ISSN:2320-0790

An International Journal of Advanced Computer Technology

Improving the Performance of Crawler Using Body Text Normalization

Farha Qureshi[1], Amer Ahmed Khan[2]

[1]PG Scholar, [2]Asst. Prof.

Computer Science Dept,

V.I.T.S Kareemnagar[1],

JNTU University[2]

farha.r.qureshi@gmail.com[1], *amer.ahmed.khan@gmail.com* [2]

Abstract— **Search engine is comprised of components like crawler, repository, indexing, querying and ranking. Work of crawler is to crawl the web and download pages. These pages are then stored in repository. The crawler mechanism should be smart enough to identify the pages that it had or had not crawled before. Here we propose a suitable mechanism that will avoid downloading of duplicate page contents and also avoid unnecessary URL extraction time. So as to meet the desired mechanism we introduce MD5 digest of body text of every page.**

Keywords: Crawler with url normalization, Crawler with whole page content MD5, Crawler with Body text normalization.

I. Introduction

Today we are standing in the era of Internet. Where the size of internet data is multiplying like a big bang. Wherever we see whatever we do is all data. Right from the textual data to the multimedia data. We are all surrounded with huge amount of data. Data is generated from desktops, laptops, PDA, handheld devices like mobiles. The data generated everyday is in great amount as compared to the one that were generated in earlier days. As the population is going on increasing and the use of internet too. Earlier days, when there was no search engines, people used to surf the internet with the links which they knew. But, as the search engine step into, world of internet was changed and is going on changing with the working advancement in the search engine. Working of search engine is not a easy task, which itself is combination of various functionalities. As you fire a keyword there are list of links that can help you find what actually you need. All over the internet there are millions of books, journals, blogs, research patents and n number of resources that consist of information that you may be expecting. So, it is necessary to collect all the relevant data from all round the available resources and index them in such a manner that they prove useful to you. Search Engine is comprised of Crawling, Storing, indexing, querying, ranking. The work of Search engine starts by gathering pages from all over the World Wide Web.

II. Working of Search Engine

Search Engine is like a Dictionary that shows you a list of documents that matches your search keyword. It not only shows you list of results that matches your keyword but also presents the result in a manner that is more relevant to your search token.

A. Types of search engine

- Powered by robots (called crawlers, ants or spiders).
- Powered by human submissions.

1) Crawler Based Search Engine: These are agents that visit individual sites, go on visiting each link in every document and download all the pages in central repository. If the pages are static then the robots never visit the pages again after storing them in repository. If the pages are dynamic then the robot even after storing the page in the repository it go on visiting it so as to maintain fresh information. Then these documents are indexed so as to attach these documents according to their search keyword.

2) Human-powered Search Engine: Such search engines totally rely on humans to submit information that is subsequently indexed and catalogued. The information submitted by the humans are then indexed based on their category. This type of search engines are rarely used at large scale. But these are useful in the organizations where small scale of data is dealt with. This kind of search engine is a failure, as its impossible for a human to collect the pages manually, as the web is growing exponentially.

B. Components of search engine

1) Web Crawler: Search engines use the automated program called web crawler which visits the every single URL of the seed URL given to it to start crawling and downloads the web pages in order to create an index of data in a local repository. Where search engines return back the information from to the desired user.

- A web-crawler is a program/software or automated script which browses the World Wide Web in a methodical, automated manner.

Components of Web Crawler:

-Seed Page:-The set of starting URL is known as “Seed Page”.

-Frontier (Processing Queue): The list of un-visited links or URLs is known as, “Frontier”.

-Parser: extract information that will feed and possibly guide the future path of the crawler.

Algorithm of a Basic Web Crawler:

Step1: Select a starting seed URL or URLs

Step2: Add it to the frontier

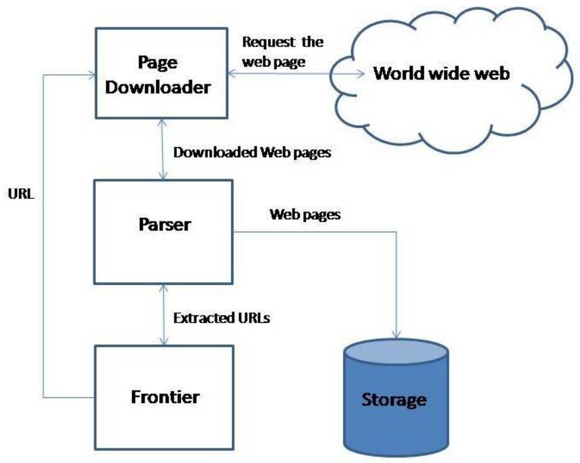
Step3: Now, pick the URL from the frontier

Step4: Fetch the web-page corresponding to that URL

Step5: Parse that web-page to find new URL links

Step6: Add all the newly found URLs into the frontier

Step7: Go to step 3 and repeat while the frontier is not empty.



