) {
    if (preg_match("/". $_GET['query'] . "/i", $data) == 1) {
      $flag = true;
      $output .= "<td bgcolor=\"#FF0000\">";
      $output .= "<b><i>";
      $output .= $data;
      $output .= "</i></b>";
    } else {
      $output .= "<td>";
    }
  }
  $output .= "</tr>";
}
print($output);
print("</table>");
?>

```

```

        $output .= $data;
    }
    $output .= "</td>";
}
$output .= "</tr>";
if ($flag) {
    print($output);
}

}
print("</table>");
fclose($file);
?>
</body>
</html>

```

search.php では与えられたクエリを登録されているデータの中から探し出し、ヒットしたカラムに色を付けて結果を表示している。

図 2 検索結果

query is fuga			
ほげ	fuga	puyo	ふが
あいうえお	hoge	fuga	ぴよ

この課題については、<http://www.coins.tsukuba.ac.jp/~s0911434/jikken-s11/1-1/> にアクセスすることで実際の動きを確かめることができる。

2 メディア情報検索システムの構成

課題 2 では、検索対象のメディアと類似度の高いものを検索するエンジンを実装した。

2.1 類似画像検索システムの実現

課題 2-1 では、予め用意されている色情報を用いて類似度の高い画像を検索する実験を行った。index.php では、以下のようにして登録されている画像を列挙している。

ソースコード 4 index.php

```

<!doctype html public "-//W3C//DTD HTML 4.01 Transitional//EN">
<html lang="JA">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>メディア情報検索の基礎 課題2-1 sample</title>
</head>

<body bgcolor="white" text="black">

<h1 align="center">メディア情報検索の基礎 課題2-1</h1>
<hr align="center" noshade width="80%"/>
<div align="center">

<script type="text/javascript">
    function search() {
        var query = window.document.getElementById("form").value;
        window.open("./search.php?query=" + query);
    }
</script>

```

```

<form>
<input type="text" name="KEYWORD" size="15" id="form" />
</form>
<button type="button" name="button" value="button" onClick="search()>search</button>
</div>
<hr align="center" noshade width="80%"/>

<div align="center">
<caption>検索対象データ</caption>

<?php
print("<table border=\"1\" cellPadding=\"5\" align=\"center\">");
$fileName = "resource/2_1_data.csv";
$file = fopen($fileName, "r");

while ($line = fgetcsv($file)) {
    $output = "";
    $flag = false;
    $output .= "<tr>";

    foreach($line as $data) {
        $output .= "<td>";
        if (preg_match("/^http/", $data)) {
            $output .= "<img src=\"$data\">";
        } else {
            $output .= $data;
        }
        $output .= "</td>";
    }

    $output .= "</tr>";
    print($output);
}

fclose($file);
?>

</table>
<form action="add.php" method="post">
追加: <input type="text" name="content" />
<input type="submit" />
</form>
</div>
<hr align="center" noshade width="80%"/>

<div align="center">
<a href="../text.html">戻る</a>
</div>
</body>
</html>

```

このページは以下のように表示される。

図3 index

メディア情報検索の基礎 課題2-1				
<input style="width: 100%; height: 30px; margin-bottom: 5px;" type="text"/> <input style="width: 100px; height: 30px;" type="button" value="search"/>				
検索対象データ				
Image1	0.30	0.32	0.38	
Image2	0.37	0.37	0.26	
Image3	0.42	0.30	0.28	

また、ページ下部のフォームから add.php にアクセスすることで、任意の画像 URL を追加できる。追加データは「名前, R, G, B, url」のフォーマットで記述する。

ソースコード 5 add.php

