WEB クローラーの基本
～見守りネットパトロールシステムとの連携～

ソリューション本部ソリューションサービス部

太田 桂吾

1. はじめに

前号（OGI Technical Reports vol.20）で紹介した「見守りネットパトロールシステム」で一部紹介したWEB(サイト)クローラーの構築、運用する際の手法、現状での問題点を報告する。

ちなみにWEBクローラーとは、人手を介することなく、プログラムにより自動的にWEBサイトから情報を取得するための仕組みの総称である。別名、bot、スパイダー、スクレイパーとも呼ばれる。近年、いわゆる“ビッグデータ“のキーワードで象徴される大容量データの解析技術が一般化されてきたこともあり、クローリングに関する様々な手法が注目されている。

2. WEB クローラーの構築

2. 1 仕組

クローラーを構築する仕組は、おおきく2種類存在する。

① 検索を行う際にブラウザの動作をシミュレートし、リクエスト情報の送信、結果の受信・解析するプログラムを作成する。

② 検索サイトから提供されているAPIを使用して、プログラムを作成する。

以降、①を「シミュレートタイプ」、②を「APIタイプ」と呼ぶ。

なお、本稿では既存の検索サイトを使用して、検索語を含むページのURLを取得することを前提としている。

ここで言う検索サイトとは、Google、Yahoo!などの広く知られたサイト、および特定サービス内の検索サイト（掲示板サイト内のプロフィール検索機能など）を指す。

2. 2 注意事項

検索サイトは広く全世界のユーザーに公開されているため、特定ユーザーによるプログラムからのリクエストによって、その機能を独占することを許可しない場合が多い。そのため、検索エンジンを使用する際には、事前にその使用条件（プログラムからのアクセスが許可されているか？など）を確認しておく必要がある。

たとえばGoogleの場合は、1秒間に数回、もしくは、ブラウザからのリクエストではなく、検索用URLを組み立てて独自に検索したような場合は、以下のエラーが表示され、同一IPからの検索は数時間できなくなる。

「Google エラー... 申し訳ありません.....ご使用のコンピュータまたはネットワークから自動リクエストが送信されている可能性があります。」

3. 検索サイト別の考察

3. 1 検索サイト① Google

(1)シミュレートタイプ

利用規約で禁止されている。また、前頁にも記載したように、たとえ検索リクエストをシミュレートしても、Google 側で様々なガードをかけており（クッキーの使用、など)、使用できない。

(2)API タイプ

Google はさまざまな API を提供しているが、検索系の API に関しては、実運用上、制限が多く、使用に耐えるものはない状況にある。

・Google Web Search API

1 ページは 8 件まで、8 ページ以上は取得できない。（この制限は公式文書には記載されていないが、実質存在する）

・Google Custom Search API

無償であれば 1 ページは 10 件まで、検索は 1 日あたり 100 回まで。以降は有償となる。

3. 2 検索サイト② Yahoo!

(1)シミュレートタイプ

可能である。現在のところ唯一使用できそうな、広く浅く検索できるタイプ。

(2)API タイプ

以前はさまざまな API を提供していたが、2013 年 8 月 14 日で、各種検索 API の提供を終了している。

3. 3 SNS サイト

(1)Twitter

以下の 2 種類の API が公開されている。

・Twitter API 1.1

任意のキーワードでツイートを検索可能。ただし、「1 回のアクセスで 100 ツイートまで」「アクセスは 15 分で 180 回まで」の制限がある。

・Twitter Streaming API

現在流れているツイートをそのまま取得する。以下の 3 種類ある。

1) FireHouse (ツイートすべて)
2) Sample(全ツイートの 1%)
3) Filter(キーワードでフィルター)

FireHouse に関しては、日本では NTT データが契約しており、NTT データから購入することになる。一般的な傾向を見るためであれば sample(全ツイートの 1%)でも十分である。なお、Filter の場合は日本語ローカライズされていないため、現在のままでは使用が難しい。

(2)Facebook

Graph API を使用することで、コメントなどのデータを取得することが可能。

3. 4 その他のサイト

(1)2ch

API の提供はないが、以下の URL へアクセスすることで「板」の一覧を取得可能。
http://menu.2ch.net/bbstable.html
また、各板の URL の最後に subback.html を付加することでスレッドの一覧を取得可能。

(2)ameba blog

ameba ブログ内で、プロフィール検索、ブログ検索をシミュレート可能。
http://search.profile.ameba.jp/general/profile/searchProfile.do

