Digital Library Curriculum Development

Module: 7-f: Crawling

(Draft, Last Updated: 2012/11/28)

1. **Module name:** Crawling

2. **Scope :**

   This module covers the basic concepts of Web crawling, policies, techniques and how this can be applied to Digital Libraries.

3. **Learning objectives:**

   Students will be able to:

   - Explain the basic concepts of Web crawling.
   - Explain the architecture of a simple crawler.
   - Explain policies followed by a crawler.

4. **5S characteristics of the module:**

   - Stream: HTML, CSS, JavaScript and other web design languages, text, videos, images and other Internet media types.
   - Structures: Refers to the structure of the Web, which is in the form of a directed graph called the Web Graph, in which the nodes are Web Pages and the edges are the hyperlinks connecting them.
   - Spaces: The World Wide Web and the physical storage issues faced by the crawlers such as a search engine repository where the data collected by the crawler is stored.
   - Scenarios: Relates to the process followed by the crawler in order to scan the Web and make copies of the pages visited.
   - Societies: Managers, Computer Programmers, and Academics involved with the development and maintenance of Web Crawlers, bots and computer programs, as well as authors of Web sites.

5. **Level of effort required:**

   Prior to class: 2 hours for readings

   In class: 2 sessions of 2 hours each

- 2 hours for understanding the basic functions of a crawler along with its working and evaluation.
- 2 hours for class activity.

6. **Relationships with other modules :**
   - 3-a: Collection development/selection policies
   - 3-e (7-e): Web (push) Publishing
   - 4-a: Information architecture (e.g., hypertext, hypermedia)
   - 7-a: Indexing and searching
   - 9-f: Cost/economic issues

7. **Prerequisite knowledge required:**
   - Basic Knowledge about the World Wide Web.
   - Basic Unix commands

8. **Introductory remedial instruction:**

   None

9. **Body of knowledge:**

   - Introduction:

   - Web Basics:

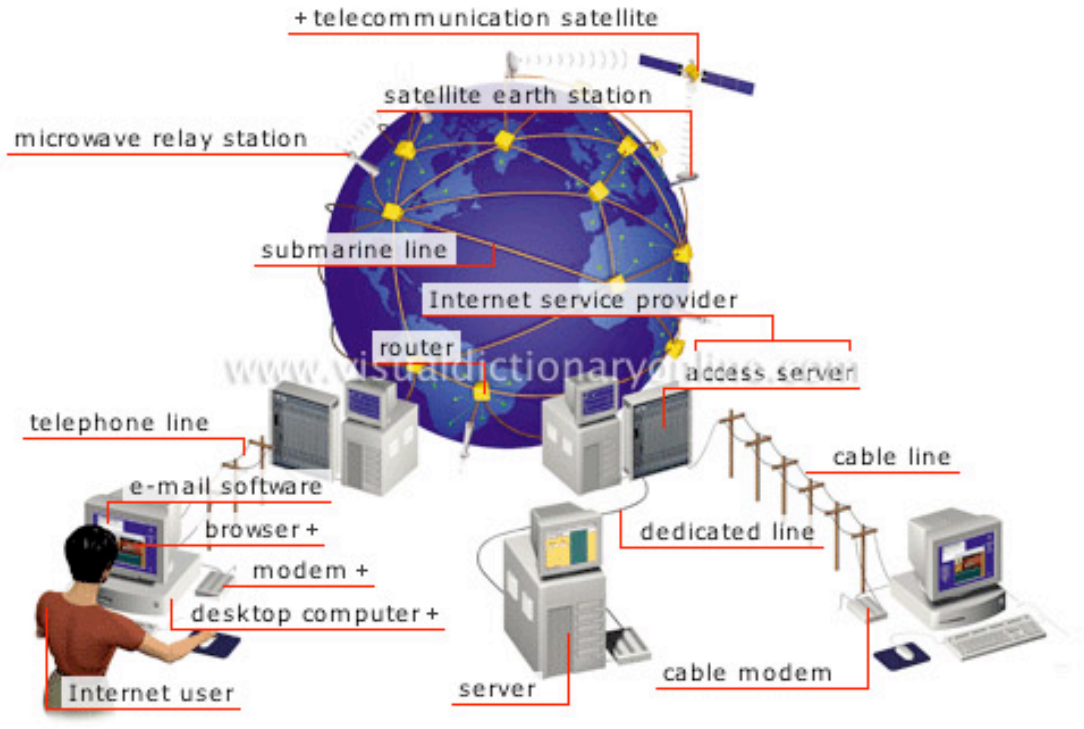


***Figure 1: The Internet**

* The figure is from: http://visual.merriam-webster.com/communications/office-automation/internet_1.php

The internet is a worldwide system of interconnected digital networks that use the standard internet protocol suite (TCP/IP) to connect us all together. It can be thought of as a network of networks consisting of millions of private and public, academic, government, and business networks that are connected together by a large number of electronic and optical network technologies. The internet consists of a vast amount of digital information, the most common of which is the interconnected hypertext documents which makes up the World Wide Web (WWW).