```
<?php
$fh = fopen("resource/2_1_data.csv", "a");
fwrite($fh, htmlspecialchars($_POST['content']) . "\n");
fclose($fh);
header("HTTP/1.1 301 Moved Permanently");
header("Location: ./index.php");
?>
```

類似度の計算は、画像の色情報を RGB の三次元ベクトルとして、内積の大きいものを類似度が高いと判定している。実際に search.php を用いて類似度の高い順に画像を並べ替えている。

図 4 検索結果

query is Image3																
Image3	0.42	0.30	0.28													
<hr/>																
<table border="1"> <thead> <tr> <th>id</th><th>Similarity</th></tr> </thead> <tbody> <tr><td>Image3</td><td>0.3448</td></tr> <tr><td>Image2</td><td>0.3392</td></tr> <tr><td>Image5</td><td>0.331</td></tr> <tr><td>Image1</td><td>0.3284</td></tr> <tr><td>Image4</td><td>0.2956</td></tr> </tbody> </table>					id	Similarity	Image3	0.3448	Image2	0.3392	Image5	0.331	Image1	0.3284	Image4	0.2956
id	Similarity															
Image3	0.3448															
Image2	0.3392															
Image5	0.331															
Image1	0.3284															
Image4	0.2956															

search.php の実装は以下のとおり。

ソースコード 6 search.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title></title>
    </head><body>
<style type="text/css">
table, td, th { border: 1px #1b1b1b solid ;
                border-collapse: collapse ; }
</style>

<?php
  $fileName = "resource/2_1_data.csv";
$file = fopen($fileName, "r");
print("query is ");
print($_GET['query']);
print("<table border=\"1\" cellpadding=\"5\">");
while ($line = fgetcsv($file)) {
  $output = "";
  $flag = false;
  $output .= "<tr>";

  $tag = $line[0];
  if (preg_match("/" . $_GET['query'] . "/i", $tag) == 1) {
    $target = $line;
    foreach ($line as $data) {
      $output .= "<td>";
      if (preg_match("/^http/", $data)) {
        $output .= "<img src=\"" . $data. "\">";
      } else {
        $output .= $data;
      }
    }
  }
}
```

```

        }
        $output .= "</td>";
    }
    print $output . "</tr>";
}
print("</table>");
fclose($file);

print"<hr>";

$fileName = "resource/2_1_data.csv";
$file = fopen($fileName, "r");
while ($current = fgetcsv($file)) {
    $similarity = ($target[1] * $current[1]) + ($target[2] * $current[2]) + ($target[3] * $current[3]);
    $ans[$current[0]] = $similarity;
}
array_multisort($ans, SORT_NUMERIC);
$ans = array_reverse($ans);
print "<table>";
print "<tr><th>id</th><th>Similarity</th></tr>";
foreach ($ans as $key => $value) {
    print "<tr><td>" . $key . "</td><td>" . $value . "</td></tr>";
}
print "</table>";

?>
</body>
</html>

```

2.2 メタデータの自動抽出

課題 2-1において予め用意されていた色情報データを、画像の登録時に自動的に生成することを考える。add.php の挙動を変更し、以下の様な処理を追加する。

1. 画像の URL を受け取る
2. wget コマンドを用いて画像をダウンロードする
3. ImageMagick の convert コマンドを用いて ppm 形式に変換する
4. ppm ファイルを読み込み、色情報データを抽出する
5. 抽出した色情報データを登録データとして書き込む

以上の操作を行う Perl スクリプトを用意し、add.php から呼び出すことで、自動的なメタデータの作成が可能になった。実装は以下のとおり。

ソースコード 7 add.php

```

<?php
/* $fh = fopen("resource/2_1_data.csv", "a"); */
$s = $_POST['content'];
exec("./add_list.pl " . escapeshellarg($s) . " ADD", $output, $ret);
print var_dump("/home/ugrad/09/s0911434/public_html/jikken-s11/2-2/add_list.pl " . escapeshellarg($s));
print "<HR>";
print var_dump($output);
print "<HR>";
print $ret;

header("HTTP/1.1 301 Moved Permanently");
header("Location: ./index.php");

?>

```

OS コマンドインジェクション攻撃を回避するため、escapeshellargs 関数を用いて危険な文字をエスケープし、Perl スクリプトに渡している。Perl スクリプトは検索の際にも用いるため、第二引数としてコマンドの”ADD”を渡し、動作を制御している。