4. プログラムの実際

4. 1 ameba Blog のプロフィール検索の自動化

シミュレーションタイプで ameba Blog のプロフィール検索を行い、その結果から amebaBlog の ID を取得する例を紹介する。ID を取得することで「見守りネットパトロールシステム」などで監視するブログサイト、リアルサイトの URL リストを構築する。

(1)開発環境

・OS：

Windows7 pro

・プログラム言語：

ActivePerl v5.16.3(perl)

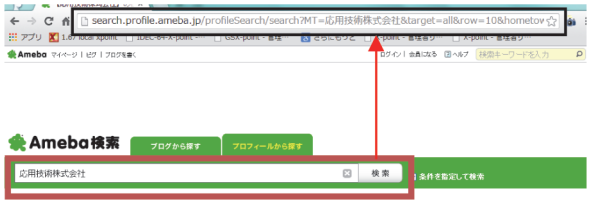
・perl モジュール：

LWP::UserAgent

HTML::TreeBuilder

(2)前準備

1) ameba Blog のプロフィール検索を実施し、その際に発行されるリクエスト文字列を確認



上記のように、実際にリクエストを発行し確認する。結果、以下の文字列が取得される。

「http://search.profile.ameba.jp/profileSearch/search?MT=応用技術株式会社&target=all&row=10&hometown=&ageFrom=&ageTo=&job=&bloodtype=&blogRow=」

上記から、検索キーワードは MT=以下に設定すればよいことがわかる。

2) プロフィール検索結果を確認

プロフィールを検索した後に表示される結果のソースを確認し、取得したい情報（この場合 ID）がどこに表示されているかを確認する。

<a class="profileImg" href="http://profile.ameba.jp/ohtakeigo" onclick="recordOutboundLinkClicked(this, 'link_profile', 'click_profile');return false;">」

上記のようなリンクが表示されているので、href 内のリンク情報から、「http://profile.ameba.jp/」を除くことで ID が取得できることが分かる。

(3)プログラム

プログラムのソースコードを図 1 に示す。

① WEB ユーザーエージェントクラスを定義
② 検索用 URL を定義する
③ URL を実行し結果を取得
④ 結果 HTML を解析する TreeBuilder を用意
⑤ TreeBuilder で解析
⑥ “a”タグの配列を取得
⑦ 条件にあう href 要素を見つける
⑧ 見つかった！

ポイントは、「WEB ユーザーエージェント」と「TreeBuilder」。特に「TreeBuilder」により、HTML 文書の解析、特定タグ、クラスのみの抽出の効果は高い。

(4)実行結果

```
C:¥>sample.pl
ohtakeigo ..... ID が取得される
C:¥>
```

ID(上記では ohtakeigi)が取得されると、そこからブログサイトの URL
http://ameblo.jp/ohtakeigo/ を組立て、「見守りネットパトロールシステム」監視を行う。

5. 今後の課題と展望

5. 1 課題

大手検索サイト提供の API は、不正 SEO の横行などから、制限がかけられていく方向であり、また仕様変更も多く、使用には注意が必要である。またシミュレーションタイプの使用も、常に仕様変更の脅威にさらされるため、これも注意が必要である。

5. 2 展望

将来的には、WEB 上の動画、音声データなどもクローリング可能なように、技術を追求していきたい。

＜参考文献＞

1) 「Spidering hacks―ウェブ情報ラクラク取得テクニック 101 選」(オライリー・ジャパン 2004/05)


```
#!/usr/bin/perl -w
use utf8;
binmode(STDOUT, ":utf8");
use LWP::UserAgent;
use Encode;
use HTML::TreeBuilder;
use URI::Escape;
my $ua = LWP::UserAgent->new;  ..... ①
$ua->timeout(10);
$ua->agent('Mozilla/5.0');
$uri ="http://search.profile.ameba.jp/profileSearch/search?M
T=応用技術株式会社&target=all&row=100&hometown=&ageFr
om=&ageTo=&job=&bloodtype=&blogRow=";  ..... ②
$html =  $ua->get($uri);  ..... ③
$result =  $html->decoded_content;
$tree = HTML::TreeBuilder->new;  ..... ④
$tree->parse($result);  ..... ⑤
$tree->eof();
@elements1 = $tree->find("a");  ..... ⑥
for ($j = 0; $j <= $#elements1; $j++) {
  if($elements1[$j]->attr('href') && $elements1[$j]->attr('href')
=~ /profile.ameba.jp/ ){  ..... ⑦
    $s=$elements1[$j]->attr('href');  ..... ⑧
    $s =~ s/http:¥/¥/profile.ameba.jp¥///;
    print $s;
  }
}
$tree->delete();
```

図 1