In the late 1980s Sir Timothy John "Tim" Berners-Lee, then a scientist at CERN, invented the World Wide Web, to share information among scientists working in different universities and institutions around the world. However, web usage caught on to such an extent that now a significant amount of the human population has access to it and uses it on a daily basis. The World Wide Web is based on a simple client-server design. The client, which is usually a web browser, communicates with a server through a simple

protocol called the HyperText Transfer Protocol (HTTP) and the server, which receives a request from a client, responds by sending back the desired information in the form of asynchronous payloads (text, images, audio, video, and other forms of information) encoded in a simple language called the hypertext markup language (HTML).

In order to know how a crawler works, we need to know a little more about the World Wide Web and its structure. In this example, we shall consider static web pages and try to visualize how they are connected together to better understand how they are crawled. A static web page is a web page whose content does not change from one request to the next. As a result, we can view the web as a directed graph consisting of static web pages connected together by hyperlinks in which each web page is considered as a node and each hyperlink is considered as an edge.

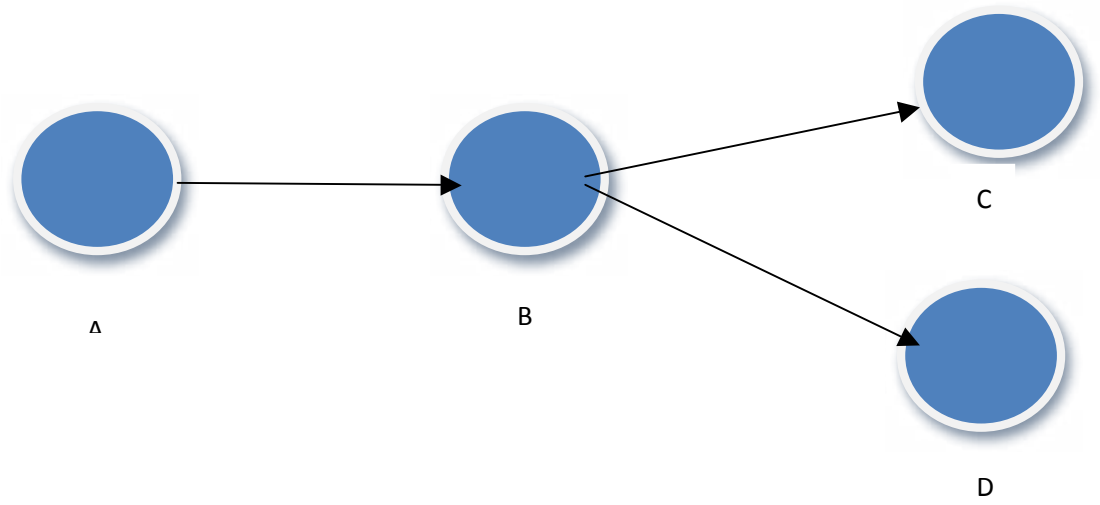


Figure 2: A simple Web Graph

Figure 2 shows a simple example of a web graph, where the nodes are web pages and the edge are hyperlinks. Hence we can consider the World Wide Web as a web graph consisting of such nodes and edges. The link name in node A that connects it to node B is usually encapsulated in an href attribute of the anchor tag which refers to node B. This is how web pages are connected with each other. As we can see, the directed graph is not strongly connected, i.e., we cannot always follow the hyperlinks and proceed from one page to the other. The hyperlinks can be divided into inlinks, which are the links **to** a particular web page, and the outlinks, which are the links **from** a particular web page.

- Web Crawlers:

A web crawler is an automated computer program that browses the World Wide Web by following certain policies as mentioned by its creator. Its main purpose is to create a copy of all the web pages visited by it for later analysis by a search engine. The search engine

usually will index this for providing faster and more efficient searches. A crawler may also be used as a means of automating website maintenance tasks by checking the validity of HTML links of web pages and the code within it. Another way of using the crawler is in a destructive manner. For example, gathering specific types of information from the World Wide Web such as harvesting e-mail addresses (usually for spam) could lead to their removal.

