スパイダリングを用いた楽曲検索システムの開発

06H091 藤田康孝

目次

1 **はじめに** 1

2 **基礎知識とシステム説明** 2

2.1 スパイダのメリット 2

2.2 データの抽出方法 3

2.3 データベース 3

2.4 URL エンコード 3

2.5 システム概要 4

2.6 開発環境 4

2.7 処理の流れ 5

2.8 内部処理 5

2.9 結果と考察 8

3 **まとめ** 12

3.1 今後の課題 13

付録 A ソースコード 15

A.1 input.html 15

A.2 main.php 16

概要

自分の欲しい情報が複数の検索サイトに渡ってあった場合、それらをひとつひとつ確認していき、データの集約を手作業でするのは非常に手間であり、時間の無駄である。そこでスパイダリングの技術を用いることによってその手間を少しでも省くことを考えた。今回はその中でも楽曲データの検索についてを考えた。多くの楽曲データの検索サイトでは数多くのデータを取り扱っている所が大半であるが、その中でも存在するデータと存在しないデータと言うものは存在している。また必要なものは楽曲データだけでサイトにある広告等は必要ではない場合がある。その情報を集約し、まとめることは容易では無い。そこで今回は楽曲データを集約し、結果を表示できるプログラムを開発することにした。本研究ではユーザ側からは検索したいワードを入力するだけである。あとはプログラムがユーザが入力したワードを基に各 Web サイトに正規表現を用いてデータを抽出し、処理が行われ表示される。本システムでは実際にアーティスト名での検索に成功した。それによって 1 つ 1 つのページを見るより手間と時間が省けた。現状の問題としてはアーティスト名での検索に対応していないこととユーザ側から表示方法を選べないことである。今後の課題として、問題点の改善と併せて利用する Web サイトを増やした時に正規表現を自動で行うように改良する。

1　はじめに

今日、インターネットでの情報の取得は容易となった。その中でも検索サイトを使って、自分の欲しい情報を検索し、情報を入手することができる。しかし自分の欲しい情報が複数の検索サイトに渡ってあった場合、それらをひとつひとつ確認していき、データの集約を手作業でするのは非常に手間であり、時間の無駄である。そこでスパイダリングの技術を用いることによってその手間を少しでも省くことを考えた。スパイダ (spider) とは、インターネットから様々な情報を取得するプログラムのこと。スパイダを用いることによって、複数の Web サイトにまたがって存在する情報を組み合わせることによって、データ蓄積や独自の検索システムを構築できるなどの様々なことが可能となる [1]。

今回はスパイダを用いて楽曲データの検索についてを考えた。多くの楽曲データの検索サイトでは数多くのデータを取り扱っている所が大半であるが、その中でも有るデータと無いデータと言うものは存在している。また必要なものは楽曲データだけでサイトにある広告等は必要ではない場合がある。その情報を集約し、まとめることは容易では無い。

そこで今回は楽曲データを集約し、結果を表示できるプログラムを開発することにした。用語とシステムの内容を説明し、結果と考察で実際の処理結果をまとめる。まとめでは本研究の内容などをまとめる。

2　基礎知識とシステム説明

この章では本研究で用いたものについての基礎知識、及びシステムの説明を行う。

2.1　スパイダのメリット

スパイダには以下のメリットがある。これにより手動で検索をしたときよりも手間や時間が大幅に節約できる。

2.1.1　リソースへのアクセス自動化

毎日見るサイト等がある場合、機械的な作業はプログラムに任せてしまい、興味のあるコンテンツだけが引き渡されれば、ユーザーの時間、手間を省くことが出来る。データの活用に時間を回せるようになる [2]。

2.1.2　バラバラに存在するデータの集約

Web サイト同士はどこかで継っているが、手作業で様々なサイトからデータを集約することは一苦労である。スパイダリングによって作業を自動化することで、情報源をまたがって集約することが出来る [2]。

