Smart Search: A Firefox Add-On to Compute a Web Traffic Ranking

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Computer Science

by

Vijaya Pamidi

December 07, 2010.

Table of Contents

1 Introduction

Search engines results are given to the user, based on the ranking and indexing strategies followed by the search algorithm of a search engine. Currently there are some tools available like www.alexa.com, www.ranking.com, www.compete.com which gives analytic data for ranking the web sites based on web traffic and the number of users who visit a web site. Alexa provides the traffic rank for a website based on two factors: The number of users that view a website and the number of pages viewed.

The main goal of my project is to create a Smart Search Firefox add-on for the Yioop search engine, an open source search engine developed by my project advisor, Dr. Chris Pollett. This Add-on will provide similar analytic data to the Yioop search engine. With the results received from the Smart Search tool, the Yioop search engine refines the search results. Eventually, users would benefit from these better search results.

The CS297 part of the master's project was done in the form of four deliverables. These will be used to develop the final project.

The CS297 report includes 7 sections: Introduction, Deliverable1 to 4, Conclusion and References. The introduction gives the purpose of the project and includes the project report. It also describes the reason and motivations of the project. Sections Deliverable 1 through Deliverable 4 explains the objectives of each deliverable and it was developed. The second section tells about Deliverable1. This includes a JavaScript program which models the Page rank algorithm by taking 10X10 matrix input. The rows represent the web pages and column values shows the outgoing links from these web pages. The result of the program is checked as the convergence of the Google's matrix on the given input matrix and the resulted page ranks given after convergence. The third section deliverable 2 includes building a simple fire fox extension with XUL. The fourth section deliverable 3 includes making toolbar fire fox extension by adding toolbar button. Functionality of the tool bar button includes alerting the user clicked links and the target link. The fifth section deliverable 4 includes make the toolbar button communicate with Yioop and the section conclusion gives a flavor of how these deliverables can be used in CS298. Last but not the least reference section gives the reference lists which were used to make the contents of the report.

2. Deliverable 1: Model Google's Page rank algorithm for 10X10 Matrix.

The Objective of this deliverable was to understand and implement Google's Page-rank algorithm for a 10X10 matrix. The algorithm needs to be developed in JavaScript. This 10X10 matrix is linked in a way to model a group of web pages connected with outgoing and incoming links. The program is checked in such a way that Google matrix should converge on a given matrix. Also the program outputs the Page-rank values for the 10 web pages given.

A brief description for Page-ranking

Internet is a source of billion of web pages which are accessible to the user. Search engine needs to find a way to give the most relevant web pages to the user by comparing the relevance and importance of all these web pages. A challenging task is to rank these web pages with an algorithm which can incorporate common usage patterns i.e. an user who visits a web page A is more likely to click the link to page B than the link to page C.

Algorithm Outline

The Power Method x ← Px will converge if P is:

Stochastic: each row contains strictly nonnegative values and the sum of the values in a row equals on

Irreducible: in the connectivity matrix it must be possible to get from any web page to any other web page

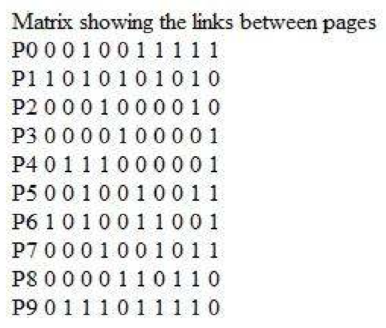
Aperiodic: every web page has a link to itself (or the user can simply refresh the page)

A detailed explanation with example can be found at-

http://en.wikiversity.org/wiki/Google_Matrix

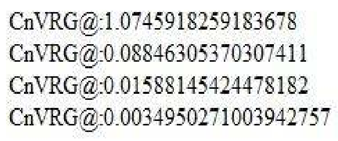
The Page-rank algorithm snippet is given below.