A web crawler is known by many names, like ants, automatic indexers, bots, worms, and Web spiders. The process by which a web crawler browses the World Wide Web is called Web Crawling or spidering. Many services, especially search engines, use this as a means of creating a fresh copy of (a portion of) the World Wide Web in their repository.

- Inner working of a crawler:

A crawler may be thought of as a bot that roams the World Wide Web by following a master's commands. A crawler usually starts with one or more URLs; these are called seeds. It picks a URL from this set and downloads the corresponding web page. This fetched page is then processed to extract the digital information as well as the links (each of which points to another page) from this page. The extracted digital information is fed into an indexer. The extracted links are then added to a URL Frontier, which consists of web pages which the crawler has not explored yet. This entire process can be visualized as traversing a web graph. The URLs from the frontier are recursively visited based on a set of predetermined policies.

- Crawler Architecture:

The process of crawling is a complex process, which can be broken down into a few functional components which work together to make crawling possible. These components are:

The URL Frontier, which is a set of all the URLs which the crawler has not yet explored. It could be called the to-do list of the crawler, that contains the list of URLs not visited by the crawler.

A DNS resolution module, which is used to determine the web server from which the web page is fetched, as specified by the URL.

A fetch module, which is used to retrieve a web page as specified by the URL

A parsing module, which is used to extract the contents and links from a web page.

A duplicate elimination module, which determines whether an extracted link is already present in the URL frontier or has already been fetched by the crawler.

