Extending Yioop with geographical location based local search

A Writing Report

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of the Computer Science

By

Vijaya Sinha

Spring 2011

Table of Contents

Introduction

It is often useful when doing an internet search to get results based on our current location. Commercial search engines like Yahoo, Google, Bing or Ask.com treat global searches differently from local searches. Google and other search engines try to determine whether the user is looking for a local result or a global result, in case the search is local it finds the location based on the IP address of the computer performing the search and return relevant local results. Current open source search engines like Nutch or Lucene [3] do not provide this facility so it would be useful to provide this facility in an open source search engine.

In my proposed project I will extend Yioop (an open source search engine distributed under the GPLv3 License) to return search results based on the IP address of the computer used to perform the search. I would get the location using the hostip.info data dumps, and plot the relevant local search results on a map using the Open street map data which provides complete earth downloads of street map data. This report provides the description of work done in CS297 for the implementation of final project in CS298.

1. Overview of deliverables:

1.1) Deliverable 1: Simple Web Crawl using Yioop

The goal of this deliverable was to understand the working of Yioop by studying its source code and performing sample crawls.

Discussion

To understand the inner workings of Yioop it was important to experiment with Yioop, for which I had to download the source code of Yioop from SeekQuarry.com and place it in the document root of Apache web server. To perform the sample web crawl I studied the documentation of Yioop and followed the steps specified to carry out the web crawl.

From my study, I discovered how to perform sample crawls using Yioop. I also learnt the work flow of the two main scripts queue_server.php and fetcher.php as they perform the crawl. From my understanding, I gathered that queue_server is responsible for maintaining queue of URL's to be seen, creating schedules for the fetcher and producing index archive bundles of crawled data which can be used to return search results The fetcher on the other hand is responsible for downloading batches of URL's provided by the queue_server which it can obtain using the scheduler.txt of url data produced by the queue_server. As the fetcher crawls the URL's it sends the summaries of downloaded pages back to the queue_server, it does this by making a request to the web server and posting data to it. This information is stored by the queue server in the data directory. The fetcher uses multi curl to fetch the batches of URL's provided by the queue server.

The study of the source code also helped me to understand how Yioop maintains the queue of URL's .The URLs maintained by the queue_server are hashed and stored in the priority queue

along with its weight. The page to crawl is determined by the weight of the URL stored in the priority queue.

The queue_server and fetcher scripts also helped me understand how Yioop follows the online page importance algorithm to decide which URL's are going to be crawled first. The Scheduler which is a part of the Queue_server gives some money (weight) to each of the URL's in the queue. When the fetcher downloads these URL it distributes the money among all the links extracted from that URL and returns the summary with the money count to the queue_server .If the page returned by the fetcher is already present in the queue of the queue_server it adds the money count returned by fetcher to its initial assigned money. In this way the URL with the most money gets the priority to be crawled first.

This deliverable helped me gain a thorough understanding of the Yioop search engine which was important to make headway into the project further.

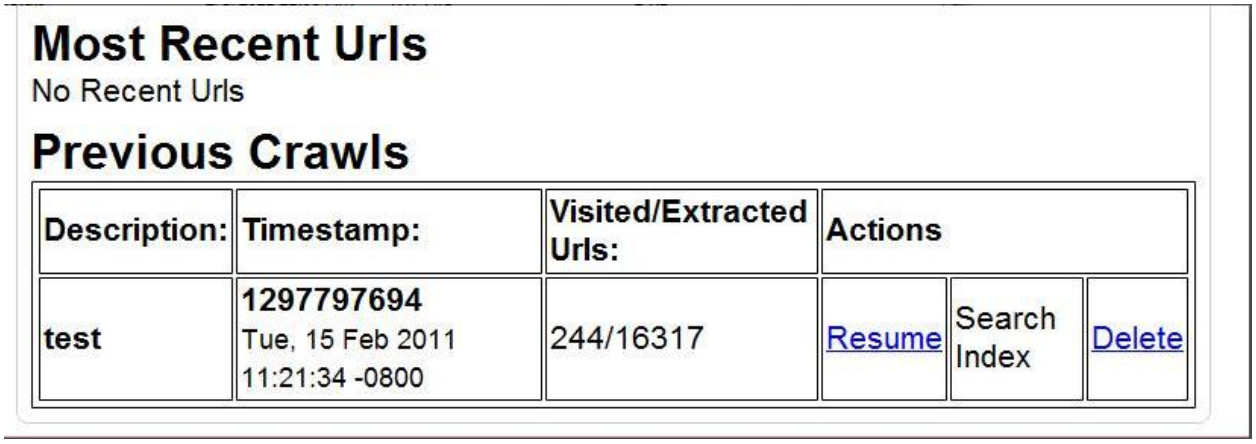


Figure 1:Sample Crawl using Yioop

1.2) Deliverable 2: An application to return geo location from Hostip database

The goal of this application was to understand the structure of hostip database, the various dependencies between the tables in the hostip database to return geo-locations based on the ipaddress provided.

Discussion:

The Hostip.info is a community-based project that provides complete raw data dumps to geolocate ip addresses. As part of this deliverable a sql dump was downloaded and imported into a mysql database. The hostip database consists of following tables:

Countries table which consists of the countries name and country code identified by primay key id.

Citybycountry table which consists of the cities, states, longitude and latitude identified by primary key city. The citybycountry table references the countries table by id.