2.1.3　他のフォーマットへの加工

1 度入手したデータは、ユーザーの必要に応じて加工、変形、再フォーマットすることが出来る [2]。

2.1.4　特定種類の情報を限定的に集約

検索を行う場合、最初に対象を限定しなければならないことがある。スパイダであれば自動的に対象を限定して検索を行い、その結果を集約できる [2]。

2.2 データの抽出方法

必要なデータを抽出するときに HTML[*1]から正規表現[*2]を用いる。実際に検索サイトから楽曲データを抽出する場合はさらに HTML の中から URL[*3]を抜き出しそこに接続してからデータを抽出することがある。その時相対 URL [*4]は絶対 URL [*5]に直さないと接続できない。

2.3 データベース

データベースは、データを集めて管理し、容易に検索や抽出などの再利用をできるようにしたもの。膨大なデータが存在していても整理・整頓されていなければデータを全て調べないとないらないが、それらを整理・整頓してあるだけでも役にはたつ。さらにデータベース管理システム (DBMS[*6]) があればユーザーがさらに快適にデータベースを利用できる。

2.4 URL エンコード

URL エンコードとは URI において使用禁止である値を使う際に行われる符号化の俗称である。URI に ASCII の非予約文字以外の文字データを扱う場合に「%xx」という形でコードを表記する [3]。

[*1] テキストなどでも可能

[*2] ここで言う正規表現はパターンマッチのこと。パターンマッチとは、データを検索する場合に、特定のパターンが出現するかどうか、またどこに出現するかを特定する手法のことである。

[*3] Uniform Resource Locator の略。
ネットワーク内の位置を示してリソースを同定する。

[*4] HTML を解析する際に/hoge/hoge.html といったプロトコル等がない URL のことを相対 URL という。

[*5] し http://www.example.com/hoge/hoge.html といった完全な URL を絶対 URL という

[*6] DataBase Management System の略
データベースを構築するために必要なデータ運用、管理するためのシステム及びそのソフトウェア

2.5 システム概要

システムとして入力したワードをプログラムに渡す。ワードが渡されればプログラム内で各 Web サイトにそのワードを渡して楽曲データを収集を行う。次にデータベース内で重複の処理を行い、ユーザに表示する。システムの概要を図 1 に示す。