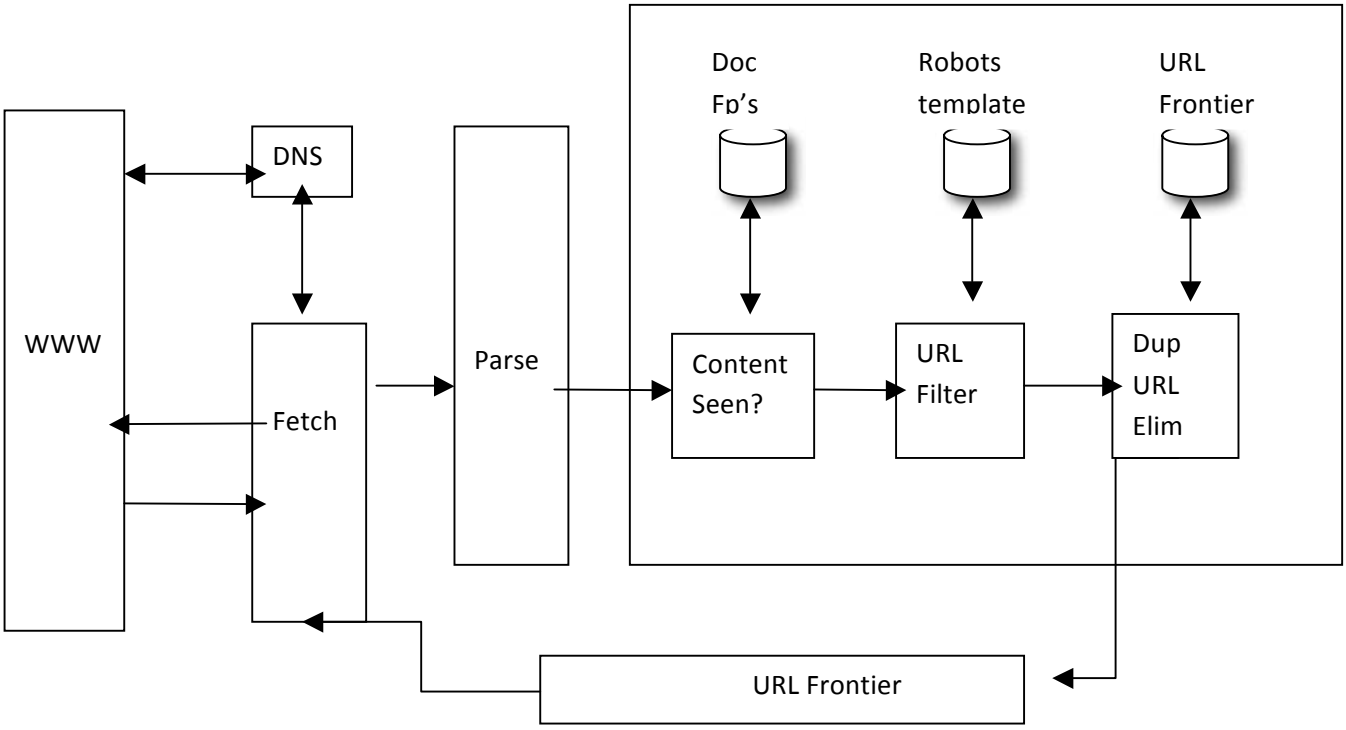


***Figure 3: Crawler Architecture**

*The figure is from: Christopher D Manning, Prabhakar Raghavan, and Hinrich Schutze, "*An Introduction to Information Retrieval*" (pages 443-469)

The process of crawling can be performed by a single system or a distributed system with different nodes running simultaneously an instance of this process.

- Crawling policies:

  In order to crawl the web, we have to be aware of the obstacles that a crawler may face during its crawl. These characteristics of the web are:

- Its very large volume
- Its really fast rate of change and
- Its dynamic page generation

  These characteristics combined together produce a myriad varieties of crawable URLs.

Even with the fastest computers and very high bandwidth, downloading the whole World Wide Web may take months. Due to this constraint, crawlers need to prioritize the pages they download. The fast rate of change implies that due to the constant updating of web pages, by the time a crawler downloads all the pages from a site, it is highly likely that the web page has been updated or deleted, or that new pages have been added to the website.

Another major obstacle that has been faced by many crawlers is the dynamically generated web pages based on user input, which makes it difficult for crawlers to avoid retrieving duplicate content. Endless combinations of HTTP GET (URL-based) parameters exist, of which only a small set returns unique content. For example let us consider a simple online shopping portal, which provides different options to users, as specified through HTTP GET parameters in the URL. If there exist five ways to sort the products (e.g., price, popularity, …), four choices of thumbnail size, and an option to enable or disable product description, then the same set of content can be accessed by forty different URLs, all of which will be linked to the same site. This mathematical combination creates problems for crawlers, as they must parse through all these pages containing relatively small changes in order to fetch unique content.

As Edwards *et al.* noted, "Given that the bandwidth for conducting crawls is neither infinite nor free, it is becoming essential to crawl the Web in not only a scalable, but efficient way, if some reasonable measure of quality or freshness is to be maintained."

The behavior of a crawler is the outcome of a combination of the following policies:

- a *selection policy* that states which pages to download,
- a *re-visit policy* that states when to check for changes to the pages,
- a *politeness policy* that states how to avoid overloading Web sites, and
- a *parallelization policy* that states how to coordinate distributed Web crawlers.

- Selection policy:

Due to the large size of the web, even a large search engine can only download a fraction of the publicly available Web. A study by Lawrence and Giles showed that no search engine indexed more than 16% of the Web in 1999. Hence, a crawler usually downloads only a part of the World Wide Web. Due to this, it is highly desirable that this fraction contains as many “useful” pages as possible.

In order to achieve this, the crawler should be able to recognize the importance of a page and prioritize it. Some methods of assigning the importance of a page are described in the

next subsection. Designing a good selection policy has an additional difficulty; it must work with a fraction of the Web, as the complete set of Web pages is not known during crawling.

- Restricting followed links:

  A crawler may only want to visit HTML (static text) pages and avoid all other internet media types. In order to make this possible, a crawler may make a HTTP HEAD request to determine a web resource's internet media type before requesting the entire page. However, to avoid making numerous HEAD requests, a crawler may examine the URL of a resource and download only those pages ending with a particular sequence of characters like .html, .htm, .asp, .aspx, .php, or a slash. The down side of this strategy is that it may cause numerous HTML Web resources to be unintentionally skipped.

  Some crawlers may also avoid URLs containing the character "?" in order to avoid spider traps that may cause the crawler to download an infinite number of URLs from a web site. However, this strategy is unreliable if the site uses URL rewriting to simplify its URLs.

- Path ascending crawling:

  Some crawlers try to download as many resources as possible from a particular Web site. That way a crawler would ascend to every path in each URL that it intends to crawl. For example, when given a seed URL of http://animals.org/elephant/baby/description.html, it will attempt to crawl /elephant/baby/, /elephant/, and /.

  A path-ascending crawler is very effective in finding isolated resources and resources which would have resulted in no inbound links on a regular crawl.

