Improving Yioop! User Search Data Usage

A Writing Report

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 Science

by

Tarun Pepira Ramaswamy

December 2011

Table of Contents

1. Introduction

One of Dr.Pollett’s previous student, Vijaya Pamidi wrote a Firefox extension which captures the URL‘s visited in Firefox and sends these to the Yioop Search Engine. One of the disadvantages of this extension is the lack of user benefit for sending the data to Yioop. This led to non usage of the extension. Also, the relevance of the URL data sent by the extension is computed in an ad hoc way.

The main goal of my project is to add a visualization component to this Firefox extension encouraging the users to use this extension. The visualization will show the user navigation across the visited URL’s providing user benefit. The extension will also have a preference pane to accept user configurable data like username and password. Currently, the user clicks on the Yioop search result page are not captured. This extension should capture the user clicks on Yioop search result page and send it to the Yioop Search Engine. My project will also implement a theoretically sound way of computing the relevance of these data, based on modern web reputation systems.

2. Deliverable 1: Firefox extension with a preference pane

2.1 Goal

The objective of the deliverable is to create a simple Firefox extension with a preference pane.

2.2 Description

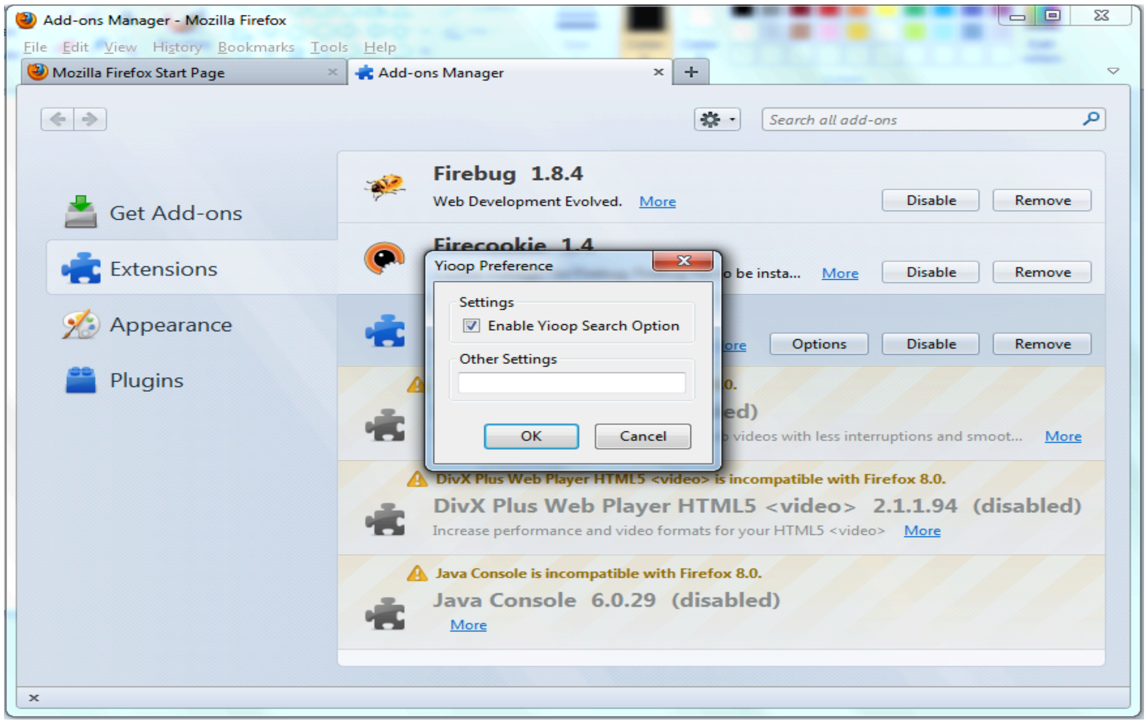
Extensions allow developers to add functionality to the browser and enhance the user interface in a way that is not directly related to the viewable content of Web pages. In Firefox, the extensions are packaged and distributed in ZIP files or Bundles, with the XPI (pronounced “zippy”) file extension. The extensions user interface is written using XUL (pronounced “zool”), CSS and JavaScript.

The basic components of the extensions are install.rdf, chrome.manifest and the main.xul file. Install.rdf contains the id and version of the extension. It also specifies required id, min and max version of the target application. The chrome.manifest contains the folder hierarchy, skin details and the main XUL file which needs to be overlayed.