2) Maintaining Database/Repository: The work of crawler is to download the pages by following the links. The seed url is given as an input to the crawler and the crawler go on following the link and download the pages. These pages are stored in the repository. This storage need to be maintained. The page content should be maintained in the correct form as the pages are dynamic.

3) Indexing: Once the pages are stored in the repository, the next job of search engine is to make a index of stored data. The indexer module extracts all the words from each page, and records the URL where each word occurred. The result is a generally very large “lookup table" that can provide all the URLs that point to pages where a given word occurs. The table is of course limited to the pages that were covered in the crawling process.

4) Querying: After creation of index, now the pages are available with their keywords that they match. The user types the query and based on the query, results are returned to the user. This module is responsible for returning the results to the user based on the query fired.

5) Ranking: There are list of matching results to the given keyword. So, these results should be framed according to the relevancy. The document that talks more about the keyword will be placed at the top and the document that talks less will be placed thereafter. This is called the ranking. There are lots of parameters based on which the pages are ranked.

This was all about the basic working of the Search Engine. Let’s have a look on various Strategies of crawling.

III. WEB CRAWLING STRATEGIES

Two major approaches used for crawling are:

I) Blind Traversing approach

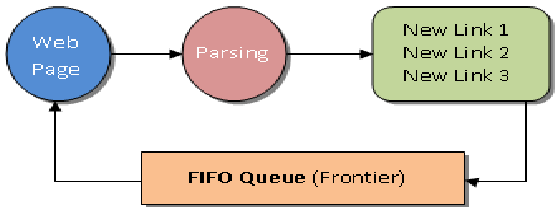
II) Best – First Heuristic approach

A. *Blind Traversing Approach*

In this approach, we simply start with a seed URL and apply the crawling process as stated earlier. It is called blind because for selecting next URL from frontier, no criteria are applied. Crawling links are selected in the order in which they are encountered in the frontier (in serial order) One algorithm widely common to implement Blind traversing approach is – Breadth First Algorithm. It uses FIFO QUEUE data structure to implement the frontier; it is very simple and basic crawling algorithm. Since this approach traverses the graphical structure of WWW breadth – wise, Queue data structure is used to implement the Frontier.

Algorithm that comes under Blind Crawling approach is- Breadth First Algorithm.

Breadth First Algorithm: A Breadth-First crawler is the simplest strategy for crawling. This algorithm was explored in 1994 in the WebCrawler as well as in more recent research. It uses the frontier as a FIFO queue, crawling links in the order in which they are encountered [5]. The problem with this algorithm is that when the frontier is full, the crawler can add only one link from a crawled page. Breadth-First Algorithm is usually used as a baseline crawler; since it does not use any knowledge about the topic, it acts blindly. That is why, also called, Blind Search Algorithm. Its performance is used to provide a lower bound for any of the more sophisticated algorithms.



Breadth-First crawling Approach

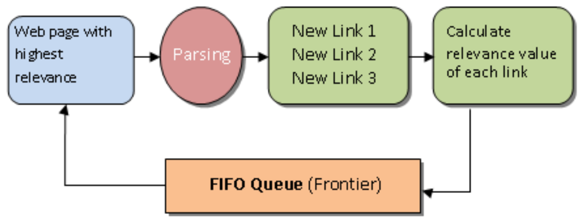
Drawbacks of Breadth First Approach: In real WWW structure, there are millions of pages linked to each other. The size of the repository of any search engine cannot accommodate all pages. So it is desired that we always store the most suitable and relevant pages in our repository. Problem with Blind Breadth First algorithm is that it traverses URLs in sequential order as these were inserted into the Frontier. It may be good when the total number of pages is small. But in real life, a lot of useless pages can produce links to other useless pages. Thus storing and processing such links in frontier is wastage of time and memory. So we should select a useful page from the frontier every time for processing irrespective of its position in the frontier. But Breadth first approach always fetched 1st link from the frontier, irrespective of its usefulness. So the Breadth First approach is not desirable.