  Many path-ascending crawlers are also known as Harvesters since they're used to "harvest" or collect all the content (like all the photos from a gallery) from a specific page or host.

- Focused crawling:

  A focused crawler or topical crawler is a web crawler that attempts to download pages that are similar to each other and that are relevant to a pre-defined topic or set of topics. In topical crawling, the crawler assumes that the topic is given whereas in focused crawling, the crawler also assumes that some labeled examples of relevant and non-relevant pages are available. Topical crawling was first introduced by Menczer, whereas focused crawling was first introduced by Chakrabarti.

  A focused crawler ideally would like to download only web pages that are relevant to a particular topic and avoid downloading all others. Hence a focused crawler needs to

predict the probability that a link to a particular page is relevant before actually downloading the page. One possible way of doing this is to consider the anchor text of links as the predictor.

Another way to achieve a focused crawl is to determine the relevance of a page after downloading its contents. The relevant pages are indexed and their contained URLs are added to the URL frontier, and the pages which fall below a relevance threshold are discarded.

- Crawling the Deep Web:

  The deep Web (also called Deepnet, the invisible Web, dark Web, or the hidden Web) refers to World Wide Web content that is not part of the surface Web, which is indexed by standard search engines. These pages are usually created by submitting queries to a database. A regular crawler will usually be unable to find such pages unless pointed to by a link from another page.

  One way to access the deep Web is by using federated search based search engines. Some search tools such as Science.gov are being designed to retrieve information from the deep Web. These tools work by identifying and interacting with searchable databases. Hence they provide us with a means of accessing the content of the deep Web.

- Re-visit policy

  The Web is considered to be very large in volume. It size coupled with its dynamic nature makes it very difficult to crawl. Even crawling a fraction of the web may take a really long time, usually measured in weeks or months. By the time a crawler has finished a single full cycle of its crawl, they Web would have changed in many ways. Many of the crawled pages would have been updated or deleted whereas some pages would have even been added to the Web.

  From the search engines' point of view, having an outdated copy of a resource reduces its credibility and reliability. As a result a search engine would like to have an updated copy of the Web. The most used cost functions used to measure this are freshness and age.

  Freshness: This is a binary measure that indicates whether a copy is accurate or not. Hence its value could be either 0 or 1.

  Age: This is a measure which indicates how outdated the local copy is.

One of the main objectives of a crawler is to keep the average freshness of pages in its repository as high as possible, or to keep the average age of pages as low as possible. These two objectives cannot be directly compared. In the first case, the crawler is just concerned with how many pages are out-dated, while in the second case, the crawler is concerned with how old the local copies of pages are.

Next we shall look at two simple revisiting policies:

- Uniform Policy: Implies re-visiting all pages in the collection with the same frequency, regardless of their rates of change.
- Proportional Policy: Implies re-visiting more often the pages that change more frequently.

- Politeness policy:

A crawler is capable of retrieving data more quickly and in greater depth than a human can. As a result they can have a crippling effect on the performance of a site. Since a crawler can access multiple pages at the same time at speeds much larger than that designed by the website (for humans), a server would have a really hard time keeping up with these requests, especially if multiple crawlers access the website at the same time.

As noted by Koster, the use of Web crawlers is useful for a number of tasks, but comes with a price for the general community. The costs of using Web crawlers include:

- network resources, as crawlers require a considerably higher bandwidth and operate for longer periods of time with a high degree of parallelism;
- server overload, since crawlers can access servers at a much higher rate than an average human can;
- poorly-written crawlers, which can crash servers or routers, or which download pages they cannot handle; and
- personal crawlers that, if deployed by too many users, can disrupt the whole network and even the Web servers.

To tackle some of these problems, usually Web servers have both implicit and explicit policies which state the rate at which a crawler can visit a website. These politeness policies must be respected by the crawler. An example of this is the robots exclusion protocol, which we discuss below. We could also specify a "Crawl-delay:" parameter in the robots.txt file in order to indicate the delay between requests.

- Parallelization policy:

- Multithreaded Crawler:

  Sequential crawling may be inefficient, as most of the time either the CPU is idle (during disk or network access) or the network interface is idle (during CPU processing). In order to tackle this problem we use multithreading, which optimizes CPU utilization as well as the network bandwidth. In a multithreaded crawler, each thread locks the URL frontier while selecting a URL that it will crawl. After it has picked a URL, it unlocks the frontier so that other threads can access it and pick a URL. While a URL is added to the frontier, the frontier is again locked to maintain consistency. The locking is necessary in order to prevent duplicates, or crawling the same URLs, since the frontier is now shared among many crawling loops.