The inputs of our Page-rank algorithm are given below



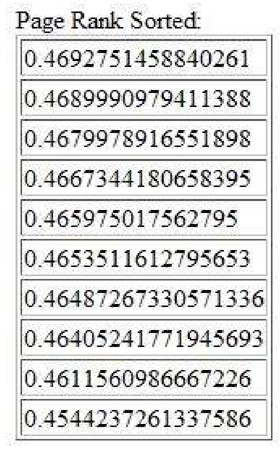
```
Matrix showing the links between pages
P0 0 0 1 0 0 1 1 1 1 1
P1 1 0 1 0 1 0 1 0 1 0
P2 0 0 0 1 0 0 0 0 1 0
P3 0 0 0 0 1 0 0 0 0 1
P4 0 1 1 1 0 0 0 0 0 1
P5 0 0 1 0 0 1 0 0 1 1
P6 1 0 1 0 0 1 1 0 0 1
P7 0 0 0 1 0 0 1 0 1 1
P8 0 0 0 0 1 1 0 1 1 0
P9 0 1 1 1 0 1 1 1 1 0
```

The algorithm convergence information looks like:



```
CnVRG@:1.0745918259183678
CnVRG@:0.08846305370307411
CnVRG@:0.01588145424478182
CnVRG@:0.0034950271003942757
```

Next we show the page rank for all the web pages from P0 to P9.



Page Rank Sorted:

```
0.4692751458840261
0.4689990979411388
0.4679978916551898
0.4667344180658395
0.465975017562795
0.4653511612795653
0.46487267330571336
0.46405241771945693
0.4611560986667226
0.4544237261337586
```

3. Deliverable 2:

The Objective of this deliverable was to make a simple Fire fox extension with a Status bar displaying "Hello, World". To learn more about Fire fox extension features like "Hello-World" menu pop up with menu items labeled was added. Each menu item alerts a message when clicked. A “New-Tool” was added to the tool menu pop up to learn about the” insert after” feature.

XUL (pronounced as “zool”) is one of the many technologies used for creating Mozilla based extensions and products. To program any Fire fox extension one should have the knowledge of XUL and the XUL documents needed to build extension.

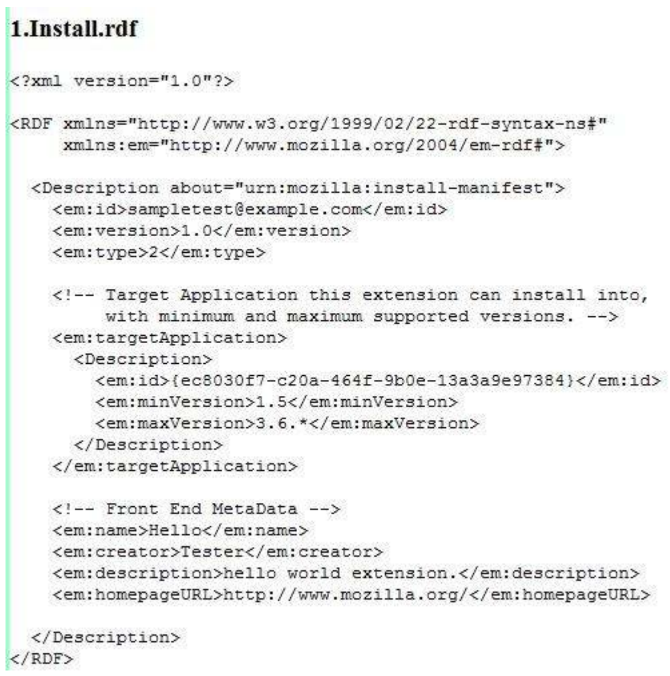
The XUL documents needed for the programmers to define an XUL user interface:

Content: the XUL document(s), whose elements define the layout of the user interface.

Skin: the CSS and image files, which define the appearance of an application

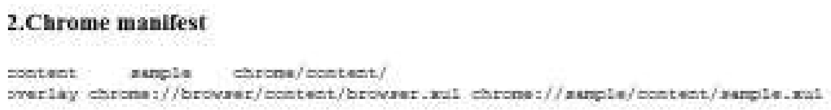
Locale: the files containing user-visible strings for easy software localization.