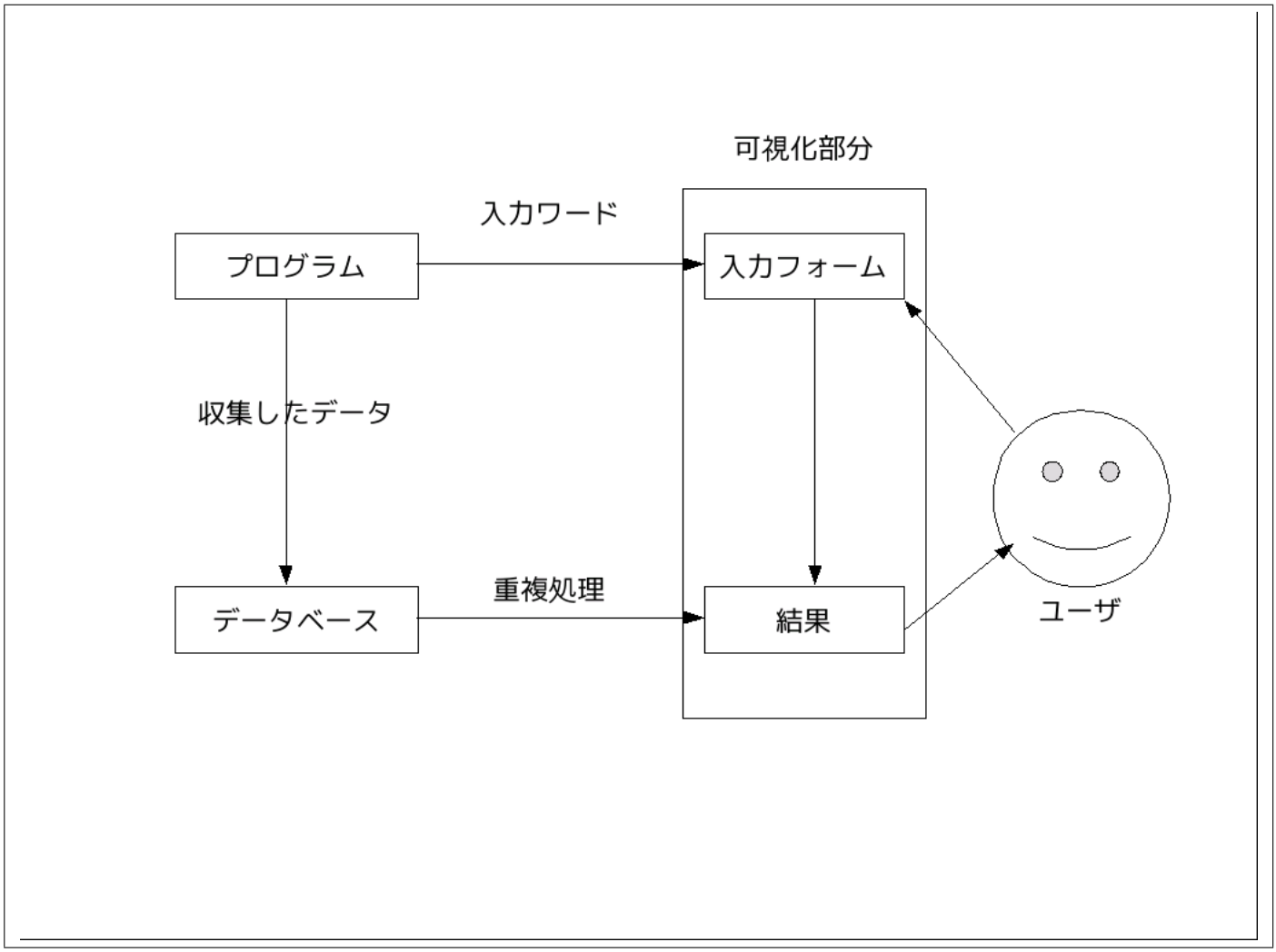


図 1　概要図

2.6 開発環境

- ハードウェア (AT 互換機)
  - CPU:Intel(R) Celeron(R) CPU 2.80GHz
  - Cache Memory:256KB
- OS：Vine Linux 4.2 [研究室内マシン]
- 開発言語：PHP 5.2.6 [研究室内マシン]
- データベース：MySQL 5.0.27 [室内マシン]

2.7 処理の流れ

ユーザーが実際に行うのは検索したいキーワードをテキストボックスに入力するだけである。その後は内部の処理によって楽曲データをまとめて HTML のテーブルを用いて表示する。入力部分は図 2 に示す。

図 2　入力部分

2.8 内部処理

2.8.1 ワードの変換

ユーザーによって入力されたキーワードで検索する際に実際のサイトの検索テキストボックスにそのワードを入れるのではなく直接 URL に渡して HTML を取得してくる。そのときサイトによって入力されたワードは URL エンコードを行う。

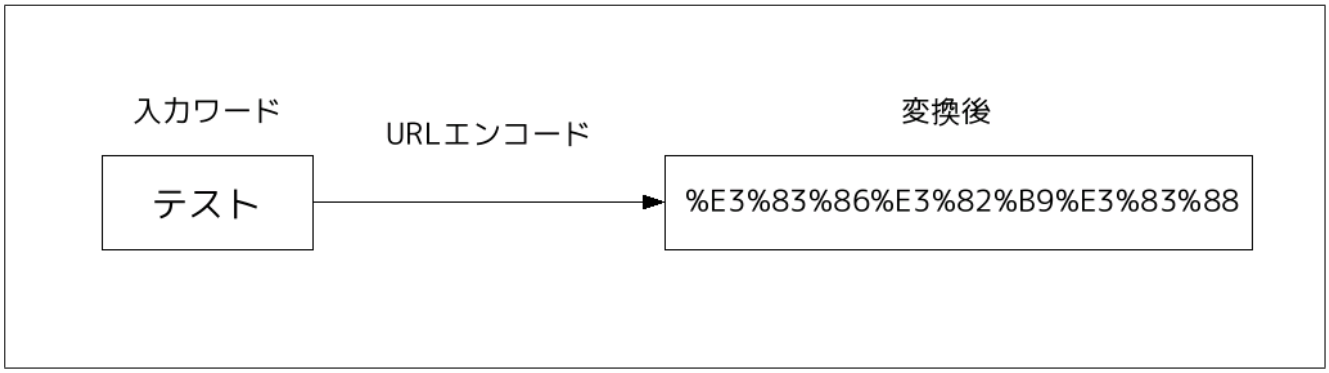


図 3　エンコード例

2.8.2 URL の抽出

URL から HTML を取得してその中から実際に楽曲データのあるページの URL を正規表現を用いて URL を判別し抽出する。このときタグの中に URL があるが、同じ様なタグのなかに関係のないものがある場合がある。それを考慮しタグの前後に必要部分にマッチする分付け足して抽出する。

