Omega - WEB2: Graphical Crawler

Team Members: Michael Byrne, Eric Olson, Xiaohan Zeng
CS419_400 - Spring 2016 - Brewster

Live Website - https://web.engr.oregonstate.edu/~olsoeric/CS419/Omega/

Project Demonstration:

This report describes how to use the Omega Team's Web Crawler application. The primary function of the Omega Web Crawler is to, of course, execute web crawls. This web crawl process begins at the user provided starting URL, follows a selected search method to evaluate identified external links, and completes when either a requested # of results are found or the web crawl encounters a website containing an optional termination keyword/phrase. The results of the web crawl will be displayed to the user for review in both a data table format, as well as a visualization.

This report will outline the code submission, basic web crawl use instructions and a more detailed review of some of the more interesting functionality and features available.

Code Review

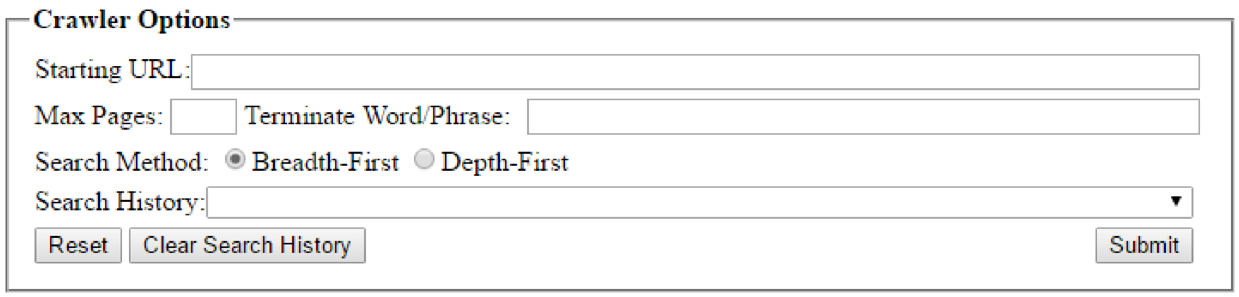
Omega_WebCrawler.zip contains the following source code files available for review:

- **index.html -** Website HTML landing page.
- **JavaScript/omega_JS.js -** Primary website interactive functionality
- **JavaScript/sketch.js -** Visualization functions
- **JavaScript/tree.js -** Visualization node spacing algorithm
- **JavaScript/p5.min.js -** p5.js library
- **Crawler/omega_DS.py** - Object and helper functions for crawler
- **Crawler/omega_webcrawl.py -** Web crawler backend
- **Crawler/process.php -** Middleware layer
- **CSS/style.css** - CSS Styling for website
- **IMG/*.*** - Images and GIF's used by website

The URL provided at the start of this report is a live hosting of the provided source code. For one to setup their own hosting they would need to copy the source code files in the provided structure into their own host server with HTML, PHP and Python support. The only code change required would be in the omega_JS.js file. Line 125 is the call to the middleware PHP and would need to be updated to the correct URL for the process.php location.

Basic Functionality

All Omega Web Crawler functionality is driven from the website and use of the application begins with the user input form on the main page shown below:



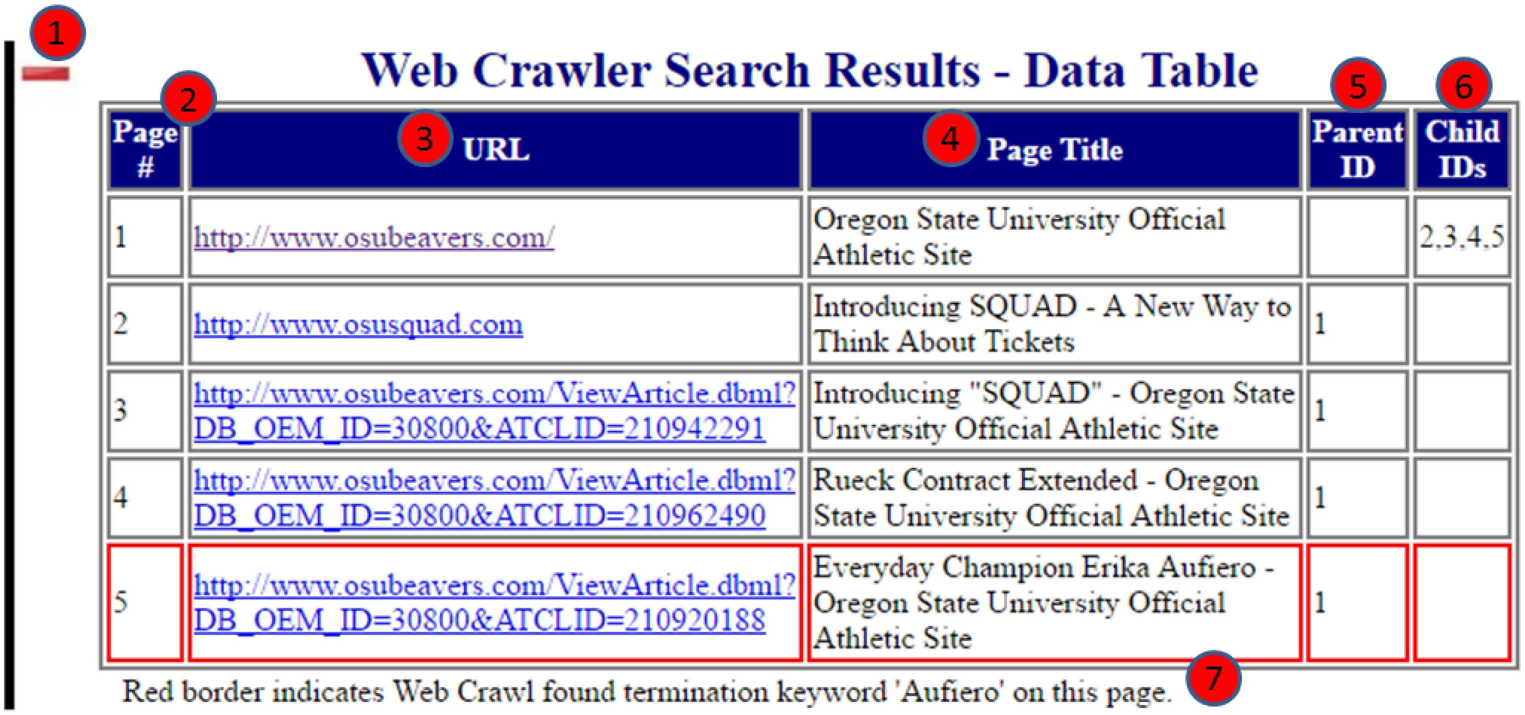
The functionality here is hopefully fairly intuitive but each field will be covered in a more detail to highlight some features that may not be obvious upon first use.

- **Starting URL** - The website from which a user wishes to begin their web crawl.
    - Starting URL is mandatory.
    - Field will accept URL's with or without Scheme. (www.fish.com vs. http://www.fish.com)
    - Invalid URL's will be validated by the crawler and return a unique error message.
- **Max Pages -** The number of identified pages at which the Web Crawler should terminate.
    - Max Pages is mandatory.
    - Field only accepts numeric values.
    - Entries are validated and if not in valid range (1-25) will return a unique error message.
    - Upper limit (25) is enforced by the backend, but can be updated at client's direction.
- **Terminate Word/Phrase -** A keyword, which if found will terminate the web crawl.
    - Termination keyword/phrase is optional.
    - Termination keyword/phrase is not case sensitive.
    - Termination supports single or multiple word phrases.
    - Only page content is searched, HTML tags, links, Image filenames etc are not considered.
- **Search Method -** The pattern in which the web crawl will proceed.
    - Search Method radio button selection is mandatory.
- **Search History -** A listing of previously submitted requests stored locally for quick retrieval.
    - Search history is only used/available if local storage is allowed per browser settings.
    - Search history is persistent from session to session and through browser closure.
    - Searches utilizing the local storage records will not invoke the backend web crawler.
    - Any successful web crawls will be stored in the history automatically.
    - Selecting a recent search from the drop down will disable other input fields to prevent user confusion. Conversely deselecting (selecting blank) a record, or clicking the reset button will re-enable input fields.
    - A new web crawl request matching the criteria of a web crawl found in the local storage will use the locally stored data instead of invoking the backend web crawler.