The code block used to develop Deliverable2 is given below:



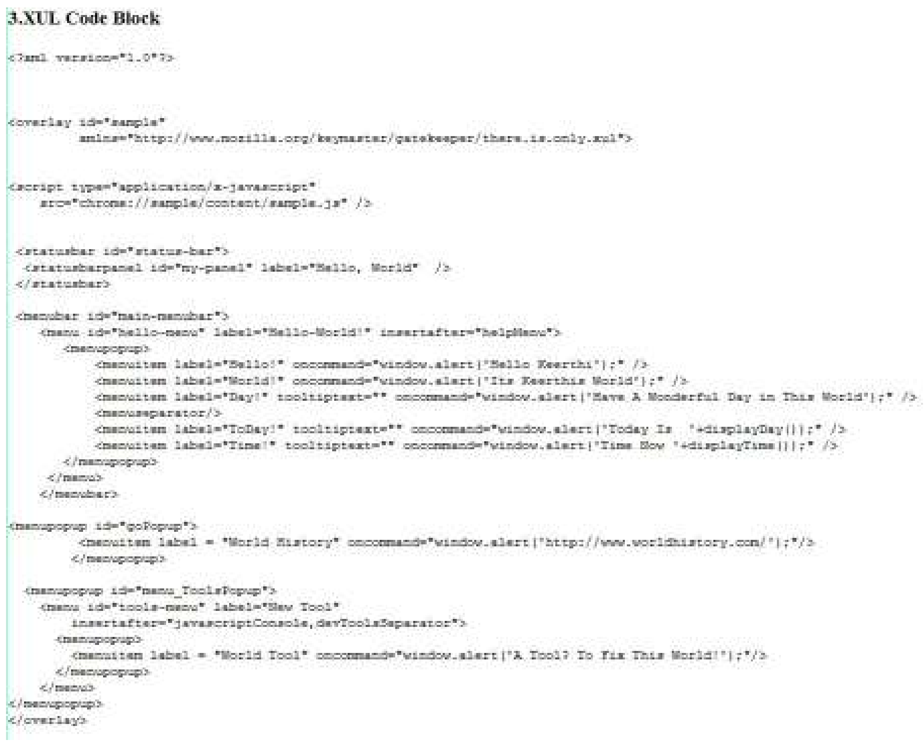
1.Install.rdf

```
<?xml version="1.0"?>

<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:em="http://www.mozilla.org/2004/em-rdf#">

  <Description about="urn:mozilla:install-manifest">
    <em:id>sampletest@example.com</em:id>
    <em:version>1.0</em:version>
    <em:type>2</em:type>

    <!-- Target Application this extension can install into,
         with minimum and maximum supported versions. -->
    <em:targetApplication>
      <Description>
        <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
        <em:minVersion>1.5</em:minVersion>
        <em:maxVersion>3.6.*</em:maxVersion>
      </Description>
    </em:targetApplication>

    <!-- Front End MetaData -->
    <em:name>Hello</em:name>
    <em:creator>Tester</em:creator>
    <em:description>hello world extension.</em:description>
    <em:homepageURL>http://www.mozilla.org/</em:homepageURL>

  </Description>
</RDF>
```



2.Chrome manifest

```
content     sample    chrome/content/
overlay chrome://browser/content/browser.xul chrome://sample/content/sample.xul
```



3.XUL Code Block

```
<?xml version="1.0"?>


<overlay id="sample"
         xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">


<script type="application/x-javascript"
    src="chrome://sample/content/sample.js" />


 <statusbar id="status-bar">
  <statusbarpanel id="my-panel" label="Hello, World"  />
 </statusbar>

 <menubar id="main-menubar">
    <menu id="hello-menu" label="Hello-World!" insertafter="helpMenu">
       <menupopup>
           <menuitem label="Hello!" oncommand="window.alert('Hello Keerthi');" />
           <menuitem label="World!" oncommand="window.alert('Its Keerthis World');" />
           <menuitem label="Day!" tooltiptext="" oncommand="window.alert('Have A Wonderful Day in This World');" />
           <menuseparator/>
           <menuitem label="ToDay!" tooltiptext="" oncommand="window.alert('Today Is  '+displayDay());" />
           <menuitem label="Time!" tooltiptext="" oncommand="window.alert('Time Now '+displayTime());" />
       </menupopup>
     </menu>
    </menubar>

<menupopup id="goPopup">
         <menuitem label = "World History" oncommand="window.alert('http://www.worldhistory.com/');"/>
        </menupopup>

  <menupopup id="menu_ToolsPopup">
    <menu id="tools-menu" label="New Tool"
        insertafter="javascriptConsole,devToolsSeparator">
      <menupopup>
        <menuitem label = "World Tool" oncommand="window.alert('A Tool? To Fix This World!');"/>
      </menupopup>
    </menu>
</menupopup>
</overlay>
```