An example of the content within a typical XPI file

```
/install.rdf                      //General information about your extension
/chrome.manifest                  //Registers you content with the Chrome engine
/chrome/
/chrome/content/                  //Contents of your extension like XUL and js file
/chrome/icons/default/*           //Default Icons of the extension
/chrome/locale/*                  //Building an Extension# Localization
/defaults/preferences/*.js        //Building an Extension# Defaults Files
/plugins/*
/components/*
/components/cmdline.js
```

In this deliverable, I created a Firefox extension that has a simple preference pane. It also shows a pop up message when a user clicks any link in the webpage. The snapshots below show these features.


Add-ons Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Mozilla Firefox Start Page
Add-ons Manager
Search all add-ons
Get Add-ons
Extensions
Appearance
Plugins
Options
Disable
Remove
More

Firebug 1.8.4
Web Development Evolved.

Firecookie 1.4

Yioop Preference

Settings

[✓] Enable Yioop Search Option

Other Settings

OK Cancel

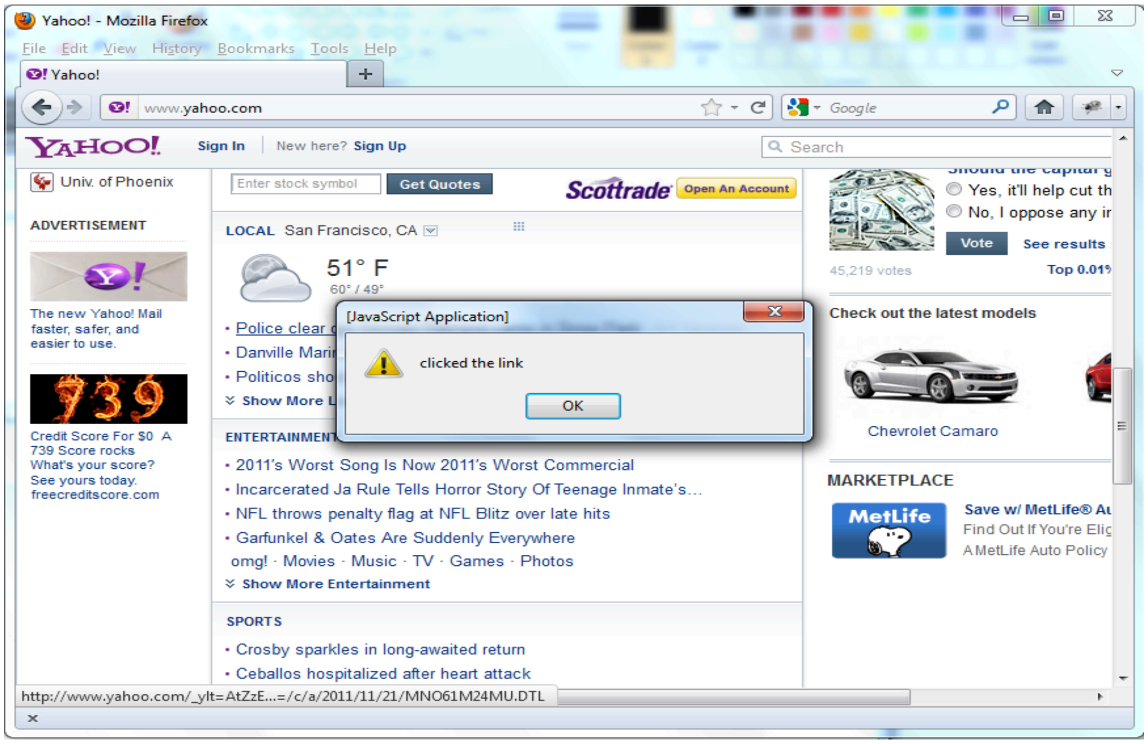
DivX Plus Web Player HTML5 <video> is incompatible with Firefox 8.0.

DivX Plus Web Player HTML5 <video> 2.1.1.94 (disabled)

Increase performance and video formats for your HTML5 <video>

Java Console is incompatible with Firefox 8.0.

Java Console 6.0.29 (disabled)



[JavaScript Application]

clicked the link

OK

http://www.yahoo.com/_ylt=AtZzE...=/c/a/2011/11/21/MNO61M24MU.DTL

3. Deliverable 2: Simple Graph using Canvas/SVG

3.1 Goal

The objective of the deliverable is to use a force directed algorithm (Fruchterman& Reingold, 1991) in calculating the nodes position in the graph and draw it using SVG or Canvas.

