

Perseus Web2 Project Demonstration

cs467 Capstone

Date: Dec 1, 2017

Perseus Web2 Team:

Ben Fondell (fondellb)

David Hijirida (hijiridd)

Greg Niebanck (niebanckg)

Source Code:

We have included our source code in a file called "perseus_web2.zip". Unzipping this file will show the following components our team has written:

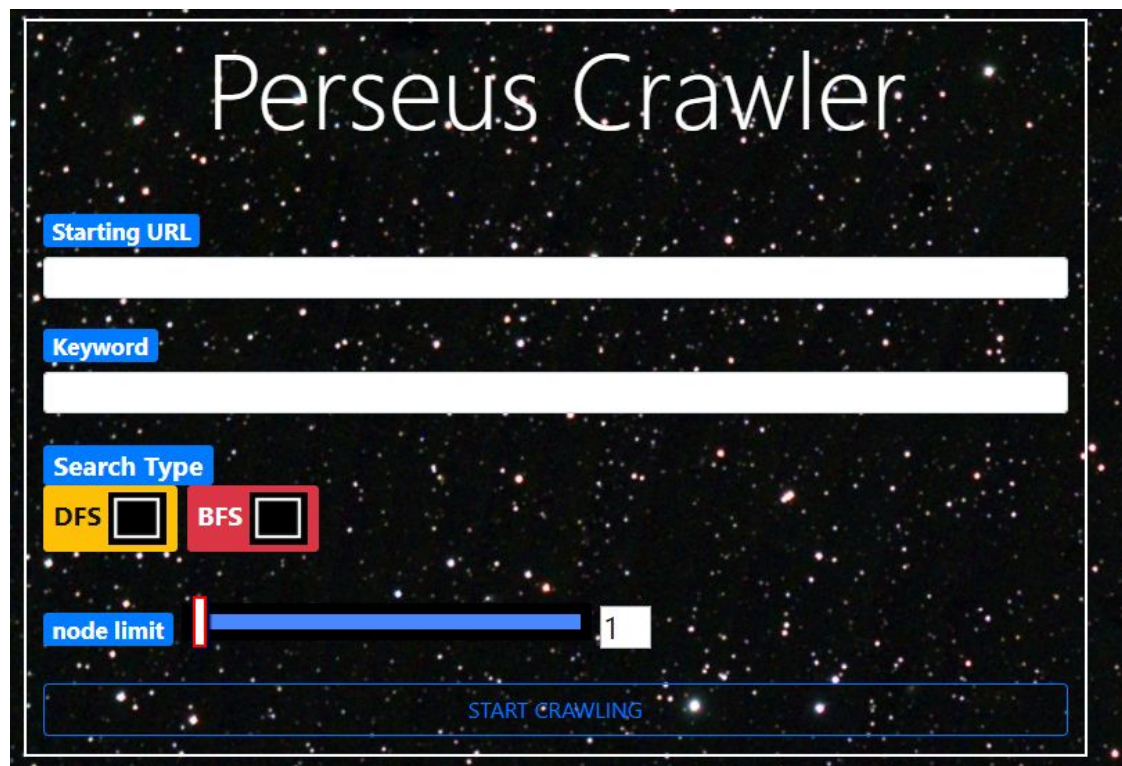
- **File 1: "cs467PerseusWeb2_HTMLParser-master"** - Includes all the files in github (https://github.com/hijirida/cs467PerseusWeb2_HTMLParser). The primary files required are:
 - **htmlparser.js** - The primary node javascript file that listens on port 3001 and handles the primary use case for clients to send a target url, and to (1) receive back a list of urls found on the target url web page, (2) return the title found on the target url web page, and (3) return an indicator whether a keyword is found on the target url web page.
 - **package.json** - provides the list of required node packages
- **File 2: "PERSEUS-WEB-CRAWLER-master"** - includes a directory called ServerCode/ which contains all the files in github (<https://github.com/GNIEBANCK/PERSEUS-WEB-CRAWLER/>). The primary files required are:
 - **mainServer.js** - contains code for generic error handling and middleware initialization.
 - **Package.json** - provides the list of required node packages
 - **./views/home.handlebars** - contains the html for the single page application. This provides the space for all form data and graphics output.
 - **./routes/index.js** - contains the routes for the serving the main page, routes for controlling the data flow between the client and the parsing API, and houses the collection manager functions. The collection manager functions create, manage, and encapsulates crawl data. This is the main interface between client requests and the parsing API.
 - **./public/scripts/home.js** - contains client-side javascript for the single page web application. These functions manage form data, and houses the traversal algorithms. This file provides code for the UI, interfaces with the server, manages collection results, and send them to the graphics modules.
 - **./public/scripts/graphics_DFS.js** - converts traversal data into graphics for DFS searches.
 - **./public/scripts/graphics_DBS.js** - converts traversal data into graphics for BFS searches
 - **./public/scripts/create_graph.js** - provides traversal animation for searches.
 - **./public/scripts/.*** - the rest of the files in this directory are middleware distribution libraries for Bootstrap, D3, JQuery, and Cytoscape.
 - **./public/css/main.css** - stylesheet that contains custom CSS rules for the single page application and the graphics module that are not provided from Bootstrap.
 - **./public/images/.*** - contains image content for the application.