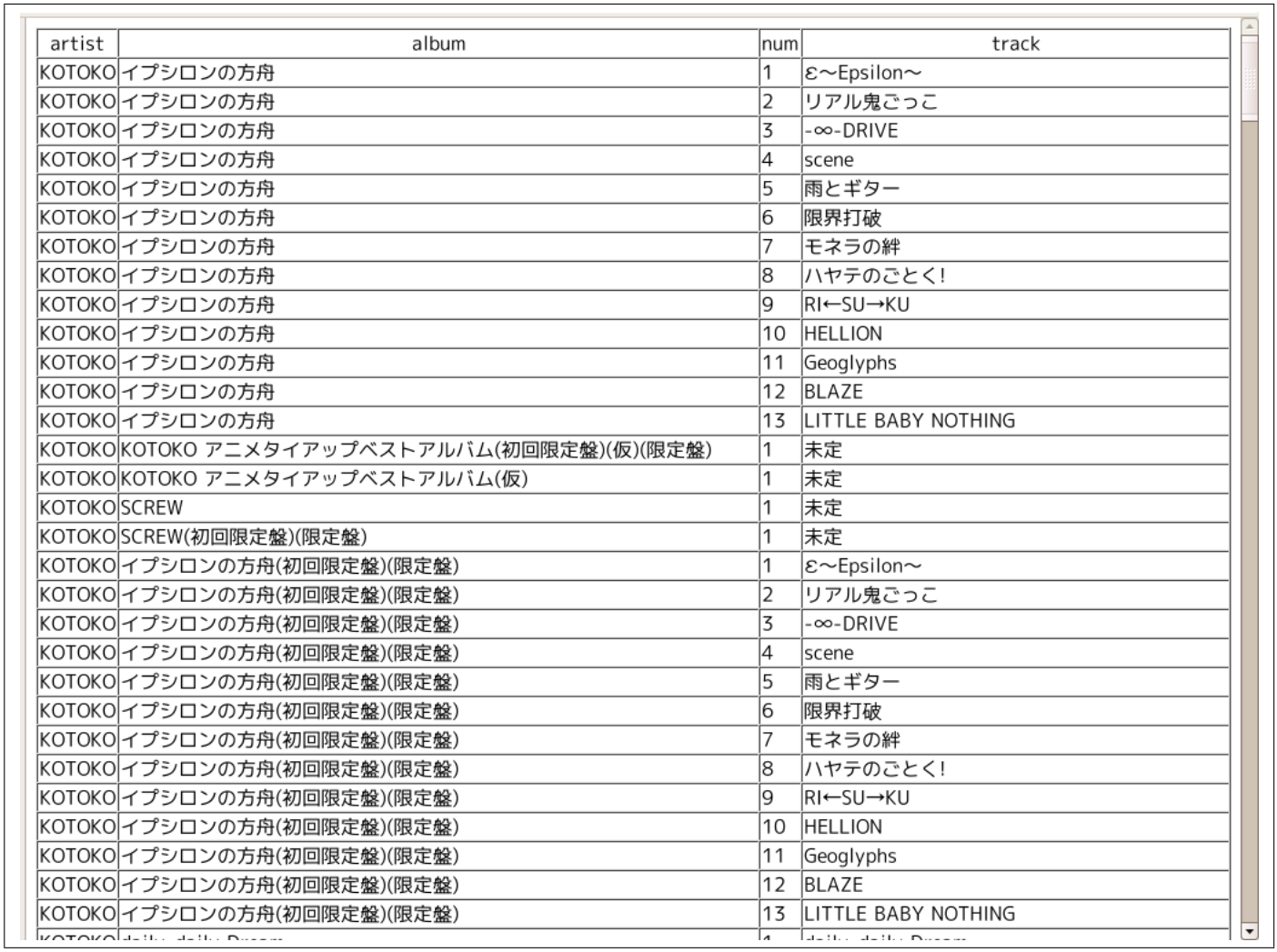
図 4　抽出部分の例

2.8.3 **データベース内の処理**

楽曲データを抽出した後データベースに抽出したデータを全て入れる。今回のシステムでは楽曲名、アーティスト名、アルバム名をデータベースに入れる。

2.8.4 **出力**

データベースのものをそのまま出力する。このときデータベース内で重複したものがある場合は出力する時に削除し、出力する。



artist	album	num	track
KOTOKO	イプシロンの方舟	1	ε～Epsilon～
KOTOKO	イプシロンの方舟	2	リアル鬼ごっこ
KOTOKO	イプシロンの方舟	3	-∞-DRIVE
KOTOKO	イプシロンの方舟	4	scene
KOTOKO	イプシロンの方舟	5	雨とギター
KOTOKO	イプシロンの方舟	6	限界打破
KOTOKO	イプシロンの方舟	7	モネラの絆
KOTOKO	イプシロンの方舟	8	ハヤテのごとく!
KOTOKO	イプシロンの方舟	9	RI←SU→KU
KOTOKO	イプシロンの方舟	10	HELLION
KOTOKO	イプシロンの方舟	11	Geoglyphs
KOTOKO	イプシロンの方舟	12	BLAZE
KOTOKO	イプシロンの方舟	13	LITTLE BABY NOTHING
KOTOKO	KOTOKO アニメタイアップベストアルバム(初回限定盤)(仮)(限定盤)	1	未定
KOTOKO	KOTOKO アニメタイアップベストアルバム(仮)	1	未定
KOTOKO	SCREW	1	未定
