Proposal: BackBrowser

Brian Lemke and Jeremy Tzou

CSCE 470 November 1, 2013

Description

We plan to create an intra-domain bidirectional link browser that provides what is essentially a “call hierarchy” tool to users of pre-indexed domains. The goal is to allow users to navigate from their current page to any other page within the same domain that either links to the current page or is linked to by the current page (standard web browsers only give the latter). Essentially, we plan to take the graph structure of a domain and allow users to browse “backwards” along links between pages.

The primary portion of our project is a web server that statically crawls a domain (or set of domains) and builds a directed graph of all the links between pages within that domain. If a given domain is indexed by the server, then a browser extension or AJAX site component can make a request to the server to find a list of all sites that are adjacent to the current site in an undirected graph. This list will be ordered based on the most relevant links, or the links that are most distinctive to the current page.

Target Users

Our tool is primarily meant for medium-size web sites with highly connected, non-hierarchical, static pages (for instance, a fan-created wiki site for a TV show, or a local online news site). Our server is designed to be general-purpose -- it can crawl and index any domain, so long as someone pays for the hosting costs. In a commercial environment, this server would either be offered as an open-source program site administrators could install on their own servers, or as a cloud-based service offered to site administrators.

Our tool would actually be used by ordinary people browsing through web pages. The primary use-case would be users looking to find additional pages that refer to the current page but are not linked to by the current page. For instance, a user browsing on a TV wiki site may see a page for a given character, and then want to view episodes that that character was in (where the episode page links to all the relevant character pages). Alternatively, a user looking at news may want to find articles that cite the article they are currently reading.

Existing Tools and Challenges

While there are existing link analysis tools similar to this, most are oriented towards site administrators or researchers, rather than general users just trying to browse a site. These services generally allow you to enter a URL, then provide a detailed list of all sites and domains it can find that link to the URL. These services provide analytics rather than a browsing experience. The primary differences between our tool and existing tools are:

1. Our tool will focus on intra-domain rather than inter-domain links, providing complete results within a limited domain (as opposed to incomplete results over the entire Internet).

2. Our tool is focused on user browsing, rather than site administration or research.
3. Our tool provides ranked results intended to direct users to the most “interesting” adjacent sites.
4. Our tool provides a well-defined server interface, allowing user interface to be implemented as either a browser plugin or site component.

The bulk of our project (and the most difficult portion) will consist of the web crawling and site indexing component (i.e. the static analysis component). We will have to completely crawl a domain, parse links from HTML content, and build a persistent graph of the link structure of the domain. To assist with this task, we will take advantage of various libraries that assist with web crawling and HTML parsing.

System Design

Our system consists of three main components -- web crawling and indexing (offline on the server), site result generation (dynamic on the server), and user display (dynamic on the client). The first two components are implemented as one server application, while the user display can be implemented via multiple different client-side applications (i.e. site component, browser plugin) communicating with the server via a defined interface.

The first step is to take an input domain (for instance, cse.tamu.edu) and crawl the domain, retrieving all publicly available documents (while respecting robots.txt, of course). Each document will be parsed and stored as a set of links to other URLs within that domain, and a directed graph will be constructed showing all the links between each site within the domain. We intend to use existing web crawling and HTML parsing libraries to assist with this portion.

Next, we will run a PageRank-based algorithm to rank the domain’s sites within that domain. In this case, the purpose of the PageRank score is not to measure “trustworthiness” or “quality”, but rather to find the most distinctive sites. That is, the sites with the lowest scores can be reached from the fewest number of sites, so these will naturally be more distinctive results for the few sites that *do* link to them.

Once the sites are ranked, the server will go “online” and accept requests from the client. The client will send the current page, and the server will return all the pages that are adjacent to the provided page in the graph, ranked in reverse PageRank order (so the most distinctive sites are first). The client will then display this list to the user, allowing the user to navigate both forwards and backwards over the graph structure of the domain.

Demonstration

To demonstrate our application, we will choose a handful of reasonable demonstration sites (domains with a limited number of highly-connected pages, such as a fan wiki, a subdomain of tamu.edu, or some such). We will also develop a Chrome browser extension to demonstrate how a client would work. Our server will index the given domains, and then we will allow users to browse the domain using the extension, going both backwards as well as forwards.