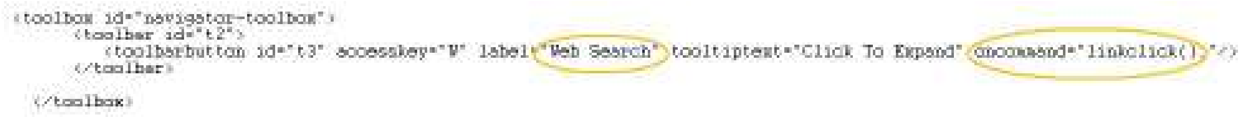
4. Deliverable 3:

The Objective of this deliverable was to make a tool-bar Fire fox extension by adding a tool bar button. The function of the tool bar button is to capture the user clicked link and target link. At this stage the captured links were checked in the alert form.

The functionality of deliverable 3 was achieved by writing a JavaScript function and calling that function when the user clicks on a link .This function captures the URL or link on which user clicked. It also captures the target link on which user clicked on the current page. When user clicks on a link an alert pops up with the link. Once clicked ok in the alert box the next alert appears with the target link. When user clicks ok on the second alert the target page will be loaded.

The code snippets for both XUL and JavaScript function is given here:

XUL code shows the tool bar button “Web Search”. The JavaScript function linkclick () is called when the “oncommand” event is triggered.

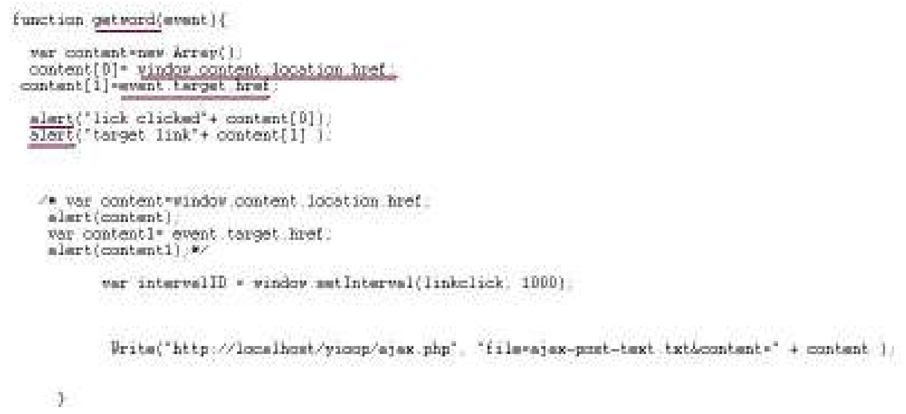


```
<toolbox id="navigator-toolbox">
      <toolbar id="t2">
         <toolbarbutton id="t3" accesskey="W" label="Web Search" tooltiptext="Click To Expand" oncommand="linkclick()"/>
      </toolbar>

  </toolbox>
```

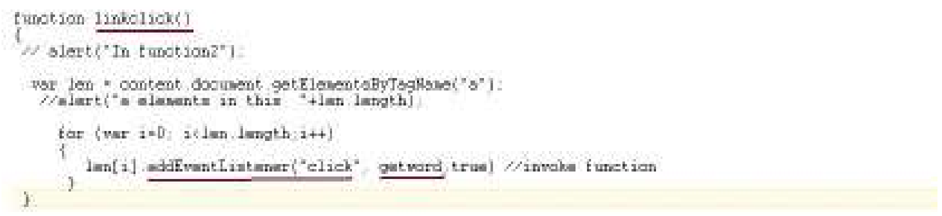
The JavaScript code below shows the function linkclick ().It gets the links with the tag name “a” in the web page. To capture the links, the command- “content.document.getElementsByTagName("a")”- was used.

