**CAP6777 Web Mining**

Homework 1: Web crawler [15 pts, Due: 05/31/2016]

**Part 1:** Question and Answers [0.75 pt/each].

1. What is HTTP? What is HTML? What are their relationships, and how are they related to web mining?

HTTP is an acronym for hyper-text transfer protocol. It is a communications programming specification for systems that send computer-readable documents between each other. HTML is an acronym for hyper-text markup language. This is a standardized document-formatting language that is both computer and human-readable, and which forms the base of web pages displayed by web browsers. HTTP is a system used to exchange HTML documents between computers over a network, and this is the basic operation of the Internet. In web mining, HTTP commands are used to gather huge numbers of HTML documents, which are then analyzed for things like links, visitors, and content. The information gained in this analysis is then used for business, scientific, political, educational, or other purposes.

1. What is a web crawler? What is the taxonomy of the web crawler? Please list major components of a web crawler.

A web crawler is an automated program that follows an algorithmic system to browse web pages and gather useful data from them, particularly for the purpose of indexing. There are two main types of crawlers: universal, which are used in large-scale search engines to examine as many pages as possible, and preferential, which only want to examine the most important pages. Preferential crawlers have two subclasses: focused, which uses a classifier based on labeled examples to organize its searching, and topical, which searches pages based on similarity. The topical crawlers have further subclasses: adaptive, which learn better methods of searching over time, and static, which use a set searching algorithm. There are also further subclasses based on how they crawl the web. The first major components of a web crawler are a list of starting URLs, called seeds, which are used to find other pages and create a list of web pages to search, called the frontier. Another component, the parser, extracts relevant data from the URLs and stores it in a repository. The last component of the web crawler is the stop criterion that tells the crawler when its job is done.

1. To develop a large scale universal crawler, the two major issues include “performance” and “policy”, please identify at least three components on how to improve the performance and how to design good policy, respectively.

To improve performance in a large-scale universal crawler, scalability is required. This means that DNS lookup times must be reduced, network bandwidth and disk throughput must be optimized, and sockets must be asynchronous. A good crawler policy will consider coverage, freshness, and relevance. That is, the crawler will have system for finding as many pages as possible and add new ones when they are created. The crawler will maintain awareness of changes made to previously examined pages and update its data. And finally, the crawler will have a system for determining the importance of a page and delivering relevant answers to a query.

1. What is a spider trap (or crawler trap) in web crawling? Please suggest at least three approaches (heuristics) for a web crawler to identify a spider trap.

A set of web pages that may intentionally or unintentionally cause a web crawler to traverse the same path indefinitely or crash. Some methods to identify spider traps are checking for URLs beyond a certain character length, being wary of sites with a large number of URLs, avoiding URLs with non-textual data types, and detecting dynamic pages.

1. What is Robots Exclusion Standard? Please use a real-world robots.txt example (e.g., <http://www.google.com/robots.txt>) to explain the key fields (allow, disallow, sitemap etc.) and the setting of the robots exclusion protocol.

Standardized text files used by websites to communicate with web crawlers, in the form of a robots.txt file available from the website. The real-world robots.txt file at <http://www.apple.com/robots.txt> has several key fields:

User-agent – specifies the name of the crawler the immediately following rules apply to. This is usually a “\*” to indicate all crawlers, but also special rules are given to specific crawlers as well.

Disallow – tells the crawler which directories of the website are not to be crawled. If the field is empty, all sections are open, the field is “/” then all sections are closed.

Allow – the opposite of disallow, but not commonly used as anything outside of disallowed is implicitly allowed. This does have use if a subsection of the disallowed directories is to be crawled.

Sitemap – an XML file that lists URLs for a website, this field is used to tell crawlers which URLs are available to them. This can also have information on when the URLs are updated and how often.

Host – a field that website mirrors use to indicate their preferred domain.

Crawl-delay – a non-standard field that instructs crawlers to throttle their visits and requests.

1. What is Cloaking and Spamdexing? Please explain how they are used to disguise the webservers and affect a search engine’s ranking.

Cloaking is where a server presents one version of a webpage to a human, but a different version to a web crawler. By manipulating what the crawler sees (many common search keywords) compared to what a human sees (a normal article), a website can artificially inflate its standing in search results.

Spamdexing another way for a website to manipulate its search result rankings. Special webpages are created and filled with repeated keywords, URL links, or other data that a web crawler would see and rank highly in its indices.