A note regarding compiled binaries. Our team did not create code that required an precomputation, but did make use of 3rd party binary code. The most significant binary we leveraged was PhantomJS (<http://phantomjs.org/>) which provides a "headless" back-end browser-like capability to evaluate web pages created with html and javascript. While there were a number of node packages that tried to emulate a headless browser (e.g., <https://github.com/cheeriojs/cheerio> or <https://github.com/tmpvar/jsdom>), they were not able to consistently render the full web page html across a broad number of website (the most troublesome happened to be yahoo.com, which phantomJS could render easily)

Instructions:

Our website can be found at this url:

<http://ec2-54-148-189-221.us-west-2.compute.amazonaws.com:16000/>

The image shows a web interface for 'Perseus Crawler' with a starry space background. At the top, the title 'Perseus Crawler' is displayed in a large, white, serif font. Below the title, there are four input fields: 'Starting URL' (a long white text box), 'Keyword' (a long white text box), 'Search Type' (two radio buttons, one labeled 'DFS' in a yellow box and one labeled 'BFS' in a red box), and 'node limit' (a blue slider bar with a white handle positioned at the far left, labeled '1'). At the bottom of the form is a large blue button with the text 'START CRAWLING' in white capital letters.

Starting a crawl

- Under "Starting URL", input a valid url to a website.
 - Note: the following url forms are accepted: "<https://google.com>" or "www.google.com" or "google.com". You may add an keyword to the crawl by entering it in the text field below the blue keyword label. If the keyword is found on any web page, we stop crawling even if we have not hit the node limit. Entering a keyword is optional.
- Choose your search algorithm by selecting appropriate box under the "Search Type" label. Check the box inside the red BFS label for a breadth first search, or the box inside the yellow DFS label for a depth first search.
- Use the slider next to the node limit label to select number of sites to visit during the crawl.
- Start the crawl by clicking the start crawling button at the bottom of the form.
 - Entering invalid urls or a site that can't be reached (or rendered in the phantomjs back-end, headless browser) will result in a message on the form stating, "*The starting url is invalid*"
 - Neglecting to choose a search algorithm will result in a message on the form asking the user to select a search type.
- The parameters entered for each search are saved in a cookie on the client's browser. **These cookies last one hour** and are used to populate the "Search History" table. (This table appears below the form for users who have the cookie in their browser data)

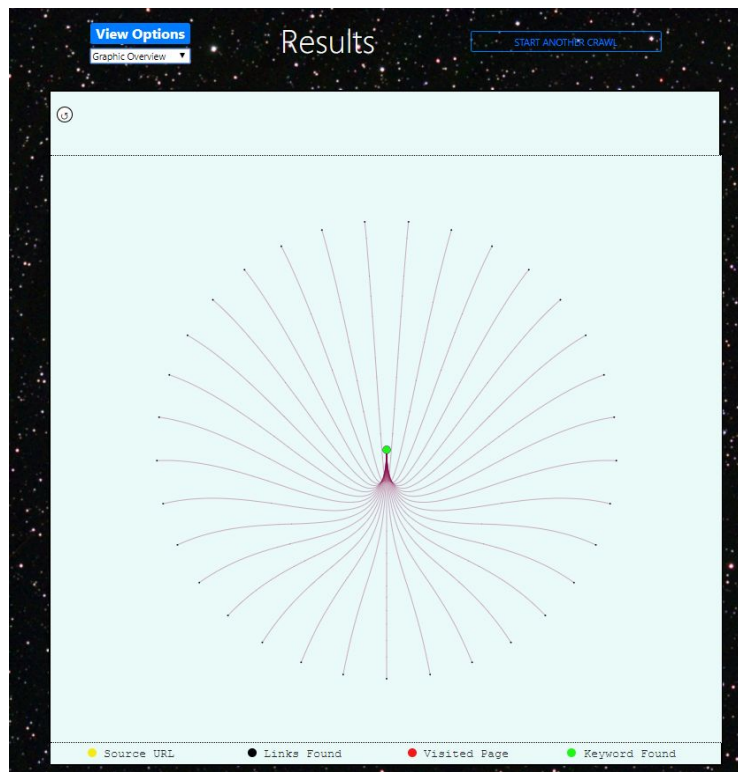
Search History

#	Starting URL	Keyword	Search Type	Limit
1	http://www.google.com		BFS	9
2	http://www.google.com		DFS	3
3	http://www.google.com		BFS	9