In function linkclick() for loop is to traverse through each link and calls addEventListner function which has three parameters (“event”, function, true) that could be passed. In this case the three parameters are (“click”, getword, true). The “getword” is another function where the user clicks an URL or a link and target links are captured. The command window.content.location.href was used to capture user clicked link or URL and event.target.href is used to capture target links. The results are stored in an array for future use. The JavaScript code snippet is given here.

The code snippet:



```
function getword(event){

  var content=new Array();
  content[0]= window.content.location.href;
 content[1]=event.target.href;

  alert("lick clicked"+ content[0]);
  alert("target link"+ content[1] );


   /* var content=window.content.location.href;
    alert(content);
    var content1= event.target.href;
    alert(content1);*/

          var intervalID = window.setInterval(linkclick, 1000);


           Write("http://localhost/yioop/ajax.php", "file=ajax-post-text.txt&content=" + content );


    }
```



```
function linkclick()
{
 // alert("In function2");

  var len = content.document.getElementsByTagName("a");
   //alert("a elements in this  "+len.length);

     for (var i=0; i<len.length;i++)
     {
        len[i].addEventListener("click", getword,true) //invoke function
     }
 };
```

5. Deliverable 4:

The Objective of this deliverable was to make the XUL tool bar button communicate with Yioop. Functionality of this deliverable was to make the tool-bar button send the captured links in Deliverable 3 to Yioop. As part of making communication with Yioop, a POST request is made to Yioop from toolbar button. All the links sent to Yioop are saved in a text file and a link is added to access this text file.

To achieve this deliverable, JavaScript was used at user end and PHP was used for POST at the server end which in this case was Yioop. As this was the development phase of the project, the Yioop code was located at the root folder at local host and the server side programming was done in the local host for deployment. The JavaScript function “Write” was coded to achieve this task. The function has got two arguments passed to it. One is URL and the other is content. This function was invoked at the end of getword function.

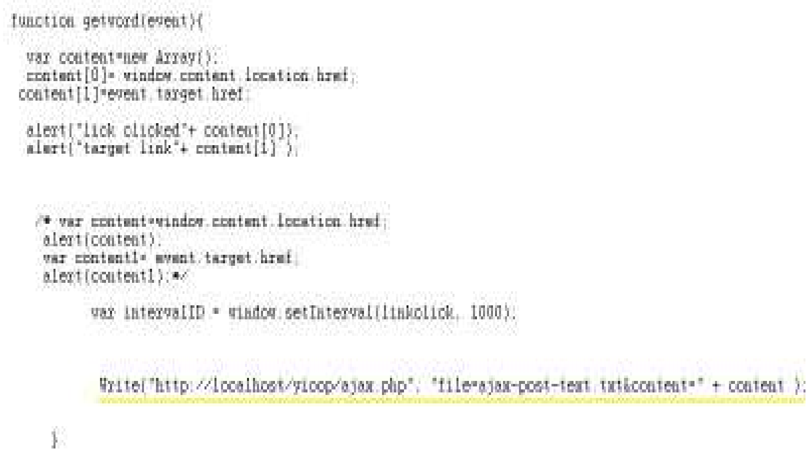
The function “Write” has “createXHR” function called in it. The “createXHR” was to establish an http request, if the “readyState ==4” is yes then a connection will be opened to the given URL and makes a POST request. Then it will POST the content to the URL, which incase here it is “http://localhost/yioop/ajax.php “.

At server side the ajax.php is responsible for the POST request and to capture the content sent from the XUL tool bar button. Once the content is received, the PHP program creates a text file with the given name in the program and writes the content into the file. once writing is done it closes the text file. Writing to the text was done in a+ mode. It is an append mode used for both writing and reading the content to the file. The a+ mode was used keeping the future project development in view.

Accessing this file is done from Yioop index page instead of accessing it with typing the location of the file in the browser. A link is created with name “Activity” in the index page of the Yioop code. When clicked on the link it navigates to the page where links were saved.