B. *Best First Heuristic Approach*

To overcome the problems of blind traverse approach, a heuristic approach called Best- First crawling approach have been studied by Cho et al. [1998] and Hersovici et al. [1998]. In this approach, from a given Frontier of links, next link for crawling is selected on the basis of some estimation or score or priority[2]. Thus every time the best available link is opened and traversed. The estimation value for each link can be calculated by different pre-defined mathematical formulas. (Based purely on the needs of specific engine)

Following Web Crawling Algorithms use Heuristic Approach:

Naive Best - First Algorithm: One Best First Approach uses a relevancy Function rel ( ) to compute the lexical similarity between the desired key-words and each page & associate that value with corresponding links in the frontier. After each iteration, the link with the highest rel ( ) function value is picked from the frontier. That is, the best available link is traversed every time which is not possible in Breadth First Approach. Since any link with highest relevancy value can be picked from the Frontier, most of the Best first algorithms use – ***Priority Queue*** as data structure. The working of Heuristic Crawling Algorithms is illustrated


Web page with highest relevance
Parsing
New Link 1
New Link 2
New Link 3
Calculate relevance value of each link
FIFO Queue (Frontier)

Best-First Crawling Approach

As clear from figure, web-page with highest relevance is picked from any position from Frontier for processing.

Page Rank Algorithm: Page Rank was proposed by Brin and Page [1] as a possible model of user surfing behavior. The Page Rank of a page represents the probability that a random surfer (one who follows links randomly from page to page) will be on that page at any given time. A page's score depends recursively upon the scores of the pages that point to it. Source pages distribute their Page Rank across all of their out links.

Formally:

$$PR(A) = (1-d) + d\,(PR(T1)/C(T1) + \ldots + PR(Tn)/C(Tn))$$

Where PR (A) is the Page Rank of a page A

PR (T1) is the Page Rank of a page T1

C (T1) is the number of outgoing links from the page T1

d is a damping factor in the range $0 < d < 1$, usually set to 0.85

The Page Rank of a web page is therefore calculated as a sum of the Page Ranks of all pages linking to it (its incoming links), divided by the number of links on each of those pages (its outgoing links). As originally proposed Page Rank was intended to be used in combination with content based criteria to rank retrieved sets of documents [Brin and Page1998].This is in fact how Page Rank is used in the Google search engine. More recently Page Rank has been used to guide crawlers [Cho et al. 1998] and to assess page quality.

IV. DIFFICULTIES IN WEB CRAWLING.

1) Crawler should avoid duplicate pages.
2) Crawler should periodically revisit the urls so as to maintain correct data in case of dynamic web pages.

Restricting the downloading of pages that are already crawled requires minute investigation of urls. If the url is already crawled then need not require to download the page again. There are many web sites those copy the pages on their web site, which results in duplicate pages. In such case the url is somewhat different but the page contents are same. Sometimes its easy to guess based on the url structure whether it consist of fresh data or duplicate one. This can avoid downloading such page and parsing the links of those pages. But this is not possible all the time. There were different url normalization schemes that allows to normalize the urls based on the predefined rules. But this never works out in all the situations. Let's consider the case of mirror sites. Mirror sites consist of exact copy of the website. As the number of users those visit a site grows, it causes more network traffic on that site. Mirror sites are created so as to reduce the load on a single server and to avail the same information rapidly to the users far from the original site. For example site situated in New York is far away for the British users, so to make faster access to the site it is good to create a mirror of the site in England, this reduces the access time and the network traffic on the server situated in New York. In this scenario same contents of one web site is available on many different locations called as mirror sites. For example wikileaks is currently mirrored on 1885 sites. From which we choose two syntactically different URLs but leads to the same contents.

http://wikileaks.as50620.net/ and http://wikileaks.dena-design.de/

In above scenario only URL normalization is not enough as syntactically different url can lead to same page contents. So, we need a technique that not only take into account urls but also something additional that will allow the crawler to identify whether the contents of two different urls are equal or not. For this reason a technique was invented called url signature.

V. MESSAGE DIGESTS 5 CRYPTOGRAPHIC HASH FUNCTION.

MD5 message digest is an algorithm widely used to generate 128bit hash value and commonly used to check data integrity [9]. So as to check duplicate data crawled by our crawler we here are generating the MD5 digest for every page downloaded by a crawler to avoid revisiting of the same links and downloading same data again and again.