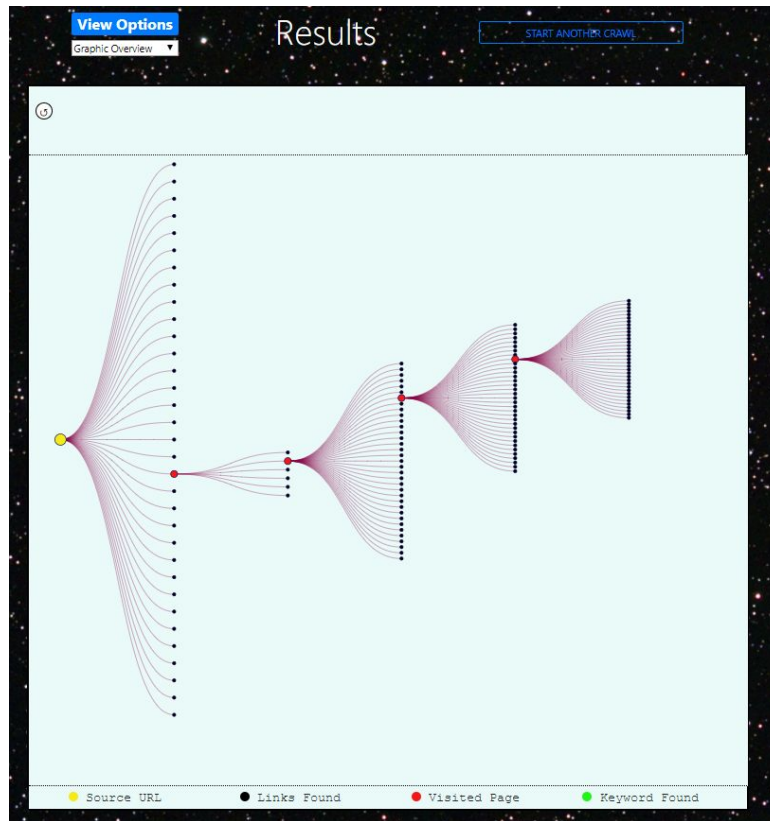
- Using the Search History Table: The Search history table may also be used to initiate previously conducted crawls. The rows of the table are sorted with the most recent searches listed at the top of the table. Clicking on the rows in the table will initiate a repeat search with all the corresponding search parameters.
- Progress is indicated by a progress bar which appears once the crawl has been initiated. Once the progress bar is full the page will display the results.
 - Note: If you enter a keyword in your crawl, the search will halt if the keyword is found on a site which was visited during the crawl.

Viewing the Results

BFS search data is represented as a circular expansion of nodes emanating from the root node displayed in the center of the graph.

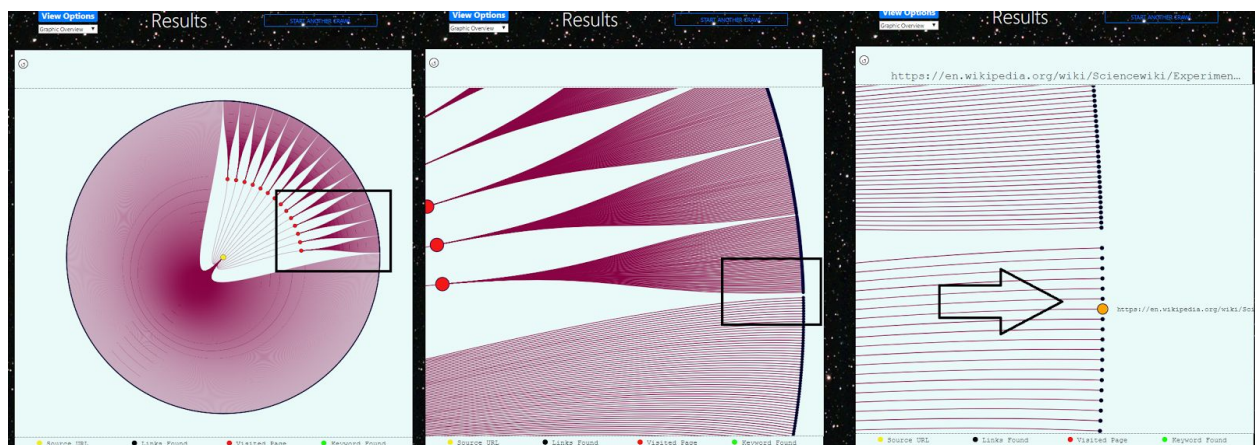


DFS search data is represented as a tree with the root node displayed on the left and child nodes branching to the right.