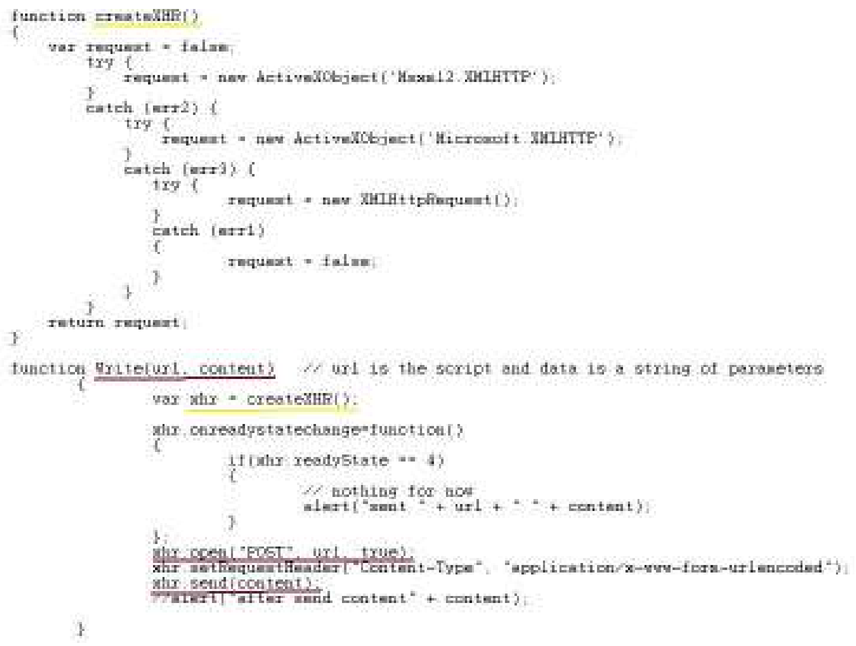
The code snippet:

JavaScript code where “Write” function is invoked in “getword” function



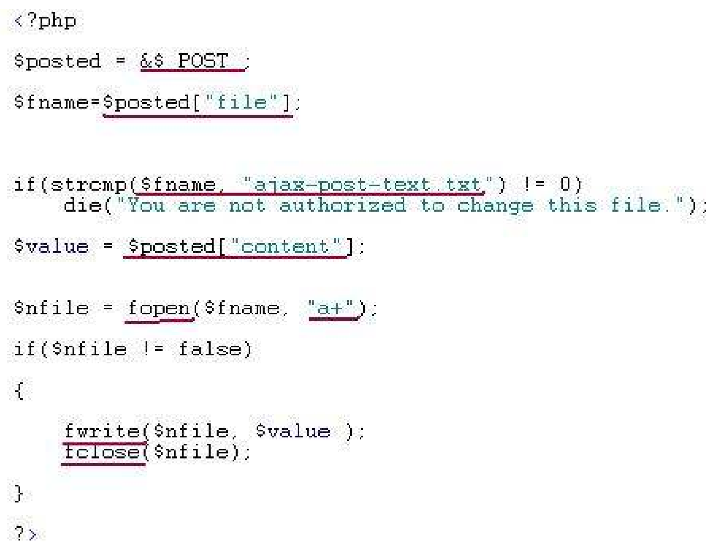
```
function getword(event){

  var content=new Array();
  content[0]= window.content.location.href;
 content[1]=event.target.href;

  alert("lick clicked"+ content[0]);
  alert("target link"+ content[1] );


   /* var content=window.content.location.href;
    alert(content);
    var content1= event.target.href;
    alert(content1);*/

          var intervalID = window.setInterval(linkclick, 1000);


           Write("http://localhost/yioop/ajax.php", "file=ajax-post-text.txt&content=" + content );

    }
```

JavaScript code that shows how the "createXHR ()" and "Write ()" functions were implemented.



```
function createXHR()
{
    var request = false;
        try {
            request = new ActiveXObject('Msxml2.XMLHTTP');
        }
        catch (err2) {
            try {
                request = new ActiveXObject('Microsoft.XMLHTTP');
            }
            catch (err3) {
                try {
                        request = new XMLHttpRequest();
                }
                catch (err1)
                {
                        request = false;
                }
            }
        }
    return request;
}

function Write(url, content)   // url is the script and data is a string of parameters
        {
                var xhr = createXHR();

                xhr.onreadystatechange=function()
                {
                        if(xhr.readyState == 4)
                        {
                                // nothing for now
                                alert("sent " + url + " " + content);
                        }
                };
                xhr.open("POST", url, true);
                xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
                xhr.send(content);
                //alert("after send content" + content);

        }
```

Ajax.php code that shows how the POST was handled and how the content was written into the text file.