Crawler takes into account the seed url, there from it starts downloading the pages by following the link. From the downloaded page it extracts the new url and adds it to queue and again downloads the new page. This working continues till there are links in the queue or for the number of links specified by the controller. Many websites simply copies the contents and there is no new data in the site. Such duplicate pages shouldn't be downloaded and followed. To avoid this situation, url normalization can be done, by which we can identify that whether the url have been parsed earlier or not [6] . The syntax based, scheme based and protocol based normalization are types of url normalization [8] .These doesn't prove much useful. Therefore the downloaded page body text normalization is the best available option. As we know that there are html tags, text, images in a page from which text plays an important role. If anyhow we are able to identify that the text of given document is same as that of the pages already downloaded then we will be in a situation to avoid following such page urls. It is not possible to match each and every word of the given document with every page available in the repository. There is a simple solution for this i.e. using an encryption algorithm that can help to match the document contents. One of the best suitable encryption algorithms is message digest. The message digest generate an unique code for the document and is able to identify even a small alphabet change. So, generate MD5 for the body text and check it with the available MD5 available in the storage which was created for every downloaded page. If the newly downloaded page MD5 matches with the any of the MD5 available in the repository that means we have found a page whose contents are already downloaded.

VI. WHY BODY TEXT NORMALIZATION IS NEEDED

Existing system concentrated on url normalization and never opted for body text normalization. As only concentrating on url normalization is not fruitful our system have implemented body text normalization. The more size of body text the more time needed to calculate the md5.

1] If the time taken is more, then the crawling process will slow down and has to suffer.

2] Unable to handle large files efficiently.

3] Takes long time to generate md5 hash.

4] Does not consider any relevance criteria while crawling.

VII. HOW TO NORMALIZE BODY TEXT

Leave the tags and just concentrate on rest of the contents of the page. This is because the duplicity of page content need not necessary in duplicity of the url path. May it be relative path? The site which copied the contents will not necessarily keep the url name same as that of the original document.

1) Finding out the data that can be neglected from the body text.

2) Only removing those portions of data from the body text, this won't affect the contents. Because the reason for generating the md5 is to avoid storing of those pages those lead to same page content.

3) Removing the vowels because it would prove easy to find out the vowels since they are only a,e,i,o,u. So, consonants are not taken into consideration as they are more compared to vowels. The time taken to create md5 of big file should not be less than the time taken to remove the vowels or some other data.

4) Removing the spaces and special symbols which normally don't play an important role.

5) Numbers are not removed because there are sites those keep track of statistics. So, avoid removing numbers.

Summary: Suppose we have a page Pi with size Si.

After normalizing using normText(Pi) we will get the page Pi` with size Si`.

Where Si`< Si.

So, as per our assumption it would take less time to generate MD5 hash.

CONCLUSION

As, per our assumption we have came to the conclusion that our crawler avoids downloading the pages containing duplicate data as compared to the other crawler that takes into account url normalization which fails to avoid downloading of duplicate pages. Body text normalization not only avoids duplicate pages but also reduces the time for creating MD5 of a given page.

REFERENCES

[1] Brin, Sergey, and Page, Lawrence, "The Anatomy of a large scale hyper textual web Search Engine", In Proceedings of the Seventh World-Wide Web Conference, 1998.

[2] Hersovici, M., Jacovi, M., Maarek, Y., Pelleg, D., Shtalheim, M. and Ur Sigalit, "The Shark-Search Algorithm – an application: tailored web site mapping", Computer Networks and ISDN systems, Special Issue on 7th WWW conference, Brisbane, Australia, 30(1-7), 1998.

[3] P. De Bra, G.-J. Houben, Y. Kornatzky, and R. Post, "Information retrieval in distributed hypertexts", Proceedings of RIAO'94, Intelligent Multimedia, Information Retrieval Systems and Management, New York, NY, 1994.

[4] Chakrabarti, S., Berg, M.V.D., and Dom, B., "Focused crawling: a new approach to topic-specific Web resource discovery", In Eighth International World Wide Web Conference, pp.545–562, May 1999.

[5] Mark Najork, Janet L. Weiner, "Breadth First search Crawling Yeilds high quality pages"WWW10 Proceedings in May 2-5 2001, Honk Kong.

[6] Berners-Lee, T., Fielding, R., Masinter, L., "Uniform Resource Identifier (URI): General Syntax", available at http://gbiv.com/protocols/uri/rfc/rfc3986.html

[7] David Hawkings Web Search Engines: Part 1 David Hawking is a principal research scientist at CSIRO ICT Centre, Canberra, Australia, and Chief Scientist at funnelback.com.

[8] Web Crawler with URL Signature – A Performance Study Lay-Ki Soon, Yee-Ern Ku ,Sang Ho Lee. 2012 4th Conference on Data Mining and Optimization (DMO) 02-04 September 2012, Langkawi, Malaysia 978-1-4673-2718-3/12/$31.00 ©2012 IEEE

[9] The MD5 Message-Digest Algorithm, available at: http://tools.ietf.org/html/rfc132130