ソースコード 8 add_list.pl

```

#!/usr/bin/perl
use strict;
use warnings;
use Data::Dumper;

```

```

my $url = shift;
chomp $url;

my $mode = shift;
my $res = "./resource";

mkdir $res unless -d $res;
chdir $res;

my $filename = 'basename $url';
chomp $filename;
unlink $filename if -f $filename;
system "/usr/local3/bin/wget $url";
-f $filename or exit(17);

my $ppm = $filename . ".ppm";
unlink $ppm if -f $ppm;
system "/usr/local3/bin/convert $filename $ppm";

open my $bin, "<", $ppm or exit(22);
binmode $bin;

my %header;
my $c;
my $offset = 0;
while(read($bin, $c, 1)) {
    last if ((chr unpack 'c1', $c) =~ /\n/);
    $header{magic} .= chr unpack 'c1', $c;
}

while(read($bin, $c, 1)) {
    last if ((chr unpack 'c1', $c) =~ /\n/);
    $header{size} .= chr unpack 'c1', $c;
}

while(read($bin, $c, 1)) {
    last if ((chr unpack 'c1', $c) =~ /\n/);
    $header{max} .= chr unpack 'c1', $c;
}

my @color;
my $idx = 0;

while(1) {
    if (read($bin, $c, 1)) {
        my $px = [unpack 'C1', $c];
        read($bin, $c, 1);
        push(@$px, unpack 'C1', $c);
        read($bin, $c, 1);
        push(@$px, unpack 'C1', $c);
        push(@color, $px);
    } else {
        last;
    }
}
close $bin;

my %rgb;
for my $px (@color) {
    $rgb{red} += $$px[0];
    $rgb{green} += $$px[1];
    $rgb{blue} += $$px[2];
}

my $total = $rgb{red} + $rgb{green} + $rgb{blue};

eval {
    if ($mode eq "ADD") {
        open my $list, ">>", "./2_1_data.csv";
        printf $list "%s, %03f, %03f, %03f, %s\n", $filename, $rgb{red} / $total, $rgb{green} / $total, $rgb{blue} /
        $total, $url;
        close $list;
    } elsif ($mode eq "SEARCH") {
        printf "%s, %03f, %03f, %03f, %s\n", $filename, $rgb{red} / $total, $rgb{green} / $total, $rgb{blue} /
        $total, $url;
    }
};

if (@_) {
    exit(83);
}

```

実際にこのスクリプトが動作する www サーバは特殊なディレクトリ構造をしているため、wget や convert などのコマンドパスは決め打ちになっている。

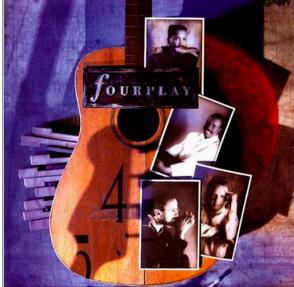
新しく実装したこれらの機能を利用することで、簡単に検索対象となる画像を追加できるようになった。実際に画像を追加し、index に反映されているスクリーンショットを以下に示す。

図 5 追加された画像

メディア情報検索の基礎 課題2-2				
<input type="button" value="search"/> 検索対象データ				
h_logo.gif	0.333340	0.311825	0.354835	 筑波大学 University of Tsukuba
mona_lisa.jpg	0.394850	0.366399	0.238752	
starry_night.jpg	0.320221	0.320350	0.359428	
image3.jpg	0.334399	0.357600	0.308001	
chrome_logo_2x.png	0.340953	0.337238	0.321810	
				

また、search.php から Perl スクリプトに"SEARCH" コマンドを渡して呼び出すことで、引数に指定した URL の画像から色情報データを抽出し、登録済みの画像から類似度の高いものを表示する事ができる。実際に検索しているスクリーンショットを以下に示す。筑波大学のロゴマークに近い色合いの画像が上位に来ているのがわかる。

