Progress in Nonlinear Dynamics and Chaos
Vol. 1, 2013, 1-7
ISSN: 2321 – 9238 *(online)*
Published on 28 April 2013
www.researchmathsci.org

Web Crawling As Nonlinear Dynamics

Chaitanya Raveendra
Professor of Computer Applications
Nehru Institute of Information Technology and Management
Coimbatore 641105 & Research scholar
Karpagam University
E.mail: chaitanya2575@gmail.com

Received 10 April 2013; accepted 17April 2013

Abstract. Web search engines work by storing information about many web pages, which they retrieve from the WWW itself. These pages are retrieved by a Web crawler, an automated Web browser which follows every link it sees. The contents of each page are then analyzed to determine how it should be indexed. In this paper we show that the web crawling is a nonlinear problem and the various methods and architectures that are evolved during the research of enhancing the performance of the search engines.

Keywords: Non linear Dynamics, Search engines, Web crawler, Automatic downloader, Web mining, Information retrieval systems, web page classification

1. Introduction

1.1. World Wide Web

The **World Wide Web** is a system of interlinked hypertext documents accessed via the Internet. With the help of a web browser, one can visit Web pages that contain text, images, videos, other multimedia content and navigate between them using hyperlinks. Most people are thinking that the Internet and the World Wide Web are same but actually they are different. The Internet is communication system that transfers the data globally. Internet provides connectivity between computers by establishing the required infrastructure and using the hardware. In contrast, the Web is one of the services communicated via the Internet. It is a collection of interconnected documents and other resources, linked by hyperlinks and URLs. In short, the Web is an application running on the Internet.

1.2. Search engines

A **Web search engine** is a tool designed to search information on the World Wide Web. The search results are usually presented in a list and are commonly called *hits*. The information required may spread across many web pages, and may consist of images, videos, text or other types of files. Some search engines also mine data available in databases or open directories. Unlike Web directories, which are maintained by human

editors, search engines operate algorithmically or are a mixture of algorithmic and human input [3]**.**

A search engine needs to perform different operations in a specific order as given below.

1. Web crawling
2. Indexing
3. Searching

Web search engines work by storing information about many web pages, which they retrieve from the WWW itself. These pages are retrieved by a Web crawler - an automated Web browser which follows every link it sees. Exclusions can be made by the use of robots.txt. The contents of each page are then analyzed to determine how it should be indexed (for example, words are extracted from the titles, headings, or special fields called meta tags). Data about web pages are stored in an index database for use in later queries. When a user enters a query into a search engine (typically by using key words), the engine examines its index and provides a listing of best-matching web pages according to its criteria, usually with a short summary containing the document's title and sometimes parts of the text.

1.3. Web crawler

A crawler is a program that retrieves web pages which is widely used by web search engines because the information on WWW is distributed and also the information environments become complex. Limited computing resources and limited time leads the research to topic driven crawler (also called focused crawler, retrieve web pages relevant a topic).Topic driven crawler carefully decides which URLs to scan and in what order to pursue based on previously downloaded pages information. Some evaluation methods for choosing URLs [1] and several special crawlers, Naive Best-First crawler and DOM crawler [2] do not have satisfying adaptability. The typical web crawler architecture is shown in fig 1 which is used by Carlos Castillo.

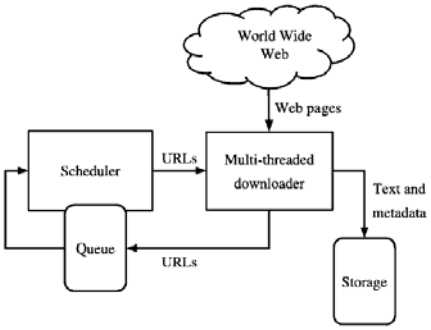

Figure 1. Typical web crawler architecture

2. Strategies for efficient crawling through URL ordering

Identifying the most important pages in short span of time is very useful when a crawler cannot visit the entire Web in a reasonable amount of time. Cho et.al has defined

importance metrics, ordering schemes to achieve the goal. Given a Web page p, we can define the importance of the page, I(p), in one of the following ways**.[1]**

1. **Similarity to a Driving Query Q** - A query Q drives the crawling process, and I(p) is defined to be the textual similarity between p and Q.
2. **Backlink Count**: The value of I(p) is the number of links to p that appear over the entire Web. We use IB(p) to refer to this importance metric.
3. **PageRank**: The IB(p) metric treats all links equally. Thus, a link from the Yahoo home page counts the same as a link from some individual's home page. The PageRank backlink metric, IR(p), recursively defines the importance of a page to be the weighted sum of the importance of the pages that have backlinks top.
4. **Forward Link Count**: a page with many outgoing links is very valuable, since it may be a Web directory.
5. **Location Metric**: The IL(p) importance of page p is a function of its location, not of its contents. If URL u leads to p, then IL(p) is a function of u.