KOTOKO	SCREW(初回限定盤)(限定盤)	1	未定
KOTOKO	イプシロンの方舟(初回限定盤)(限定盤)	1	ε～Epsilon～
KOTOKO	イプシロンの方舟(初回限定盤)(限定盤)	2	リアル鬼ごっこ
KOTOKO	イプシロンの方舟(初回限定盤)(限定盤)	3	-∞-DRIVE
KOTOKO	イプシロンの方舟(初回限定盤)(限定盤)	4	scene
KOTOKO	イプシロンの方舟(初回限定盤)(限定盤)	5	雨とギター
KOTOKO	イプシロンの方舟(初回限定盤)(限定盤)	6	限界打破
KOTOKO	イプシロンの方舟(初回限定盤)(限定盤)	7	モネラの絆
KOTOKO	イプシロンの方舟(初回限定盤)(限定盤)	8	ハヤテのごとく!
KOTOKO	イプシロンの方舟(初回限定盤)(限定盤)	9	RI←SU→KU
KOTOKO	イプシロンの方舟(初回限定盤)(限定盤)	10	HELLION
KOTOKO	イプシロンの方舟(初回限定盤)(限定盤)	11	Geoglyphs
KOTOKO	イプシロンの方舟(初回限定盤)(限定盤)	12	BLAZE
KOTOKO	イプシロンの方舟(初回限定盤)(限定盤)	13	LITTLE BABY NOTHING

図 5　結果画面

2.9 結果と考察

今回の研究で実際にスパイダリングを用いてアーティスト名で検索をし、その結果を表示することができた。これは目的としていた検索での時間を短縮することができ、また、複数の Web サイトを用いることによって情報量を増やすことにも成功したということになる。これにより複数検索する手間なども削減できた。