    - Search history records are uniquely keyed on a combination of all search criteria (starting URL, max pages, termination word/phrase & search method)
    - Due to the shared nature of local storage by domain all Omega Web Crawl records stored to local storage have a unique project identifier. This ensures that other projects hosted @ web.engr.oregonstate.edu and also storing local data will not interfere.
- **Reset** - This button will clear all input fields, and remove any prior data table or visualizations on the page.
- **Clear Search History -** This button will remove any local storage records stored for this application.
    - Other local storage records related to web.engr.oregonstate.edu but not created by this application will not be deleted.
- **Submit -** Starts a new web crawl with the user input parameters.

__Web Crawler Results - Data Table__

Upon a successful web crawl, or search history selection a data table detailing the pages visited



Web Crawler Search Results - Data Table

Page #	URL	Page Title	Parent ID	Child IDs
1	http://www.osubeavers.com/	Oregon State University Official Athletic Site		2,3,4,5
2	http://www.osusquad.com	Introducing SQUAD - A New Way to Think About Tickets	1	
3	http://www.osubeavers.com/ViewArticle.dbml?DB_OEM_ID=30800&ATCLID=210942291	Introducing "SQUAD" - Oregon State University Official Athletic Site	1	
4	http://www.osubeavers.com/ViewArticle.dbml?DB_OEM_ID=30800&ATCLID=210962490	Rueck Contract Extended - Oregon State University Official Athletic Site	1	
5	http://www.osubeavers.com/ViewArticle.dbml?DB_OEM_ID=30800&ATCLID=210920188	Everyday Champion Erika Aufiero - Oregon State University Official Athletic Site	1	

Red border indicates Web Crawl found termination keyword 'Aufiero' on this page.

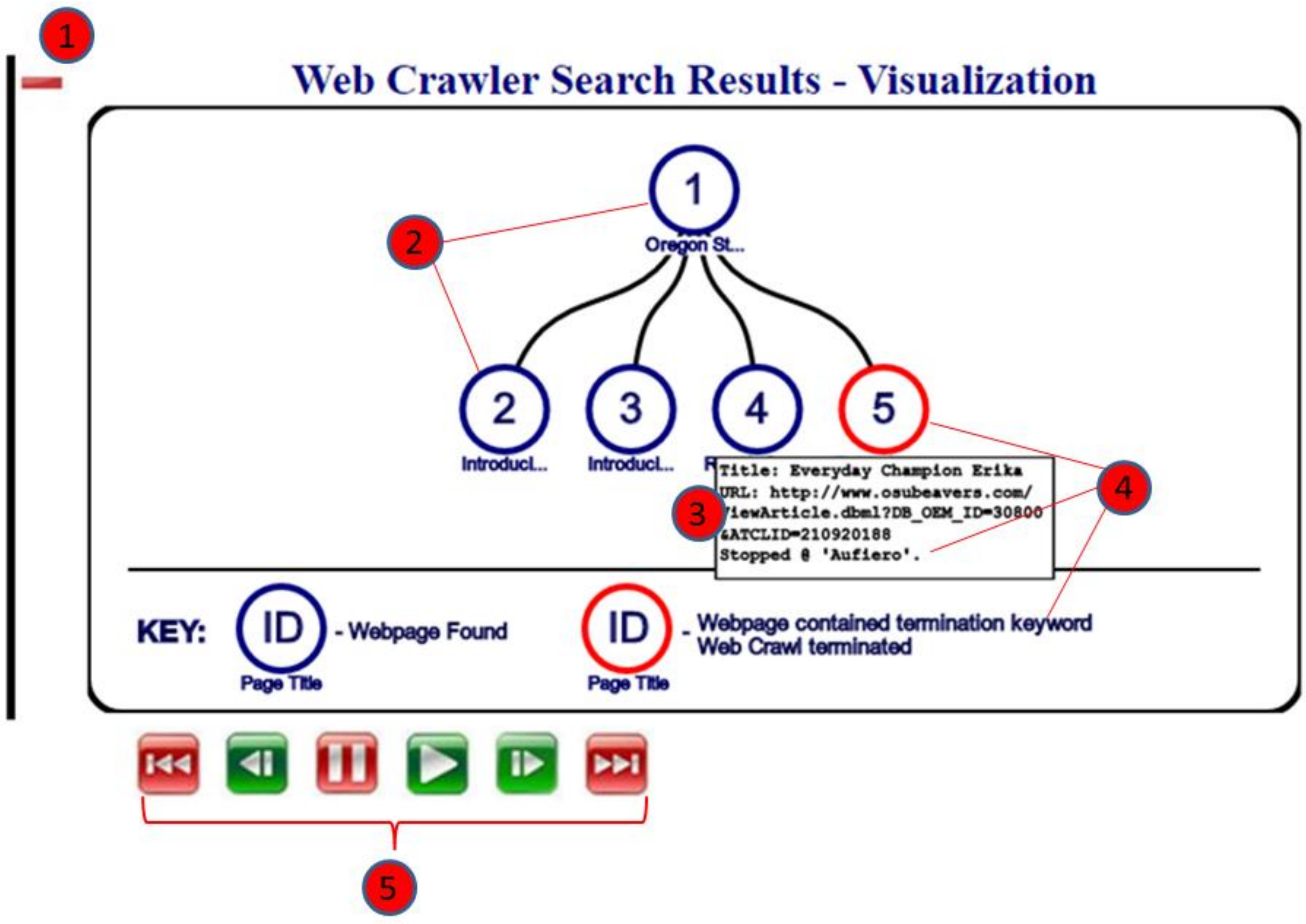
will be displayed to the user. An example of a successful web crawl data table is shown below:

1. **Collapse/Expand** - The entire data table section can be minimized to save space on the page. Once minimized this icon changes to a plus sign which will return the data table to the page.
2. **Page ID** - This is a unique identifier for each webpage identified in the web crawl. These also reflect the order in which the pages were identified during the web crawl.
3. **URL -** The absolute URL of the identified pages. This may have been calculated from a relative href link. This link is clickable and will take a user to the page in question in a new browser tab.
4. **Page Title -** The page title for each identified website.

5. **Parent ID -** This is the unique page ID for the page on which a link to this record was found.
6. **Child IDs** - A listing of all unique page ID's for pages to which this website had an external link.
7. **Terminated Search -** If web crawl encountered the optional termination keyword, this will be called out by red highlighting around the page in which the keyword was found.

__Web Crawler Results - Visualization__

Upon a successful web crawl, or search history selection a graphical representation of the pages visited and the order traversed will be displayed to the user. The results of the web crawl are displayed as a general tree where identified pages are represented as nodes, and connecting links as edges. An



```
Title: Everyday Champion Erika
URL: http://www.osubeavers.com/
/iewArticle.dbml?DB_OEM_ID=30800
&ATCLID=210920188
Stopped @ 'Aufiero'.
```

example of a successful web crawl visualization is shown below:

1. **Collapse/Expand** - The entire visualization section can be minimized to save space on the page. Once minimized this icon changes to a plus sign which will return the data table to the page.
   - Minimization does not interfere with an ongoing 'web crawl animation' discussed below, and progress can be viewed once visualization is re-expanded.
2. **Node's Identified** - All nodes identified are displayed with their unique page ID, and an abbreviated page title.

3. **Additional Info -** Upon mousing over any existing node an expanded data box will be created (node #5 expanded above).
    - Expanded data includes full page title
    - Expanded data includes full URL
    - Expanded data may include the termination keyword if search was terminated
4. **Terminated Search -** If the web crawl encountered the optional termination keyword the node in which the keyword was found will be highlighted in red, and the expanded data box for this node will contain the keyword as well.
5. **Player Controls -** These controls are used to give the user a chance to iterate through the web crawl progression, or watch an animated recreation of the crawl.
    - ⏮ - Moves the visualization back to the beginning with only the starting web-page.
    - ◀ **-** Moves the visualization one step back, removing the last identified node.
    - ⏸ **-** Pauses an ongoing animated web crawl.
    - ▶- Begins an animated playthrough of the web-crawl in the order traversed. Begins at whatever stage the visualization is at currently, and ends when all nodes are displayed.
    - ▶- Moves the visualization one step forward, adding the next identified node.
    - ⏭- Moves the visualization to the end, with all nodes displayed.