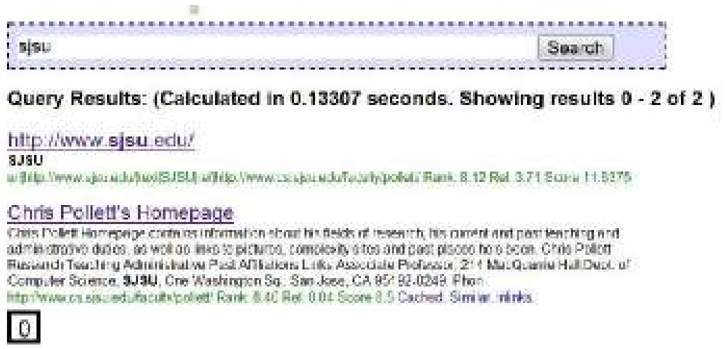


```
<?php

$posted = &$_POST ;

$fname=$posted["file"];


if(strcmp($fname, "ajax-post-text.txt") != 0)
    die("You are not authorized to change this file.");

$value = $posted["content"];


$nfile = fopen($fname, "a+");

if($nfile != false)

{

    fwrite($nfile, $value );
    fclose($nfile);

}

?>
```

6. Yioop Code Study:

The Smart search Add -on was built to work with Yioop search engine. The code study was to know the flow of function control from the point where user enters the search query in the text field at index page to the point where search engine gives the results to the user. To understand the Yioop code in a better way, a small task like changing the color of a part of the search results was done. The matching word in the results for the search query was changed to color green. For example if user entered the word "sjsu" in search text field, each "sjsu" word in the results was changed to color green.

The Yioop results before change.



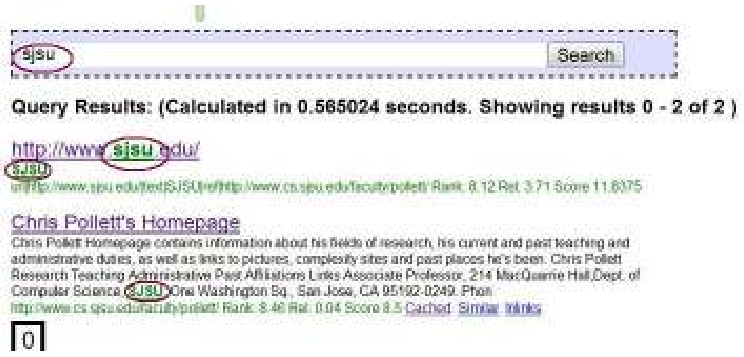
Query Results: (Calculated in 0.13307 seconds. Showing results 0 - 2 of 2)

http://www.sjsu.edu/
SJSU
u:http://www.sjsu.edu/text|SJSU|u:http://www.cs.sjsu.edu/faculty/pollett/ Rank: 8.12 Rel: 3.71 Score 11.8375

Chris Pollett's Homepage
Chris Pollett Homepage contains information about his fields of research, his current and past teaching and administrative duties, as well as links to pictures, complexity sites and past places he's been. Chris Pollett Research Teaching Administrative Past Affiliations Links Associate Professor, 214 MacQuarrie Hall,Dept. of Computer Science, **SJSU**, One Washington Sq., San Jose, CA 95192-0249. Phon
http://www.cs.sjsu.edu/faculty/pollett/ Rank: 8.40 Rel: 0.04 Score 8.5 Cached. Similar. Inlinks

0

The Yioop results after the color change.