図 6 検索結果

ID:h_logo.gif  筑波大学 University of Tsukuba		
id	Similarity	
fourplay.jpg	0.33520650702	
starry_night.jpg	0.33417324127	
		

search.php の実装は以下のとおり。Perl スクリプトから帰ってきた色情報データと既存の画像を比較し、類似度の高い順に並び替えて表示している。

ソースコード 9 search.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title></title>
    </head><body>
      <style type="text/css">
        table, td, th { border: 1px #1b1b1b solid ;
          border-collapse: collapse ; }
      </style>

      <?php
      $s = urldecode($_GET['query']);
      exec("./add_list.pl " . escapeshellarg($s) . " SEARCH", $output, $ret);
      $target = str_getcsv($output[0]);
      print "ID:" . $target[0] . "<br>";
      print "<img src=\"\" . $s . \"\">";

      print "<hr>";

      $fileName = "resource/2_1_data.csv";
      $file = fopen($fileName, "r");
      while ($current = fgetcsv($file)) {
        if (!strcmp($target[0], $current[0])) { continue; }
        $similarity = ($target[1] * $current[1]) + ($target[2] * $current[2]) + ($target[3] * $current[3]);
        $ans[$current[0]] = $similarity;
      }
      array_multisort($ans, SORT_NUMERIC);
      $ans = array_reverse($ans);
      print "<table>";
      print "<tr><th>id</th><th>Similarity</th><th>Image</th></tr>";
      foreach ($ans as $key => $value) {
        print "<tr><td>" . $key . "</td><td>" . $value . "</td><td><img src=\"./resource/" . $key . "\"></td></tr>";
      }
      print "</table>";

    ?>
  </body>
</html>
```

<http://www.coins.tsukuba.ac.jp/~s0911434/jikken-s11/2-2/> にアクセスすることで、これらが実際に機能しているページを触ることが出来る。

2.3 メディア情報検索システムの実現

課題 2-3 では今までとは異なり画像ではないメディアの類似度を計量し、同じように検索を行えるようなシステムを実装した。

この課題で対象とするメディアは、英文などローマ字のアルファベットからなるテキストファイルである。単語の出現する頻度をベクトルとして表現し、コサイン類似度を計算することで、それぞれの文書の傾向から類似度が計算できる。コサイン類似度の計算については「コサイン尺度（コサイン類似度）の計算^{*1}」を参考にし、類似度の計算部分は同ページに掲載されているサンプルを利用した。

また、出現頻度が高く、かつ特徴として意味を成さない助詞や冠詞などの“stop words”は「Stop Words^{*2}」のリストを利用し、これらを抽出時に除外した。単語の切り出しや集計などは Perl で行い、算出したベクトルはテキストファイルとして保存している。

英文テキストのコサイン類似度を計算するために実装した Perl スクリプトを以下に示す。

*1 <http://private.ceek.jp/archives/003891.html>

*2 <http://www.webconfs.com/stop-words.php>

```

#!/usr/bin/perl

use strict;
use warnings;
use Data::Dumper;

our $WRITE = 1;
our @STOP_WORDS;

open my $fh, "<", "stop_words.txt";

while (<$fh>) {
    push @STOP_WORDS, $_;
}
close $fh;

sub extract_words {
    my $title = shift;
    my $url = shift;
    my $text = shift;
    my @words = split /[^a-zA-Z]+/, $text;
    my @nonstop = grep {
        my $ret = 1;
        for my $sw (@STOP_WORDS) {
            if ($sw =~ /$_/i) {
                $ret = 0;
                last;
            }
        }
        $ret;
    } @words;
}

my %all;
for my $w (@nonstop) {
    $w =~ tr/A-Z/a-z/;
    if (exists $all{$w}) {
        $all{$w]++;
    } else {
        $all{$w} = 1;
    }
}

my @array = %all;