図 6 は実際に Web サイトで検索した結果である。

図 6　検索サイトでの検索結果

検索結果では 44 件の検索結果がでた。しかしここからさらにリンクを辿り、楽曲データを見ないといけなくなる。

本研究での検索結果は図 5 及び図 7 の結果である。

KOTOKO	Starlight Symphony-KOTOKO LIVE 2006-IN YOKOHAMA ARENA〈通常版〉	22	Leave me hell alone
KOTOKO	Starlight Symphony-KOTOKO LIVE 2006-IN YOKOHAMA ARENA〈通常版〉	23	Princess Bride!
KOTOKO	Starlight Symphony-KOTOKO LIVE 2006-IN YOKOHAMA ARENA〈通常版〉	24	きゅるるんKissでジャンボ♪♪
KOTOKO	Starlight Symphony-KOTOKO LIVE 2006-IN YOKOHAMA ARENA〈通常版〉	25	さくらんぼキッス～爆発だも～ん～
KOTOKO	Starlight Symphony-KOTOKO LIVE 2006-IN YOKOHAMA ARENA〈通常版〉	26	Short Circuit
KOTOKO	Starlight Symphony-KOTOKO LIVE 2006-IN YOKOHAMA ARENA〈通常版〉	27	覚えてていいよ
KOTOKO	ハヤテのごとく!(限定盤)	1	ハヤテのごとく!
KOTOKO	ハヤテのごとく!(限定盤)	2	泣きたかったんだ
KOTOKO	ハヤテのごとく!(限定盤)	3	ハヤテのごとく!-instrumental-
KOTOKO	ハヤテのごとく!(限定盤)	4	泣きたかったんだ-instrumental-
KOTOKO	ハヤテのごとく!	1	ハヤテのごとく!
KOTOKO	ハヤテのごとく!	2	泣きたかったんだ
KOTOKO	ハヤテのごとく!	3	ハヤテのごとく!-instrumental-
KOTOKO	ハヤテのごとく!	4	泣きたかったんだ-instrumental-
KOTOKO	きれいな旋律(限定盤)	1	きれいな旋律
KOTOKO	きれいな旋律(限定盤)	2	rush
KOTOKO	きれいな旋律(限定盤)	3	きれいな旋律＜instrumental＞
KOTOKO	きれいな旋律(限定盤)	4	rush＜instrumental＞
KOTOKO	きれいな旋律	1	きれいな旋律
KOTOKO	きれいな旋律	2	rush
KOTOKO	きれいな旋律	3	きれいな旋律＜instrumental＞
KOTOKO	きれいな旋律	4	rush＜instrumental＞
KOTOKO	UZU-MAKI(限定盤)	1	・-Introduction-
KOTOKO	UZU-MAKI(限定盤)	2	UZU-MAKI
KOTOKO	UZU-MAKI(限定盤)	3	サイダー

図 7　検索結果 2

本研究の場合は同じアーティストの場合でも 50 件以上の検索結果が得られた。また、リンクを辿らないで楽曲データを得られる。また余計な広告なども表示されない。

現在の仕様ではアーティスト名での検索にしか対応していない。今回用いた Web サイトではアーティスト名の他にもディスク名や曲名でも検索をすることができたが、それらに対応することができなかった。また、Web サイトによって HTML の構成が違い、実際に抽出する URL やデータの部分を抜き出すための正規表現を Web サイト別に用意しなくてはならなかった。しかしこれは Web サイトを追加したい場合にその Web サイトの URL だけではなく、データを抽出する正規表現も追加で記入しなくてはならない。

3　まとめ

今回の研究でスパイダリングを用いた楽曲検索システムを開発した。これにより複数の Web サイトの楽曲データをまとめることによって、実際にページを見る手間と時間の短縮につながった。また PHP での作成なので、インターネット環境があれば使える。

今回の研究では実験的に 2 つのサイトを用いた。その結果欲しい情報を収集することができた。しかし重複処理の時に空白文字の有無で同じ内容の結果が表示されてしまうことがあった。これにより少し見づらくなってしまった。

3.1 今後の課題

- 今回はアーティスト名の検索しか対応していない。しかし検索サイトでは曲名検索やアルバムのタイトルなどでも検索できる。よって今回のシステムでもそれらの検索方法を追加する。
- 同じ結果の処理方法同じ結果のものがあった場合、曲名やアルバム名に空白文字の有無で重複処に引っかからず、そのまま表示されてしまう。よって空白文字の有無でも重複処理を行い表示されないようにする。
- 表示方法の変更今回は表示で HTML のテーブルを用いてアーティスト名、アルバム名、トラック番号、曲名を表示させた。しかしこれでは見づらいので見やすい表示に変更する。またユーザーが欲しい情報だけ表示出きるように改良を行う。
- 検索対象の増加今回は検索サイトを 2 つしか使用しなかった。しかし 2 つでは少ないので、他のサイトも検索対象に入れる。
- 正規表現の自動化プログラムに検索対象のサイトを追加した時等に正規表現を 1 つずつ追加するのは手間である。なのでサイトごとに自動で欲しいデータ部分を抽出する正規表現を、自動で作成するようにする。