The crawler's goal is that if possible visits high $I(p)$ pages before lower ranked ones, for some definition of $I(p)$. The crawler will only have available $I(p)$ values, so based on these it will have to guess what are the high $I(p)$ pages to fetch next. They used crawl and stop, crawl & stop with threshold, limited buffer crawl models to evaluate their crawler performance.

A crawler keeps a queue of URLs it has seen during a crawl, and must select the next URL to visit from this queue. The ordering metric O is used by the crawler for this selection, i.e., it selects the URL u such that $O(u)$ has the highest value among all URLs in the queue. The O metric can only use information seen (and remembered if space is limited) by the crawler. With a good ordering strategy, crawlers that can obtain a significant portion of the hot pages relatively early could be developed.

3. Types of crawlers

3.1. Focused crawlers

Focused crawlers are programs that wander in the Web, using its graph structure, and gather pages that belong to a specific topic. The most critical task in Focused Crawling is the scoring of the URLs as it designates the path that the crawler will follow, and thus its effectiveness [7]**.**

Ioannis Partalas, George Paliouras, Ioannis Vlahavas, proposed a focused crawler that is based on the RL framework **[5**]. More specifically, RL is employed for selecting an appropriate classifier that will in turn evaluate the links that the crawler must follow. The introduction of link classifiers reduces the size of the search space for the RL method and makes the problem tractable. The selected classifier is the one that scores the URLs of a visited page. The classifier score high if a URL belongs to the relevant class. The mechanism that is used for training the RL module is the Q - learning algorithm **[6].** Q-learning finds an optimal policy based on the action-value function, Q(s, a). The Q - function expresses the benefit of following the action a when in state S. The selection of a classifier in a specific page is associated with the expected relevance of the next page (state) that the crawler will fetch. The features that will be used to represent both the

states and the actions are defined based on the literature of focused crawling; the following features are chosen to represent a state-action pair:

- Relevance score of a page with respect to the specific domain.
- Relevance score of the page, computed by the selected classifier (action).
- Average relevance score of the parents of the page that is crawled.
- Hub score.

Soumen Chakrabarti, Martin van den Berg, Byron Dom proposed a focused crawler with two hypertext mining programs that guide the crawler: a classifier that evaluates the relevance of a hypertext document with respect to the focus topics, and a distiller that identifies hypertext nodes that are great access points to many relevant pages within a few links. Focused crawling acquires relevant pages steadily while standard crawling quickly loses its way, even though they are started from the same root set. Focused crawling is robust against large perturbations in the starting set of URLs [8].

3.2. Multi-threaded crawlers

A sequential crawling loop spends a large amount of time in which either the CPU is idle (during network/disk access) or the network interface is idle (during CPU operations). Multi-threading, where each thread follows a crawling loop, can provide reasonable speed-up and efficient use of available bandwidth. Note that each thread starts by locking the frontier to pick the next URL to crawl. After picking a URL it unlocks the frontier allowing other threads to access it. The frontier is again locked when new URLs are added to it. The locking steps are necessary in order to synchronize the use of the frontier that is now shared among many crawling loops (threads). The model of multi-threaded crawler follows a standard parallel computing model [9].

3.3. Context focused crawler

Context focused crawlers [10] use Bayesian classifiers to guide their crawl. However, unlike the focused crawler described above, these classifiers are trained to estimate the link distance between a crawled page and the relevant pages. The context focused crawler is trained using a *context graph* of L layers corresponding to each seed page. The seed page forms the layer 0 of the graph.

The pages corresponding to the in-links to the seed page are in layer 1. The in links to the layer 1 pages make up the layer 2 pages and so on. We can obtain the in-links to pages of any layer by using a search engine. Once the context graphs for all of the seeds are obtained, the pages from the same layer from each graph are combined into a single layer. This gives a new set of layers of what is called a *merged context graph*.

A set of naive Bayes classifiers are built, one for each layer in the merged context graph. All the pages in a layer are used to compute Pr(*tjcl*), the probability of occurrence of a term t given the class cl corresponding to layer. A prior probability, $\Pr(cl) = 1=L$, is assigned to each class where L is the number of layers. The probability of a given page p belonging to a class cl can then be computed (Pr(*cljp*)). Such probabilities are computed for each class. The class with highest probability is treated as the winning class (layer). However, if the probability for the winning class is still less than a threshold, the crawled page is classified into the \other" class. This \other" class represents pages that do not have a good fit with any of the classes of the context graph. If the probability of the winning class does exceed the threshold, the page is classified into the winning class.