if ($WRITE) {
    open my $fh, ">", "vector.txt";
    print $fh $title . ", ";
    print $fh $url . ", ";
    print $fh join ", ", @array;
    print $fh "\n";
}

return {title => $title, url => $url, vector => \%all};
}

sub cosine_similarity {
    my ($vector_1, $vector_2) = @_;

    my $inner_product = 0.0;
    map {
        if ($vector_2->{$_}) {
            $inner_product += $vector_1->{$_} * $vector_2->{$_};
        }
    } keys %{$vector_1};

    my $norm_1 = 0.0;
    map {
        $norm_1 += $_ ** 2
    } values %{$vector_1};
    $norm_1 = sqrt($norm_1);

    my $norm_2 = 0.0;
    map {
        $norm_2 += $_ ** 2
    } values %{$vector_2};
    $norm_2 = sqrt($norm_2);

    return ($norm_1 && $norm_2) ? $inner_product / ($norm_1 * $norm_2) : 0.0;
}

sub read_vector {
    open my $fh, "<", "vector.txt";
    my @vector_list;
    while (<$fh>) {
        my @line = split /\s*,\s*/, $_;
        my $title = shift @line;
        my $url = shift @line;
        my %hash = @line;
        push @vector_list, {title => $title, url => $url, vector => \%hash};
    }
}

```

```

        }
        close $fh;
        return @vector_list;
    }

my $url = shift;
my $mode = shift;
my $res = "./resource";

mkdir $res unless -d $res;
chdir $res;

my $filename = 'basename $url';
chomp $filename;
unlink $filename if -f $filename;
system "/usr/local3/bin/wget $url";
-f $filename or exit(17);

my $text = "";
open $fh, "<", $filename;
while(<$fh>) {
    $text .= $_;
}
close $filename;

if ($mode eq "ADD") {
    $WRITE = 1;
    my $data = extract_words($filename, $url, $text);
} elsif ($mode eq "SEARCH") {
    $WRITE = 0;
    my @vector_list = read_vector();
    my $data = extract_words($filename, $url, $text);

    my @similarity;
    for my $elem (@vector_list) {
        push @similarity, {similarity => cosine_similarity($elem->{vector}, $data->{vector}), target => $elem};
    }

    my @sorted = sort { $b->{similarity} <=> $a->{similarity}} @similarity;
    for my $sim (@sorted) {
        print "<tr><td><a href=\"". $sim->{target}{url} . "\">" . $sim->{target}{title} . "</a></td><td>" .
            $sim->{similarity} . "</td></tr>\n";
    }
}
}

```

実際に RFC や日本国憲法の英訳、推理小説、ソースコード、Perl6 の仕様書など様々なドキュメントを登録し、類似度を判定した。

無作為に選んだ RFC ドキュメントを元に類似度順に並べた結果を以下に示す。概ね登録済みの RFC ドキュメントが高い類似度だと判定されている。

図 7 検索結果 1:適当に選んだ RFC

search with http://www.rfc-editor.org/rfc/rfc6353.txt	
document	similarity
rfc4222.txt	0.143450267783719
S02-bits.pod	0.122604155120614
rfc4621.txt	0.0733725046403186
gpl.txt	0.068233696066404
LICENSE.Artistic2.0.txt	0.0383660957221128
index.html	0.0232284665465225
ch1.html	0.0145119686644232
domino.pl	0.0110843440966891
fle_e.txt	0.01033615259275
morgue_e.txt	0.00304402090762064

また、別の例として、登録済みの Perl6 の仕様書の別のセクションを与えた場合の結果を以下に示す。登録済みの仕様書の類似度が非常に高く、また続く 2 つ目のソースコードも Perl6 で記述されているものである。これらの結果から、この類似度判定はそれなりの精度を出せていることが分かる。

図 8 検索結果 2:仕様書の別セクション

search with https://raw.github.com/perl6/specs/master/S06-routines.pod	
document	similarity
S02-bits.pod	0.676482851898842
domino.pl	0.193817365859662
gpl.txt	0.0894317225169207
rfc4222.txt	0.0671462675088
LICENSE.Artistic2.0.txt	0.0545317651847087
rfc4621.txt	0.0335572772415045
ch1.html	0.0234770773127775
index.html	0.020529326703036
morgue.e.txt	0.00698841759995854
file.e.txt	0.00670712465706997

<http://www.coins.tsukuba.ac.jp/~s0911434/jikken-s11/2-3/> にアクセスすることで、この類似度判定が実際に動作しているページを触ることが出来る。

3 メディア情報検索における数理的モデリングの基礎

課題 3 では、ページランクに基づく検索結果の重み付けについて実験を行った。

3.1 PageRank の算出

ページランクを算出するための仮想的なドキュメントの集合として、次のようなグラフ構造を題材にした。

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \quad (1)$$

このグラフ構造からページランクを算出する Perl スクリプトを以下に示す。入力フォーマットは列の先頭に ID とドキュメント名のついた CSV からなるテキストファイルである。

ソースコード 10 pagerank.pl