3.2 Description

SVG

Scalable Vector Graphics (SVG) is an XML markup language for describing two-dimensional vector graphics. It does not lose any quality if they are zoomed or re-sized.

This is a sample code to draw a circle using SVG

```
<circle cx="350" cy="90" r="50" fill="blue" />
```

And this is the sample code to draw a line using SVG is

```
<line x1="100" y1="50" x2="350" y2="90"
style="stroke:rgb(50,250,0); stroke-width:2" />
```

Canvas

<canvas> is an HTML element which can be used to draw graphics via scripting (usually JavaScript). For example, it can be used to draw graphs, make photo compositions, create animations or even do real-time video processing.

This is a sample code to draw a circle using Canvas:

```
var canvas = document.getElementById("myCanvas");
var context = canvas.getContext("2d");
context.beginPath();
context.arc(50, 50, 20, 0, 2 * Math.PI, false); //First 3
parameters are centerX, centerY, radius
context.fillStyle = "#8ED6FF";
context.fill();
context.lineWidth = 1;
context.strokeStyle = "black";
context.stroke();
```

And this is the sample code to draw a line using Canvas:

```
var canvas = document.getElementById("myCanvas");
var context = canvas.getContext("2d");
context.moveTo(50, 50); //x and y co-ordinates for point1
context.lineTo(100, 100); //x and y co-ordinates for point2
context.stroke();
```

Force Directed Algorithm

Force directed algorithms are a class of algorithms for drawing graphs in an aesthetically pleasing way. The generally accepted aesthetic criteria are

- Distribute vertices evenly in the frame
- Minimize edge crossings
- Make edge lengths uniform
- Reflect inherent symmetry
- Conform to the frame

For this deliverable, we are using the algorithm proposed by Fruchterman & Reingold, 1991 in the paper "Graph drawing by force-directed placement" to calculate the node positions.

Each node is initialized with a random position, a mass (currently, fixed to value 1), velocity (0, 0) and force (0, 0). The pseudo code for the algorithm is

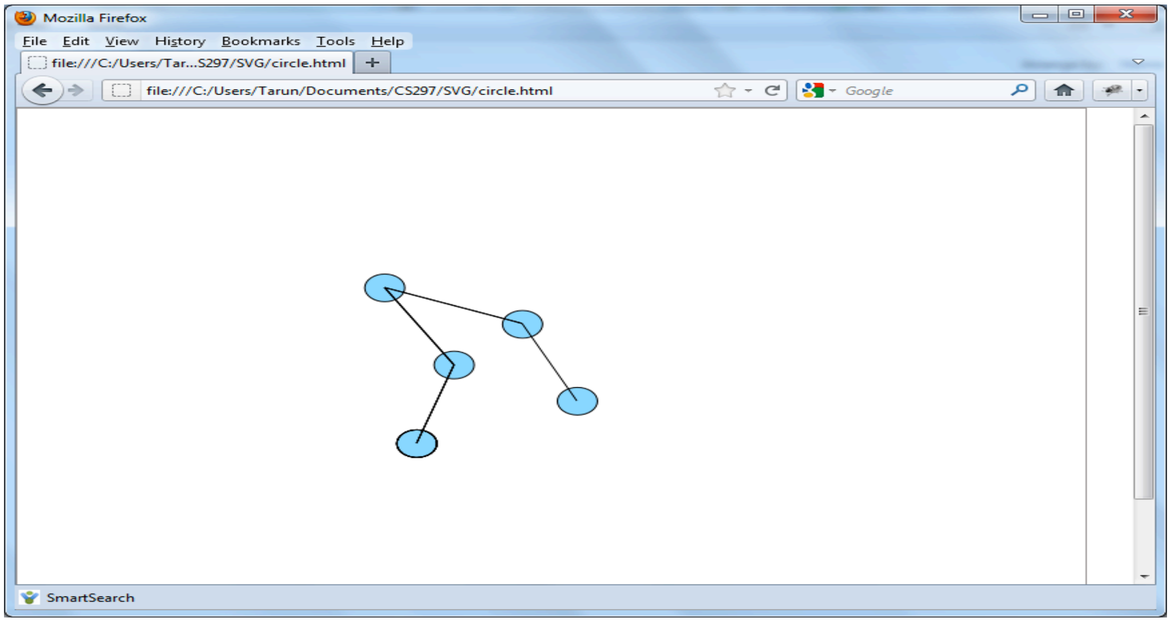
```
Loop

    kinetic_energy = 0;
    for each node
        // running sum of total force on this particular node
        net_force = (0, 0);
        for each other node
                net_force = net_force +
                        Coulomb_repulsion (this_node, other_node);
        next node

        for each spring connected to this node
                net_force = net-force +
                            Hooke_attraction (this_node, spring);
        next spring

        // without damping, it moves forever and the damping
        //constant is fixed to 0.5 (0 < damping < 1)
        this_node.velocity = (this_node.velocity +
                              timestep * net_force) * 0.5
        this_node.position = this_node.position +
                              timestep * this_node.velocity
        kinetic_energy = kinetic_energy + this_node.mass *
                         (this_node.velocity)^2
    next node
until kinetic_energy < 0.01 //A small constant
```