謝辞

本研究を行うに当たり、ご指導及びご協力頂いた大垣斉講師、藤井信夫教授、水野貴弘先輩、小山翔平先輩、情報安全工学実験室のメンバー及び卒業生の方々、その他、見守ってくださった方々に深く感謝の意を表します。

参考文献

[1] Spider, [http://mikilab.doshisha.ac.jp/dia/research/report/2005/0813/008/report20050813008.html]
[2] Kevin Hemenway, Tara Calishain 共著 村上雅章訳「SPIDERING HACKS」オライリージャパン,2004
[3] URL エンコード,[http://ja.wikipedia.org/wiki/URL エンコード]

付録 A　ソースコード

A.1　input.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
    <meta http-equiv="content-type" content="text/html; charset=euc-jp">
    <title>検索システム入力部分</title>
    </head>
    <body>

 <h1>
            楽曲検索システム
        </h1>

        <form action="sotu.php" method="post" >

            <p>検索キーワード:</p>
            <input type="text" name="word" width="50" >


            <input type="submit" value="検索" >

        </form>

    </body>
    </html>
```

A.2 main.php

```
<?php
//文字を受け取る
$key = urlencode(mb_convert_encoding($wordg, 'UTF-8', 'auto'));
$key2 = urlencode(mb_convert_encoding($wordg, 'EUC-JP', 'auto'));
//受け取った文字列を urlencode で変える
//mysql 定義
$host = "localhost";
$user = "fuzita";
$pass = "";
$db = "fuzita";
$sql = mysql_connect($host,$user,$pass);
$sdb = mysql_select_db($db,$sql);

//正規表現
$pattern = '> <a href="(\/search\/album[^*"]+)';
$name = '<div class="track\_name">([^*<]+)';
$title = 'Album<\/strong> >([^*<]+)';
$patterna = 'Search<\/a> > ([^*-]+)';
$page = '\/search\/index[^*"]+';

//proxy の設定
$opts = array(
    "http" => array(
      'proxy' => "tcp://P.a4w:9090",
   ),
);


//ストリーム設定
$context = stream_context_create($opts);


//gracenote
//対象 URL
$gurl = 'http://www.cddb.jp/search/?query='.$key.'&search_type=artist';
//対象の URL を開く
$gfile = file_get_contents($gurl, false, $context);
//ページ数のカウント
preg_match_all( "/".$gpage."/i", $gfile, $ggets );
$pageg = count($ggets[0]);
```

```
for($c = 1; $c <= $pageg; $c++){
    //ページを開く
    $urlg ='http://www.cddb.jp/search/index.php?query='.$key.'&search_type=artist&from=0&pageID='.$c;
    $fileg = file_get_contents($urlg, false, $context);

    //マッチしたものを格納する
    preg_match_all( "/".$pattern."/i", $fileg, $match );
    $kazu = count($match[1]);
    for($i = 0; $i < $kazu; $i++){
        $buffer = "http://www.cddb.jp".$match[1][$i];
        $gurl = file_get_contents ($buffer, false, $context);

        $buf = mb_convert_encoding($gurl, "EUC-JP", "auto");
        preg_match("/".$title."/i", $buf, $disc);
        preg_match("/".$patterna."/i", $buf, $artist);
        preg_match_all("/".$name."/i" ,$buf, $gets);
        $track = count($gets[1]);
        for($j = 0; $j < $track; $j++){
        $t = ($gets[1][$j]);
        mysql_query("insert into searchs(artist, album, num, track)values('$artist[1]', '$disc[1]', $j+
        }
     }
}