```
#!/usr/bin/perl
use strict;
use warnings;
my $delta = 1 / 100;# %

my @url_list = ();
my $matrix = [];

open my $fh, "<", shift or die "Failed to open data file.";

while(<$fh>) {
    my @column = split ',';
    my $idx = shift @column; # index
    my $url = shift @column; # url
    $url_list[$idx] = $url;
    my $j = 0;
    map {
        $matrix->[$j]->[$idx] = $idx == $j ? 0 : $_;
        $j++;
    } @column;
}
```

```

}

close $fh;

matrix_normalize($matrix);

for (1..100) {
    $matrix = matrix_double($matrix);
    last if is_converged($matrix);
}

map {
    printf "%d,%s,%f\n", $_, $url_list[$_], $matrix->[$_-]>[0];
} 0..(scalar @{$matrix} - 1);

my $prev_matrix = undef;
sub is_converged {
    my $matrix = shift;
    unless( $prev_matrix ) {
        $prev_matrix = $matrix;
        return 0;
    }
    map {
        my $p = $prev_matrix->[$_-]>[0];
        next if $p == 0;
        my $d = ( $p - $matrix->[$_-]>[0] ) / $p;
        if( $delta < abs( $d ) ) {
            $prev_matrix = $matrix;
            return 0;
        }
    } 0..( scalar @{$matrix} - 1 );
    return 1;
}

sub matrix_normalize {
    my $matrix = shift;

    my @row_total = ();
    my $i = 0;
    map {
        my $j = 0;
        map {
            $row_total[$j] += $_;
            $j++;
        } @_;
        $i++;
    } @{$matrix};

    $i = 0;
    map {
        die "\nPage:$i has no outer link\n" if $_ == 0;
        $i++;
    } @row_total;
}

$i = 0;
map {
    my $matrix_i = $_;
    my $j = 0;
    map {
        $matrix_i->[$j] /= $row_total[$j];
        $j++;
    } @{$matrix_i};
    $i++;
} @{$matrix};

sub matrix_double {
    my $matrix = shift;
    my $ret = [];

    my $i = 0;
    map {
        my $matrix_i = $_;
        my $j = 0;
        map {
            my $sum = 0;
            map {
                $sum += $matrix_i->[$_] * $matrix->[$_-]>[$j];
            } 0..( scalar @{$matrix_i} - 1 );
            $ret->[$i]->[$j] = $sum;
        } @{$matrix_i};
        $i++;
    } @{$matrix};
    matrix_normalize( $ret );
    $ret;
}

```