- Keeping Crawlers at bay:

  Many hosts on the World Wide Web place certain portions of their web site off limits to a web crawler. This message is communicated to a crawler through a robots.txt file. This standard is known as the Robots Exclusion Protocol. The robots.txt file is placed at the root of the URL hierarchy of that site so that the crawler reads it at its first contact with a particular site. Below is an example of a robots.txt file that disallows all crawlers, except the crawler called "myCrawler," from visiting the URLS that are placed in the file hierarchy starting with the name "/mysite/privateFiles/"

```
User-agent: *
Disallow: /mysite/privateFiles/

User-agent: myCrawler
Disallow:
```

A simple robots.txt file.

- URL Normalization:

  Another important process that we should be aware about is URL normalization. URL normalization is the process by which a URL is modified or standardized as required. URL normalization is used to transform a URL into a normalized or canonical form in order to determine if two syntactically different URLs are equivalent. Web crawlers use URL normalization in order to avoid adding duplicates to the URL frontier.

- Evaluation of a Crawler:

  A crawler may be classified as a good crawler depending on the type and the quality of information it fetches. Since the World Wide Web is dynamic and a website may update its pages periodically, it is important that a crawler fetches pages that are "useful". However, a major problem that a crawler faces is recognizing these useful pages. Since a crawler is an automated program, it should be capable of understanding what the contents of a web page mean and should classify them as important or not. Hence in order to evaluate a crawler, we need to see the useful of the pages fetched by the crawler as well as the crawl time of a crawler.

- Web Page Importance:

  These are some of the methods that have been used in order to access the importance of a web page and hence its "usefulness":

- Keywords in documents: If a page contains some or all the keywords in a particular query, the page may be considered as relevant. We may also consider the frequency of occurrence within the page.

- Similarity to seed pages: The pages corresponding to the seed URLs are used to measure the importance of the crawled pages. This is done by calculating the cosine similarity of a document (containing seed pages combined together) and the crawled page. This is used as the page's relevance score.

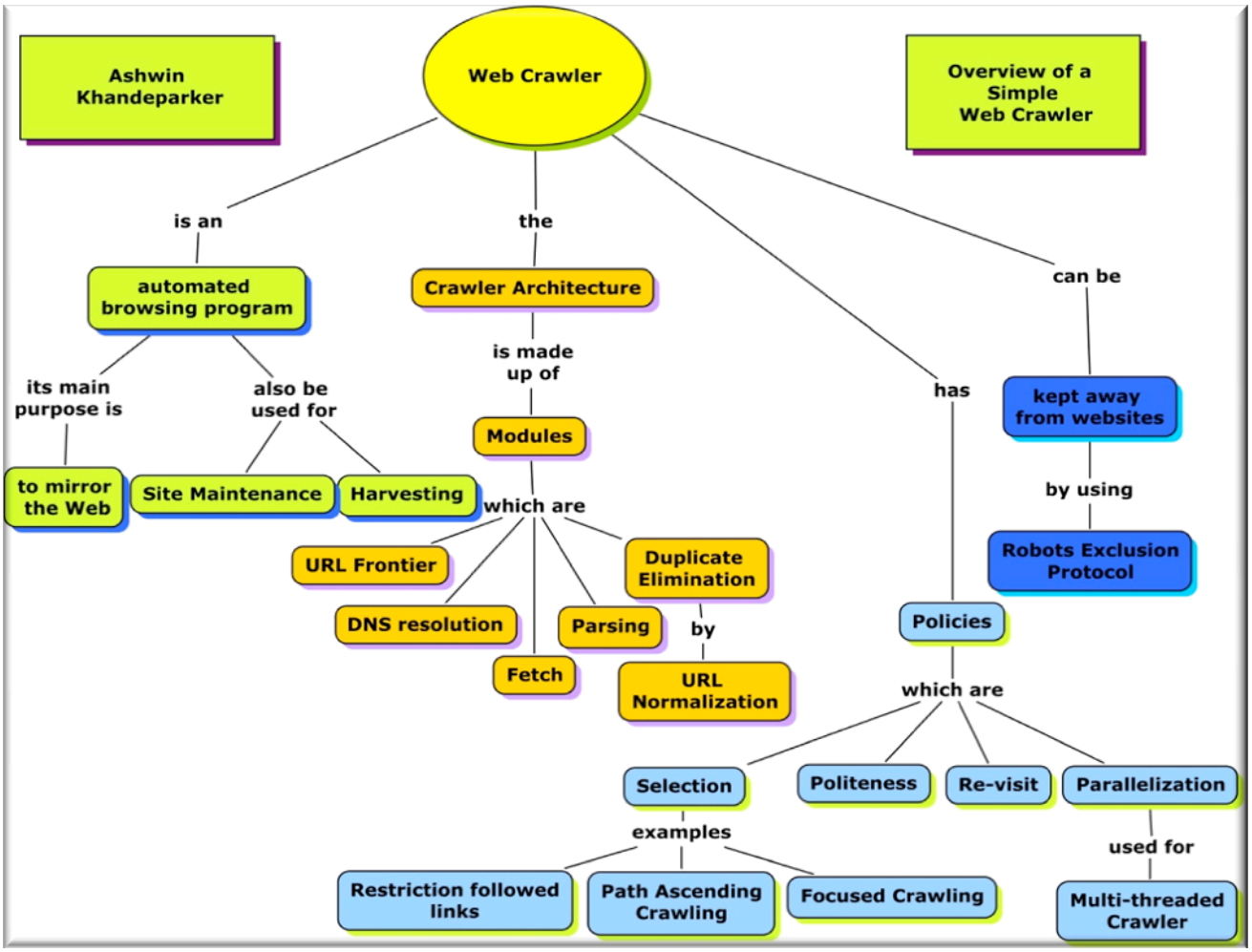
- Similarity to a query: Comparing the user query and the resulting pages that are fetched by the crawler and looking at similarity (whether the user needs are met) gives us an idea of the importance of the web page.