The **ip4** tables (255) containing the second and third quad of the ip4_address and the corresponding ip4 table number denotes the first quad. The ip4 table references the citybycountry table by city and countries table by id.

The application consists of a simple form where in the user can enter the ip address and the application will query the hostip database to retrieve the corresponding geo location in terms of Country, City, State, Country_code ,Latitude and Longitude. Below is a snapshot of the application.

Figure 2:App to return Geo-Location

1.3) Deliverable 3: An application to extract open street map data

Description:

Open street map provides free map data of whole earth such as street maps and is distributed under the open source license. Planet.osm provides a snapshot of the complete Open Street map database. The file is a xml formatted osm file. The Open Street map data can be downloaded from any of the mirrors from http://wiki.openstreetmap.org/wiki/Planet.osm. Regions can be extracted from the osm file using a bounding box or region to define a region.

In order to extract the region specific to our needs, we need to understand the structure of xml formatted planet file.

Shortened version of complete Osm planet Xml file:

<**osm** version="0.6" generator="OpenStreetMap planet.c" timestamp="2011-02-16T01:11:04Z">

<**bound** box="-90,-180,90,180" origin="http://www.openstreetmap.org/api/0.6" />

<**changeset** id="878132" " max_lon="141.4390275" max_lat="43.0435509" user="masaminh" uid="93901">

</**changeset**>

<**node** id="270387" version="1" changeset="99256" user="nickw" uid="94">

```
 <tag k="created_by" v="osmeditor2" />
  </node>
<way id="99947113" version="1" changeset="7299165" user="isnogoud" uid="78608">
   <tag k="building" v="yes" />
   <tag k="source" v="cadastre-dgi-fr source : Direction Générale des Impôts - Cadastre. Mise à jour : 2011" />
   <tag k="wall" v="no" />
</way>
  <relation id="1430044">
   <tag k="name" v="Rue Geoffroy-Drouet" />
   <tag k="type" v="associatedStreet" />
 </relation>
</osm>
```

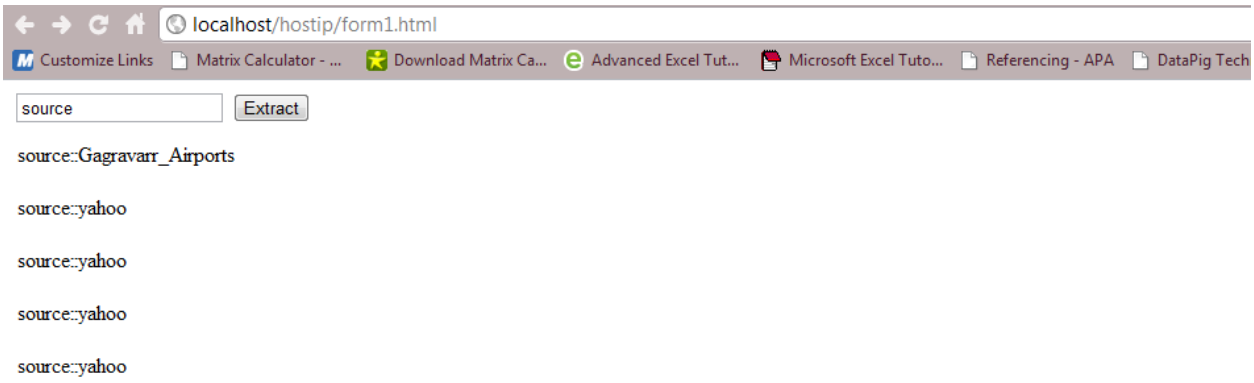
Discussion:

The goal of this deliverable was to extract some data from the planet.osm xml formatted file. DOM parser was used in order to parse and extract relevant data from the planet.osm file. As part of this deliverable I extracted the tag elements by scanning through the whole file and finding out tags denoted by the attribute k.

Code snippet for the extraction:

```
   for($j=0;$xml->node[$j]!=null && $y<5;$j++) {
     for($p=0;($xml->node[$j]->tag[$p]!=null );$p++){
     $Response['k']=(string)$xml->node[$j]->tag[$p]->attributes()->k;
     if($Response['k']==$get){
     $Response['v']=(string)$xml->node[$j]->tag[$p]->attributes()->v;
     echo $Response['k']."::".$Response['v']."<br><br>";
     $y++;
```

}



2) Conclusion

As part of CS297 I have developed components which would be helpful in the final implementation of the project in CS298. The applications that I have developed as part of CS297 have helped me understand the structure of open source data hostip.info and also helped me to understand the open street map which will be helpful in the final implementation. The implementation of the project is majorly dependent on both these datasets so understanding these datasets was essential for the final implementation. My work in CS297 has helped me gain a complete insight into it.

In CS298 I will be rendering the planet.osm data on tiles and use the hostip.info data to get the geo-locations. Based on the geo-locations, I will be able to extract relevant data from the planet.osm to make a visual map of it. I will be using Deliverable 2 and Deliverable 3 to accomplish this part of the implementation.

3) References

[1] [Hostipdatabase] Retrieved from http://www.hostip.info/

[2] [Open Street map] Retrieved from http://wiki.openstreetmap.org/wiki/Main_Page

[3] [Open Source Search engines]http://www.searchtools.com/tools/tools-opensource.html

[4] S. Buttcher, C. L. A. Clarke, and G. V. Cormack. Information Retrieval: Implementing and Evaluating Search Engines, MIT Press. 2010.