```
}
```

実行結果として得られたページランクを以下に示す。

1. 0.238318
2. 0.093458
3. 0.070093
4. 0.126168
5. 0.191589
6. 0.065421
7. 0.214953

3.2 PageRank を用いた Web 検索システムの構築

課題 3-1 で算出した PageRank を用いて、検索結果の順序を決定するように実装を行った。検索対象のドキュメントとしては、ある程度同じ単語が含まれるであろう RFC を適当に選び、登録を行った。

検索の処理を行う search.php の実装を以下に示す。今回はドキュメントの追加などは考えないため、ソースコード中にページランクを直接記述してある。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title></title>
</head><body>
<style type="text/css">
table, td, th { border: 1px #1b1b1b solid ;
border-collapse: collapse ; }
</style>
<?php

$list = array("0_rfc6879.txt" => 0.238318,
"1_rfc6840.txt" => 0.093458,
"2_rfc6396.txt" => 0.070093,
"3_rfc5752.txt" => 0.126168,
"4_rfc5698.txt" => 0.191589,
"5_rfc5664.txt" => 0.065421,
"6_rfc5619.txt" => 0.214953);

print("query is ");
print($_GET['query']);

foreach($list as $doc => $pagerank) {
    $fp = fopen("documents/" . $doc, "r");
    $text = "";
    while(!feof($fp)) {
        $text .= fgets($fp);
    }
    fclose($fp);

    if (preg_match("/" . $_GET['query'] . "/i", $text)) {
        $match[$doc] = $pagerank;
    }
}

if (count($match) == 0) {
    print "一致する検索結果はありませんでした。";
} else {
    array_multisort($match, SORT_NUMERIC);
    $ret = array_reverse($match);
    print("<table>");
    print "<tr><th>entry</th><th>pagerank</th></tr>";
    foreach ($ret as $doc => $pagerank) {
        print "<tr><td><a href=\"documents/" . $doc . "\">" . $doc . "</a></td><td>" . $pagerank . "</td></tr>";
    }
    print("</table>");

?>
</body>
</html>
```

検索クエリを受け取ると、登録されているドキュメントを順に読み込んでマッチングを行い、ヒットしたものについては記録しておく。すべてのドキュメントの検索が終了したら、ヒットしたドキュメントをページランクに基いて並べ直し、検索結果として表示するという処理を行なっている。

実際に検索を行った結果を以下に示す。

図9 検索結果 1:キーワード「RFC」

query is RFC	
entry	pagerank
0 rfc6879.txt	0.238318
6 rfc5619.txt	0.214953
4 rfc5698.txt	0.191589
3 rfc5752.txt	0.126168
1 rfc6840.txt	0.093458
2 rfc6396.txt	0.070093
5 rfc5664.txt	0.065421

すべてのドキュメントが検索にヒットしており、検索結果はページランクに基いて並べられていることが分かる。

図10 検索結果 1:キーワード「algorithm」

query is algorithm	
entry	pagerank
4 rfc5698.txt	0.191589
3 rfc5752.txt	0.126168
1 rfc6840.txt	0.093458
5 rfc5664.txt	0.065421

7つのうち4つのドキュメントに“algorithm”という単語が含まれている。これらもまたページランクに基いて並べられている。

以上、ページランクを活用して検索結果を表示する機能が実装できた。このページは <http://www.coins.tsukuba.ac.jp/~s0911434/jikken-s11/3/> にてアクセスできる。

4 考察

4.1 類似度判定について

抽出方法がまづかったのか、RGBによる内積はあまり精度が高いとは言えない結果になってしまった。多くの画像は全体的に各色が3.3に近い値をとる傾向にあるため、あまり大きな差が出なかったのが原因だと考えられる。全体に占める各色の割合だけでなく、明るさなども計算に入れることで精度をあげられるのではないだろうか。

一方、文章のコサイン類似度判定はとてもうまく行ったように思う。語幹の抽出などは行わずに三単現や過去形などの接尾辞変化などを無視して統計を取ったにもかかわらず、有意といえる差で類似度が判定できたので満足している。

4.2 感想

普段何気なく使っているGoogleなどの検索エンジンであるが、利用者がより簡単に目的とする情報にたどり着けるように、色々な工夫が成されていることが分かった。単純にリンク数の多いページのページランクが高いということもなく、とても良くできた仕組みだということが、実際に実装してみることで実感できたと思う。