- Classifier Score: A classifier (bot) may be trained by a human to identify the pages that are relevant to the needs of the users. This classifier may then provide automated scores to each crawler page based on its estimated relevance. The training is done by specifying a small sample of relevant pages as positive examples.

- Link-based popularity: using algorithms such as Page Rank or HITS, which provide popularity estimates of each of the crawled pages. A simpler method would be to use just the number of in-links to the crawled page to derive similar information. Many variations of link based methods using topical weights are choices for measuring topical popularity of pages.

- Retrieval System Rank: N different crawlers are started from the same seed and allowed to run till each crawler gathers P pages. The entire set of N*P pages collected from the crawler are ranked against the initiating query or description using a retrieval system such as SMART. The rank provided by the retrieval system for a page is used as its relevance score.

10. **CMap:**



11. **Apache Tika :** The Tika application jar (tika-app-x.y.jar) can be used as a command line utility for extracting text content and metadata from all sorts of files. This runnable jar contains all the dependencies it needs, so you don't need to worry about classpath settings to run it.

The usage instructions are shown below.

usage: java -jar tika-app-x.y.jar [option] [file/Document/URL]

Options:

-? or --help Print this usage message

-v or --verbose Print debug level messages

-g or --gui Start the Apache Tika GUI

-x or --xml Output XHTML content (default)

-h or --html Output HTML content

-t or --text Output plain text content

-m or --metadata Output only metadata

Description:

Apache Tika will parse the file(s) specified on the command line and output the extracted content or metadata to standard output. Instead of a file name you can also specify the URL of a document to be parsed. If no file name or URL is specified (or the special name "-" is used), then the standard input stream is parsed.

12. **Evaluation of learning objective achievement:**

In their presentation, the students need to analyze and give a brief presentation about their findings based on the content taught in this module. The students should show good understanding of the Web Crawler basics, Crawler architecture, and Crawler policies.

13. Exercises / Learning activities:

a. Seed URLs are used as inputs to Web crawlers. Kindly list ten such URLs related to the topic: Storm Sandy.

b. For this exercise Apache Tika should be used. (Please submit the command line queries for all the exercises in this section.) We have covered the command-line utilities in section 11 of the tutorial above. Extract the following contents from the seed URLs collected in 13.a (use separate seed URLs for each) using command line queries:

i. Plain Text

ii. HTML

iii. metadata

iv. XML

Note: for this exercise, download the latest version of Apache Tika **(**http://www.us.apache.org/dist/tika/tika-app-1.2.jar**)**

c. The following questions are related to Robots.txt**:**

i. Write a file to only allow the crawler "Seeker" to visit the URLs placed in the file hierarchy "/storm/sandy".

ii. Write a file to disallow all crawlers from accessing the following directories:

1. /~storm/sandy/

2. /~new jersey/storm-sandy.html

d. The following questions are related to the URL frontier**:**

i. Why is it better to partition hosts (rather than individual URLs) between the nodes of a distributed crawl system?

ii. Why should the host splitter precede the Duplicate URL Eliminator?

e. Web search engines A and B each crawl a random subset of the Web. Some of the pages crawled are duplicates – exact textual copies of each other for different URLs. Assume that duplicates are distributed uniformly amongst the pages crawled by search engines A and B. Further, assume that a duplicate is a page that has exactly two copies – no pages have more than two copies. Search engine A indexes pages without duplicate elimination whereas search engine B indexes only one copy of each duplicate page. The two random subsets have the same size before duplicate elimination. If 45% of search engine A's indexed URLs are present in search engine B's index, while 50% of search engine B's indexed URLs are present in search engine A's index, what fraction of the Web consists of pages that do not have a duplicate?

14. **Resources**

Readings for students:

- Pant, Gautam; Srinivasan, Padmini; Menczer, Filippo (2004), "Crawling the Web" http://dollar.biz.uiowa.edu/~pant/Papers/crawling.pdf (12/8/2009)
- Christopher D Manning, Prabhakar Raghavan, Hinrich Schutze "*An Introduction to Information Retrieval*" (pages 443-469) http://nlp.stanford.edu/IR-book/information-retrieval-book.html (12/8/2009)

Reading for Instructor:

- Pant, Gautam; Srinivasan, Padmini; Menczer, Filippo (2004), "Crawling the Web"
- http://dollar.biz.uiowa.edu/~pant/Papers/crawling.pdf (12/8/2009)
- Christopher D Manning, Prabhakar Raghavan, Hinrich Schutze "*An Introduction to Information Retrieval*" (pages 443-469)
- http://nlp.stanford.edu/IR-book/information-retrieval-book.html (12/8/2009)
- Cho, J.; Garcia-Molina, H.; Page, L. (1998-04), "Efficient Crawling Through URL Ordering", Seventh International World-Wide Web Conference, Brisbane, AustraliaBorgman, C. L. (2003).
- http://ilpubs.stanford.edu:8090/347/ (12/8/2009)
- Heydon, Allan, Najork, Marc (1999-06-26) Mercator: A Scalable, Extensible Web Crawler
- http://www.cindoc.csic.es/cybermetrics/pdf/68.pdf (12/8/2009)
- Cristian Duda,Chong Zhou , Gianni Frey, Donald Kossmann. "AJAXSearch: Crawling, Indexing and Searching Web 2.0 Applications "
- http://www.dbis.ethz.ch/research/publications/AjaxSearchVLDB08.pdf (12/8/2009)

IR-related resources from Wikipedia and other sources:

- Web Crawler: http://en.wikipedia.org/wiki/Web_crawler (12/8/2009)
- Web Crawler: http://www.devbistro.com/articles/Misc/Implementing-Effective-Web-Crawler (12/8/2009)
- **Focused Crawler:** http://en.wikipedia.org/wiki/Focused_crawler **(12/8/2009)**

15. **Glossary:**

- URL: Uniform Resource Locator
- WWW: World Wide Web
- HTTP: Hypertext Transfer Protocol

- *Crawler: A Web crawler is a computer program that browses the World Wide Web in a methodical, automated manner.

- *Web Page: A webpage or web page is a document or resource of information that is suitable for the World Wide Web and can be accessed through a web browser.

- SMART: Self-Monitoring, Analysis, and Reporting Technology

*From http://www.wikipedia.org/

16. **Additional useful links:**

- Some commercial Web Crawlers:
  - Winweb Crawler: http://www.winwebcrawler.com/index.htm (12/8/2009)
  - Spinn3r: http://spinn3r.com/ (12/8/2009)
  - Advanced Site Crawler: http://www.innovative-sol.com/sitecrawler/index.htm (12/8/2009)
  - Website Ripper Copier: http://www.tensons.com (12/8/2009)

17. **Contributors**

- Initial authors:
  - Dr Edward A. Fox
  - Ashwin S Khandeparker
- Updated by:
  - Mohamed M.G. Farag
  - Mohammed Saquib Akmal Khan
  - Prasad Krishnamurthi Ganesh
  - Gaurav Mishra
- Reviewers:
  - Seungwon Yang
  - John Ewers
  - Tarek Kanan
  - Ashwin Palani
  - Kiran Chitturi
  - Dr Edward A. Fox