//yahoomusic
//正規表現
$words = '\/">(.*)<\/a> <\/td>';
$boge = '<a href="http:\/\/music.yahoo.co.jp\/artist\/dtl\/(.*)\/">'.$wordg.'<\/a><\/td>';
$pty1 = '「<a href="(.*)">.*<\/a>」 ';
$titley = '<td class="lft"><a href=".*">(.*)<\/a> <\/td>';
$ay = '<p class="artist"><a href=".*">(.*)<\/a><\/p>';
$aly = '<h1>「(.*)」<\/h1>';

//ページ開く
$urly = 'http://search.music.yahoo.co.jp/musicsearch?cc=&cp='.$key2;
$filey = file_get_contents($urly, false, $context);

//検索アーティストの部分のみ抜き出す
preg_match_all( "/".$boge."/i", $filey, $page2);
//print_r ($page2);
```

```
$page3 =’http:\/\/music.yahoo.co.jp\/artist\/record\/’.$page2[1][0].’\/[0-9]+\/-rel\/’;
$urly2 = ’http://music.yahoo.co.jp/artist/record/’.$page2[1][0];
//そのアーティストの作品ページの URL 作成

//print $urly2."\n";
//print $page3."\n";

//作成した URL に接続
$filey2 = file_get_contents($urly2, false, $context);
//ページ数の所得
preg_match_all("/".$page3."/i", $filey2, $page4);
//print_r ($page4);
$pagec = count($page4[0]);
//print $pagec;

for($d = 0; $d < $pagec/2; $d++){
    //ページを開く
    $urly3 = ’http://music.yahoo.co.jp/artist/record/’.$page2[1][0].’/’.(1+(20*$d)).’/-rel/’;
    //print $urly3;



    $filey3 = file_get_contents($urly3, false, $context);
    //開いている状態

    //作品ごとの URL の所得
    preg_match_all( "/".$pty1."/i", $filey3, $matchy );
    //print_r ($matchy);
    $kazuy = count($matchy[1]);
    //print $kazuy."\n";
    for($i = 0; $i < $kazuy; $i++){
        $buffery = $matchy[1][$i];
        //print $buffery."\n";
        $gurly = file_get_contents($buffery, false, $context);

        preg_match("/".$aly."/i", $gurly, $alyg);
        //print_r ($alyg)."\n";
        preg_match("/".$ay."/i", $gurly, $ayg);
//print_r ($ayg)."\n";

preg_match_all("/".$titley."/i", $gurly, $titleyg);
        //print_r ($titleyg[1]);
        $tracky = count($titleyg[1]);
        for($j = 0; $j < $tracky; $j++){
```

```
        $ty = ($titleyg[1][$j])."\n";
        mysql_query("insert into searchs(artist, album, num, track) values('$ayg[1]', '$alyg[1]', $j+1,
        }
     }
}



$hyou = "select distinct * from searchs";
//$hyou = "select  * from searchs group by album";
$resul = mysql_query($hyou,$sql);



<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

echo "<table border=1 celpadding=0 cellspacing=0>";
// テーブルのフィールド名の取得
// 指定されたデータベース、テーブル内のフィールドリストの取得
$rs2 = mysql_list_fields('fuzita','searchs',$sql);

// 結果セット内のフィールド数の取得
$num_rows = mysql_num_fields($rs2);

// フィールドがある場合
if($num_rows > 0){
   // 表のヘッダー部（フィールド名）の表示開始
   print "<tr>\n";

   // 結果セット内のレコードを順次参照
   for($h = 0; $h < $num_rows; $h++){
      // フィールド名の取得
      $field_name = mysql_field_name($resul,$h);
      // フィールド名をセル内に表示
      print "<td align=center>{$field_name}</td>\n";
      }
   // 表のヘッダー部の表示終了
   print "</tr>\n";

   // 結果セットの各レコードを順次、連想記憶に格納する
   while($arr_record = mysql_fetch_assoc($resul)){
      // 行の表示の開始
```

```
      print "<tr>\n";

      // 連想配列のキー値をフィールド名に、
      // 値をフィールド値として取り出す
      foreach($arr_record as $field_name => $field_value){
      // フィールド値をセル内に表示
      print "<td>{$field_value}</td>\n" ;
      }
    // 行の表示の終わり
    print "</tr>\n";
    }
}
// 表の表示の終了
print "</table>\n";

// データベースサーバの切断
//mysql_close($db);

?>
```