3.4. WIRE project crawlers

This approach is to provide the crawler with access to all the information about the collection to guide the crawling process effectively. This can be taken one step further, as there are tools available for dealing with all the possible interactions between the modules of a search engine, as shown in Fig 2. The indexing module can help the Web crawler by providing information about the ranking of pages, so the crawler can be more selective and try to collect important pages first. The searching process, through log file analysis or other techniques, is a source of optimizations for the index, and can also help the crawler by determining the "active set" of pages which are actually seen by users. Finally, the Web crawler could provide on-demand crawling services for search engines. All of these interactions are possible if we conceive the search engine as a whole from the very beginning [11].

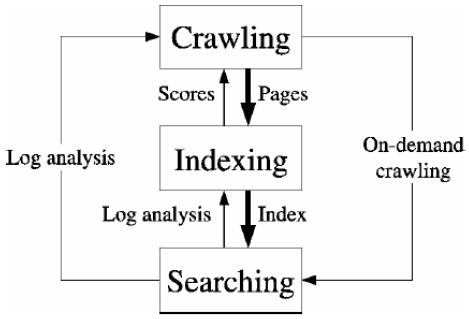

Figure 2. Cyclic architecture of search engine

3.5. Distributed crawlers

Distributed web crawling is a distributed computing technique whereby Internet search engines employ many computers to index the Internet via web crawling. The idea is to spread out the required resources of computation and bandwidth to many computers and networks. As of 2003 most modern commercial search engines use this technique. Google and Yahoo uses thousands of individual computers to crawl the Web.

Newer projects are attempting to use a less structured, more ad-hoc form of collaboration by enlisting volunteers to join the effort using, in many cases, their home or personal computers. LookSmart is the largest search engine to use this technique, which powers its Grub distributed web-crawling project.

This solution uses computers that are connected to the Internet to crawl Internet addresses in the background. Upon downloading crawled web pages, they are compressed and sent back together with a status flag (e.g. changed, new, down, redirected) to the powerful central servers. The servers, which manage a large database, send out new URLs to clients for testing.

4. Nonlinear dynamics

In mathematics, a **nonlinear system** is a system which is not linear, that is, a system which does not satisfy the superposition principle, or whose output is not proportional to its input. Less technically, a nonlinear system is any problem where the variable(s) to be

solved for cannot be written as a linear combination of independent components. A non homogeneous system, which is linear apart from the presence of a function of the independent variables, is nonlinear according to a strict definition, but such systems are usually studied alongside linear systems, because they can be transformed to a linear system of multiple variables [4].

Nonlinear problems are of interest to physicists and mathematicians because most physical systems are inherently nonlinear in nature. Nonlinear equations are difficult to solve and give rise to interesting phenomena such as chaos. The weather is famously nonlinear, where simple changes in one part of the system produce complex effects throughout. The dynamic nature of the WWW, makes the crawling as a non linear dynamic problem.

5. Dynamics in web functionality

Today the dynamics of the WWW documents is achieved by using various scripting languages like CGI programming, Java script, VB script. These languages are used to achieve the validation in the client side programming. The dynamic content of the web page is generated by the web server based on the request. The http request is processed by the server and the responses are generated by servlets, ASP or JSP programs. These programs received the input through Http request object and generated the response by running a ASP/JSP program in server side. The generated dynamic content is encapsulated as a http response and given to the client

6. Web Crawler: View as a nonlinear dynamic problem

We are interested in giving a pattern for a search engine on the web pages. This search must be made for a specific purpose which can be studied to the functionalities and the links between them. Our objective will be minimize the run time and gain more information from the web pages. Thus the problem can be well put as a problem in weighted directed graph. This can also be treated as a routing problem under randomized algorithmic approach. After comparing various page ranking and web crawling algorithm, we finalize that the problem can be studied under non linear dynamics. .

REFERENCES

1. Cho, H. Garcia-Molina and L. Page, *Efficient crawling through URL ordering*, In: Proceedings of the 7th World Wide Web Conference (1998).
2. G. Pant and F. Menczer, *Topical crawling for business intelligence*, In: Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL), (2003).
3. http://en.wikipedia.org/wiki/Web_search_engine
4. http://en.wikipedia.org/wiki/Nonlinear_system
5. R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, (1999).
6. C.J.Watkins and P. Dayan, '*Q-learning*, *Machine Learning*, 8 (1992), 279–292.
7. Ioannis Partalas, George Paliouras, Ioannis Vlahavas, Reinforcement Learning with Classifier Selection for Focused Crawling, (2002).

8. Soumen Chakrabarti , Martin van den Berg, Byron Dom, *Focused crawling: a new approach to topic-specific Web resource discovery*, Computer Network 31(11-16), 1999.
9. V. Kumar, A. Grama, A. Gupta and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms*, Benjamin/Cummings, (1994).
10. M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori, Focused crawling using context graphs, In Proc. *26th International Conference on Very Large Databases* (VLDB 2000), Cairo, Egypt, (2000), 527-534.
11. Carlos Castillo, *Effective web crawler*, Ph.D. Thesis, University of Chile, November 2004