1. What is “frontier” in the web crawler? Please suggest two approaches to manage the frontier, and explain the strength and possible weakness of these approaches.

The frontier is a data structure that holds the list of URLs a crawler will visit and index. This list is sorted and can grow very fast. To manage the frontier, a crawler could implement a breadth-first search or a depth-first search. The breadth-first approach tends to get closely-related pages and is likely to find more relevant links, but the frontier list grows rapidly and can become excessively large. The depth-first approach keeps the frontier smaller, but is less likely to stay near related pages and instead get “lost in cyberspace.”

**Part 2:** Simple web crawling practice [6 pts]

The attached java file (Webcrawler.java) includes a simple web crawler, which starts from a user provided URL (specified by the 1st parameter) to crawl and download web pages, and save the content to a local file (specified by the 2nd parameter). Users can also specify the maximum number of web pages N to be download (3rd parameter). If ignored, the default value is N=20.

**Tasks:**

1. Please download the java file, and use Eclipse or Netbean to build a web crawling project. Please provide a seed url and collect at least 50 web pages from the web crawler [1 pt]
   1. Please submit the original seed URL, and also include all web pages (in one file).

My project is attached as WebCrawler.jar, and is run with the same command line parameters as the provided Webcrawler.java file. From the seed URL <http://ameblo.jp> (a Japanese blogging site), I collected 50 web pages and saved their HTML into the output.txt file, also attached.

1. In the webcrawler.java, the crawler uses a FIFO queue and a Breadth-First-Search strategy to download web pages. Please change the preference of the crawler (i.e., modify the policy), to collect webpages related to a specific topic. For example, you can modify the program to only crawl web pages related to special topics, such as “biology”, “sports”, “information technology” etc. The purpose here is to build a simple preferential crawler, so web crawler will collect documents related to specific topics. [Please note that you must design/implement a heuristic to check whether an URL (i.e., a document) is related to the defined topic or not, before fetching the content of the URL. A simple heuristic is that, if current document is related to the defined topic, all URLs inside the document are related to the defined topic. You cannot fetch the document then discard the irrelevant ones. All fetched documents must be saved to your repository]
   1. Please draw the flow chart (or the pseudo code) of your preferential web crawler design [1 pt], and explain how does your approach/design make web crawling focusing on special topics [1 pt].
   2. Please turn in the revised source code of your program [1 pt]
   3. Given the same set of seed URLs, please use original Webcrawler.java to collect 200 web pages, and also use your new preferential web crawler to collect 200 web pages. For 200 web pages collected from different web crawlers, please check the percentage of the web pages containing the specific keywords, and report the values in your report [1 pt].
   4. Please suggest one additional approach (show your design as a flow chart or a diagram) which may help improve the accuracy of preferential web crawler, so the collected documents are closely related to the topics [1 pt] (No need to implement this design).

**Part 3:** Hands-on Apache Nutch Web Crawler Project Practice [3.75 pts]

Apache Nutch (<http://nutch.apache.org/>) is a powerful and robust open-source web crawler project, which can support large scale (and distributed) web crawling. Apache also has sorl indexing engine (<http://lucene.apache.org/solr/>) which can combine Nutch and Sorl to build a powerful search engine.

In the third part of the homework, you will practice the installation and configuration of Apache Nutch web crawler. You will need to properly install and configure an Apache Nutch web crawler, fetch files from some web servers, and report the results.

Requirement:

1. Please follow the instructions in the “Apache.Nutch.installation.docx” file, install and configure a Nutch web crawler. Please capture three screenshots to show that
   1. Cygwin has been properly installed and running on your computer [0.5 pt]
   2. Notch has been downloaded and configured as showing in the instructions [0.5 pt]
   3. Notch can successfully lunch a web crawler task. [1 pt]
2. Please provide a seed URL (such as [www.amazon.com](http://www.amazon.com), [www.mit.edu](http://www.mit.edu), [www.yelp.com](http://www.yelp.com)) to collect at least 1000 web pages online. Please include the fetched files (the original downloaded files from the web servers) in one file, and submit the file online through blackboard. [1 pt].
3. Please explain the meaning of the Nutch parameters, and change at least three parameters to design three crawling tasks. Please report and explain the crawling results for each parameter setting [0.75 pt].