Graphic Overview:

For both search types, the default viewing option is 'graphical overview'. The display plots all the sites visited during the crawl as well as all of the available links found on those pages. Each node is color coded to identify the root node, sites visited during the crawl, and links to sites that were not visited. The node with content containing the keyword will be colored green. A key for deciphering the color code is displayed beneath the graph.



The graph may be zoomed and panned to better navigate the nodes. You may un-zoom and recenter the graph using the reset button in top left corner. The url details will be made visible in the top center of the viewing area

when hovering the cursor over a node. Clicking a node will display the associated page in a new tab in the user's browser.

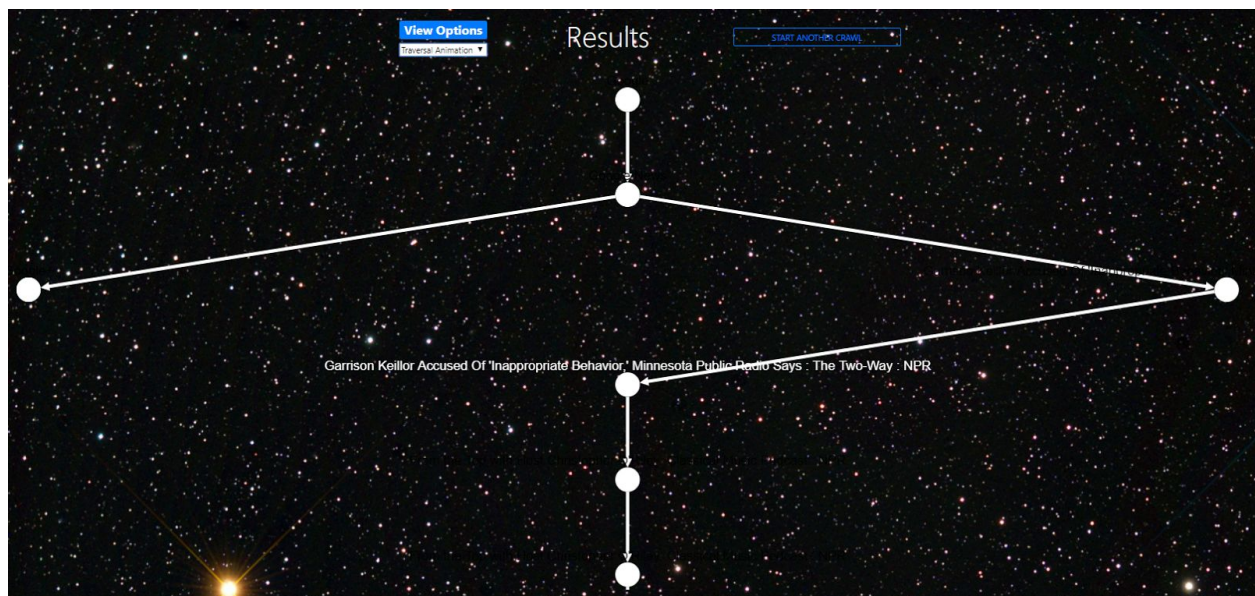
Node List:



#	URL	Parent Row	Children
1	http://www.google.com	root	33
2	https://photos.google.com/?tab=wq&pageId=none	1	6
3	https://www.google.com/chrome/?hl=en_US	2	35
4	https://www.google.com/chrome/?hl=en_US/plus.google.com/+Chrome	3	35
5	https://www.google.com/chrome/?hl=en_US/plus.google.com/%20Chrome/www.youtube.com/user/googlechrome	4	35

Viewing options may be adjusted using the drop down menu at the top left of the screen. The user may select a "Node List" which shows a table starting with the root url and the sites that were visited and analyzed during the crawl. Each row displays the URL of the visited site, the number of links on that site, and the row number for the parent site. Clicking a row in the node list will open a new window in the browser with that corresponding URL.

Traversal Animation:



The final view option called "Traversal Animation" which shows the order in which the nodes were visited in relationship to each other. This option is considerably less complex than the graphic overview. This option is a product of extra slack time and aims to add to the constellation theme in an aesthetic way by displaying the traversals as connected dots in space. Each website's title is displayed above the node upon mouse over. Clicking on a node will open the website in a new tab in the browser.

Use the 'Start Another Crawl' button to return set up a new new crawl form. This is a single page application so using the back button will navigate the user away from the page.