Review of Paper 3 :- CRAWLING THE WEB

Ramkumar Rajabaskaran(85241493)

"Crawlers" are programs or bots that use the graph structure of the web to start from a seed page(s) and fetch the hyperlinks to other web pages present in that page and use it to move from page to page till they traversed a specific range of pages or there is no more page to visit. This can be done within a few minutes, to fetch up to 10000 pages. But a crawler must be aware of spider-traps and are limited by the internet speed, parsing of text and canonicalizing URL's. The Crawler maintains a list of unvisited URL's to visit called the **"Frontier".** Each iteration, the crawler picks up its URL's from this frontier and fetches new URL's which are fed back into the Frontier. There are different kinds of Frontier's that are depended on whether speed/memory is a parameter. Usually the frontier is a FIFO Queue or Priority Queue where the URL's are arranged based on their score. There is also a Hash-table associated with the URL's that have been visited to identify whether the URL has been visited or not. Once the Priority Queue is full, only the **best** MAX URL's are kept in the Frontier (Best-First Crawler). A history of the web-pages is also kept mostly in the disk to do a post crawl analysis and evaluations. Usually a Hash-Function is used to store the pages.

In order to fetch the pages, we must issue a **"HTTP request"** through a HTTP client for a page read its response. Then we must parse the response to obtain the useful information. There is also a condition that a crawler must identify the restrictions that are placed on a web-page by reading the **"Robots.txt".** This file provides the access policy for different users that may try to crawl the site and we must not crawl those URL's mentioned as disallowed. It is efficient to cache the access policies of recently visited server's. After fetching the pages, we must parse the file by canonizing it, remove stop-words and stem the words to help prevent duplicates. Canonization is to change the relative URL's into absolute URL's using the base URL based on some heuristics. A way to avoid traps is to limit the URL sizes to 128 or 256 characters. The system recognizes 9 words as Stop words {an, and, by, for, from, of, the, to, with}. Stemming is to change words into its root form like observed, observes, observing, observation into observe. This may lead to precision loss. Crawler Infrastructure may sometimes also consider Multi-threaded crawlers which improve efficiency but need lots of synchronization and Frontier locking policies to avoid errors. They also need to deal with empty frontier as it may seem to be empty but threads executing may have fetched URL's afterwards, hence the thread is put to sleep for some time.The fetching of URL's is dependent on the crawling algorithm used. The various algorithms discussed in this paper are Naive Best-First Crawler, SharkSearch, Focused and Context Focused Crawler and InfoSpiders.

In the **Naïve Best-First Crawler**, the Web-pages are stored as a vector of words, weighted by the frequency of its occurrence. The crawler then computes the cosine similarity of the page to the query and scores the unvisited URL's based on this similarity value. In a Multi-thread implementation, this acts as a best-N-First crawler and shows great superiority on the retrieval of relevant pages for N = 256. It keeps an upper bound on the frontier size and fetches the best N URL's. **SharkSearch** is like the best-first search algorithm but with a more refined notion of potential scores where the anchor text and the text surrounding the links influence the score of links. It assigns a score based on the neighboring links and on the inherited score. The neighboring score is based on the context surrounding the link and the anchor text. **Focused Crawlers** classify pages based on a topic taxonomy. The user provides URL's of interest and they get classified based on their taxonomy. The user then corrects the automatic classification through an interactive process. It then uses the example URL's to build a Bayesian classifier that find the probability that a page p belongs in a category c. A relevance score is associated based on this probability. The **Context Focused Crawlers** also use the Bayesian classifier but they provide the link distance from a crawled page to relevant pages. If we are searching for graph theory, it would score mathematical and statistic university pages more than some other page like Law School. A normal crawler will put a math's home page as low priority having not found "graph theory" in it, but a context focused crawler will give it higher priority. The Final Algorithm discussed was the **InfoSpiders.** They use an adaptive population of agents to search for relevant pages based on a adaptive query list and neural net to decide which links to follow. The algorithm provides an exclusive frontier to each agent and limits it to the links in that page alone due to limited memory. By some improvements, this algorithm is found to outperform the native best-first crawler. A back-propagation algorithm is used for learning.

The Crawlers are to be evaluated on its ability to retrieve relevant pages. It does so based on some methods that are used to measure page importance. They are Keywords in a document (frequency with which words appear) , Similarity in a query , Similarity to seed pages, Classifier Score which is a classifier trained to identify pages based on relevancy to the information need or task, Retrieval System Rank and Link-based Popularity. We can also measure the performance of a crawler with measures of precision and recall.

There are a few applications of Crawlers discussed in the paper. MySpiders is a java applet that implemented the InfoSpiders and best-first algorithms. The results are displayed dynamically. It is a multi-thread implementation. The system uses Google Web API to get few seed pages and starts from them. Another application was CORA which built topic-specific portals such as for collecting and maintaining research papers in Computer Science.

Thus this paper discusses about the methods and algorithms that are used when crawling the web from building a crawling infrastructure, various crawling. The latter half of the paper talks about how to evaluate a crawler and finishes with various application of the crawlers such as MySpiders and CORA. Crawlers are becoming more important tools to support web-portals and search engines. The various steps to crawl a web were discussed. Initially we need to parse the response received and extract valid data. Since the web-content is dynamic, the need for crawling algorithms that reduce the memory used and increase speed is required as mentioned in the algorithms above. The Crawlers also need to be evaluated for their relevancy thus bringing scope for improvement. Hence, they can to become more efficient, precise and robust.