<u>Logging and Errors</u>

There are a number of scenarios that can cause an error starting with user input, extending all the way to the backend server. Every effort to address any preventable errors was made, and user friendly error messages are generated otherwise. When an error is generated during a web crawl the error # and error message are presented on screen to the user. Our error matrix is provided below for reference:

Result	Result_text	Generated By:	Notes:
0	"Successful Web Crawl"	Server	
1	"Invalid # of parameters passed. Requires 4 or 5, received <parameter_count>"	Server	Technically this refers to the # of command line parameters passed by the PHP. It does not refer to the number of GET Parameters passed to the PHP page by the GET request.
2	"Invalid Starting URL."	Server	Will check raw URL as well as URL plus valid schemes.
3	"Invalid Method Type, only DF or BF allowed."	Server	
4	"Invalid Max # of Pages, between 1 and 25 allowed."	Server	
5	"Invalid Starting URL"	PHP	
6	"Invalid Method Type, only DF or BF allowed."	PHP	
7	"Invalid Max # of Pages, between 1 and 25 allowed."	PHP	
8	"Web Crawler Server unavailable."	PHP	PHP validation that omega_webcrawl.py server is present.

9	"Web Crawler Error. (Invalid response from server)"	PHP	PHP validation of JSON formatted response from server. (Assumes output, see Result 12 for unhandled Python errors)
10	"Website Encountered With Foreign Language Not Supported. Please Try Another Search."	Server	Initially found issue with some foreign language sites, but now generally handled correctly by server.
11	"localStorage data has been corrupted, please 'Clear Search History' to delete data."	JS - Front End	Triggered when data stored in localStorage is no longer valid JSON.
12	"Error with response from Web Crawler, please try again."	JS - Front End	Triggered when HTTP response from PHP/Server is not properly formatted JSON. Generally caused by unhandled Python error.

In addition to error handling there is built in backend logging. By default the logging is set to 'INFO' and logs are stored in the working server directory in a file called omega_webcrawl_log. The 'INFO' setting keeps a record of incoming search request parameters (URL, # requested, method and termination keyword), as well as the outgoing JSON formatted response text. There is also a 'DEBUG' setting that can be used if the client wants to see verbose output from the crawler as it performs its internal search algorithm. This level of detail is really only intended to find and correct bugs.

Testing and Examples

The Omega team did internal testing using a variety of websites to ensure proper functionality of the web crawl, data transmission, and results display. We found http://www.osubeavers.com/ to be a very interesting visualization at the 25 Max Return limit for both breadth and depth-first searches. In addition to testing on public sites we also generated a number of test environments that allowed us to explore specific crawler functionality with known expected results. For each environment there is a URL to start a new web crawl, and a pre-made visualization showing the expected results, and what about the test environment is unique. Those test sites are available publically and listed below:

Termination:

Starting URL - http://web.engr.oregonstate.edu/~olsoeric/CS419/TERM/PageA.html

Visual - http://web.engr.oregonstate.edu/~olsoeric/CS419/TERM/TERM_Visual.JPG

Breadth First vs Depth First:

Starting URL -https://web.engr.oregonstate.edu/~olsoeric/CS419/DFBF/PageA.html

Visual - http://web.engr.oregonstate.edu/~olsoeric/CS419/DFBF/DFBF_Visual.JPG

Relative URLs:

Starting URL - http://web.engr.oregonstate.edu/~olsoeric/CS419/Rel/A.html

Visual - http://web.engr.oregonstate.edu/~olsoeric/CS419/Rel/REL_Visual.JPG

Example from John Q. Walker Tree Alignment Algorithm:

Starting URL - http://web.engr.oregonstate.edu/~olsoeric/CS419/QVis/0.html

Visual - http://www.cs.unc.edu/techreports/89-034.pdf (Page 17)