We are using canvas to draw the graph as it felt simpler and easy to manage. This is the snapshot of the graph on which Force Directed algorithm is used.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Users/Tar...S297/SVG/circle.html
file:///C:/Users/Tarun/Documents/CS297/SVG/circle.html
Google
SmartSearch

4. Deliverable 3: Index User Clicks in the Yioop Search Result Page

4.1 Goal

The objective of the deliverable is to send the user clicked URL's in the Yioop search result page to the search engine.

4.2 Description

Currently, the user clicked URL's in the Yioop search result page is not captured. This can be achieved by capturing these data and send to a controller in the Yioop search engine. Then, the controller creates a data file with this information in the schedules folder of the Yioop data. The Yioop search engine indexes the data later.

I have modified the extension developed by Vijaya Pamidi, Dr. Pollett's previous student to capture these data and send it to Yioop. On clicking the link in the Yioop Search result page, the following function gets called.

```
function addData(event) {
      var language = content.document.
                     getElementsByTagName("html")[0].
                     getAttribute("lang");

      if(language == null){
          language = content.document.
                     getElementsByTagName("html")[0].
                     getAttribute("xml:lang");
      }

      var queryURL = window.content.location.href;
      var word = queryURL.split("q=").pop().replace("+"," ");
      var URL = event.target.href;
      var time = new Date();
      var yioopurl = "http://localhost:85/yioop/";
```

```
            var record = word + "|:|" + URL + "|:|" + queryURL
                         + "|:|" + time + "|:|" + language + "\n";
            var present = queryURL.indexOf("yioop.com");

            if(present != -1) {
                uploadAsyc(yioopurl, record);
            }
      }
```

The uploadAsync() sends the data to Yioop server.

```
      function uploadAsync() {

            var params = "c=traffic&u=root&p=&a=toolbarTraffic&b="
                          + record;
            var xhr = new XMLHttpRequest();
            xhr.open("POST", url, true);
            xhr.setRequestHeader("Content-Type", "application/x-www-
            form-urlencoded");
            xhr.send(params);
      }
```

In the Yioop Server part, there is a traffic_controller.php which reads the parameter sent by the extension code and writes to the schedules folder with the below folder structure

`ToolbarData<timestamp>/<timestamp>/At<timestamp>From<localAddress>WithHash<HashValue>.txt`

5. Conclusion

As the part of CS297, I have created components which would be helpful in the final implementation of CS298. The Firefox extension with preference pane created in deliverable1 acts as building block for storing the user provided data. The simple graph created in deliverable2 has the implementation of the force directed algorithm. It can be extended to add animation and legends to provide detailed information about the visited URL's. The third deliverable indexes the user clicked URL's in the Yioop search result page.

In CS298, I will be adding animation to graph and show the URL information in the generated graph. I will also implement a theoretically sound way of computing the relevance of the user visited URL's based on modern web reputation system.

6. References

[1] Building an extension. (2011). Retrieved December 05, 2011, from https://developer.mozilla.org/en/Building_an_Extension

[2] SVG. (2011). Retrieved December 05, 2011, from https://developer.mozilla.org/en/SVG

[3] Canvas Tutorial. (2011). Retrieved December 05, 2011, from https://developer.mozilla.org/en/Canvas_tutorial

[4] Fruchterman, M. J., and Reingold, M. 1991. Graph drawing by force-directed placement. Software – Practice and Experience, 21, 1129-1164

[5] Smart Search: A Firefox Add-On to Compute a Web Traffic Ranking. (2011). Retrieved December 05, 2011, from http://www.cs.sjsu.edu/faculty/pollett/masters /Semesters/Fall10/vijaya/CS%20298%20Project%20Report.pdf

[6] Force-based algorithms (graph drawing). (2011). Retrieved December 05, 2011, from http://en.wikipedia.org/wiki/Force-based_algorithms_(graph_drawing)