Query Results: (Calculated in 0.565024 seconds. Showing results 0 - 2 of 2)

http://www.sjsu.edu/
SJSU
u:http://www.sjsu.edu/text|SJSU|u:http://www.cs.sjsu.edu/faculty/pollett/ Rank: 8.12 Rel: 3.71 Score 11.8375

Chris Pollett's Homepage
Chris Pollett Homepage contains information about his fields of research, his current and past teaching and administrative duties, as well as links to pictures, complexity sites and past places he's been. Chris Pollett Research Teaching Administrative Past Affiliations Links Associate Professor, 214 MacQuarrie Hall,Dept. of Computer Science, SJSU, One Washington Sq., San Jose, CA 95192-0249. Phon
http://www.cs.sjsu.edu/faculty/pollett/ Rank: 8.46 Rel: 0.04 Score 8.5 Cached Similar Inlinks

0

7. Conclusion:

The conclusion part of the report gives a brief description of how each deliverable developed in CS297 is going to be used to complete the requirement for the CS298.

Deliverable 1: Model Google's Page rank algorithm for 10X10 Matrix.

The outcome of this deliverable is a clear understanding of Page-rank algorithm. This outcome will be definitely helpful to understand the Yioop's ranking algorithm and how to recalculate the Page-rank based on the data statistics received by Yioop. This will be done in the second phase of the project CS298.

Deliverable 2: Making a simple fire fox extension.

The outcome in this deliverable is a basic learning of how to program a Fire fox extension and the document contents that are to be included. This is used to develop the user inter face part of the project. With JavaScript code the XUL content can have added functionality. As most part of the user interface section of the development was done in CS297. Few changes like color, image and CSS properties may need to be extended based on the final requirement. If the user interface requires a basic or simple feature, then the tasks finished in deliverable 2 and 3 could be sufficient. This will be directly included in CS298.

Deliverable 3: Making a tool-bar fire fox extension by adding a tool bar button.

The task of this deliverable was to add a toolbar button to the toolbar in the browser window. The functionality of this tool bar button is to alert the user clicked links and the target links. The captured links will be used to process the ranking criteria. This extra step gives refined user centric results for the search queries searched with Yioop search engine. Deliverable 3 is definitely useful as it is one of the main features. This will be used in CS298 as an extension made to it in the form of deliverable 4.

Deliverable 4: Communicate to Yioop with deliverable3.

The task of this deliverable is to send the captured user clicked links and the target links to Yioop and make Yioop save this data store in a text file at the server side. This is basically achieved with POST. There is hyperlink "Activity" provided at Yioop's index page. When clicked on the link it navigates to the page where links were saved.

This completed task of this deliverable will be used in CS298 to complete the entire requirement of refining the query results. This step of the development is planned in such a way to re- rank the web pages on top of the Yioop's ranking by taking the analytic data given by the Smart search add on which resides at user end.

In Spring 2011, as part of CS298 I will extend the CS297 project to make the Smart search Add-on send the user end information to Yioop. For this I will need to do two more features: (1) number of users that view a web site (2) Probability that user leaves some other website and enters the current one and the probability that user clicks on the link provided on the current page. I will work on making Smart search send this analytic data periodically to Yioop. . I will also take the content from the text file at Yioop and store it in a database. To achieve this I will work on the server side programming to make Yioop receive data from Add-on and refine the results. Finally I will also need to test on the Smart search results if they are better than the regular.

8. References:

[1] **http://en.wikipedia.org/wiki/XUL**

[2] http://en.wikiversity.org/wiki/Google_Matrix

[3] Google's Page Rank and Beyond: The Science of Search Engine Rankings by Amy N. Langville and Carl D. Meyer- 2006.

[4] http://developer.mozilla.org/en/docs/Building_an_Extension":Official page of Mozilla.

Article References:

[1] Konstantin Avrachenkov and Nelly Litvak. The effect of new links on Google Page Rank. Technical report, INRAIA, July 2004

[2] Matthew Richardson and Pedro Domingos.The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank. Advances in Neural Inforamtion Processing Systems, 14:1441-8, 2002.

[3] Taher H. Haveliwala (1999). Efficient computation of PageRank. Technical report, Stanford University, Stanford, CA.