Topical Web Crawling Using Weighted Anchor Text and Web Page Change Detection Techniques

DIVAKAR YADAV
JIIT University Noida
INDIA
dsy99@rediffmail.com

AK SHARMA
YMCA Faridabad
INDIA
ashokkale2@rediffmail.com

JP GUPTA
JIIT University Noida
INDIA
jp.gupta@jiit.ac.in

Abstract: In this paper, we discuss about the focused web crawler and relevance of anchor text as well as method for web page change detection for search engine. We have proposed a technique called weighted anchor text which uses the link structure to form the weighted directed graph of anchor texts. These weights are further used for deciding the relevance of the web pages as the indexing of these pages is done in the decreasing order of weights assigned to them. Weights are assigned for every incoming link for a node of the directed graph. We applied our algorithm on various websites and observed the results. We deduce that the algorithm can be very useful when incorporated with other existing algorithms. As Web usage has increased exponentially in the past few years. This collection of enormous web pages is highly changing and web pages show a rapid change, the degree of which varies from site to site. We discuss the relevance of change detection and then move on to explore the related work in the area. Based on this understanding we propose a new algorithm to map changes in a web page. After verifying results on various web pages we observe the relative merits of the proposed algorithm.

General Terms:
Algorithm, Design and analysis, Experimentation, Change, Collection, Performance, Updation.

Key Words:
Anchor Text, Directed Graph, Topical Focused Crawling, Web Crawler, DOM, Checksum, Change detection.

1. Introduction:

The enormous size of the World Wide Web has almost made it impossible even for the best of the search engines to cover the entire information database of the Web. Studies have shown that no search engine indexes more than 16% of the Web. The crawler of the search engine downloads only a fraction of Web pages hence it is desirable that the downloaded content contains the most relevant pages. A web crawler is a program or automated script which browses the World Wide Web in a methodical, automated manner. Search engines, use web crawler as a means of providing up-to-date data. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches. One of the major concerns in designing this crawler is the ever evolving nature of the web.

A Focused or a Topical web crawler attempts to download web pages relevant to a set of pre defined topics. Link context forms an important part of Web based information retrieval tasks. Topical crawlers follow the hyperlinked structure of the Web using the scent of information to direct themselves toward topically relevant pages. For deriving the appropriate scent, they mine the contents of pages that are already fetched to prioritize the fetching of unvisited pages. Topical crawlers depend primarily on contextual information. This is because topical crawlers need to predict the benefit of downloading unvisited pages based on the information derived from pages that have

been downloaded. One of the most common predictor is the anchor text of the links.

An anchor text is the clickable part of any hyperlink. The words contained in the anchor text can determine the ranking that a page will receive in a search engine. Anchor text usually gives the user relevant descriptive or contextual information about the content of the link's destination. The objective of search engines is to provide highly relevant search results. That is why it is recommended to make anchor text of a link as relevant as possible to the page and its contents.

Similarly in a web collection, a surfer has access to information on virtually every topic. In this collection pages are being added, removed, modified and rearranged. These modifications may also take place within an extremely short span of time. For example popular news websites like Times of India or online store sites like baazi.com update their content every hour or even in time intervals shorter than that. In order to make optimum use of this changing collection, varied systems have been conceptualized with the goal of bringing to the web users a simplified view of the web and efficiently deal with the changes on web.

Web pages have shown a rapid change, and the rate of change varies from site to site. Therefore managing the local collection fresh as much as possible becomes an issue. If a site that changes in week say, a government website then daily update of that page just leads to waste of precise bandwidth. On the other hand a website such as news website needs to be updated from recursively at short intervals of time.

In an ideal world detecting changes in a page is easy. The simplest implementation would track the “last modified” date returned by the HTTP server. A more accurate mechanism would retain cached copies of Web pages. It is easy to compare a document with a previously cached version and determine if there has been any change. In a similar way, it would be possible to check for changes in specific sections of the document. These cases would return a Boolean “yes/no” indication of change.But in fact we do not live in an ideal world; a difficulty arises when attempting to determine the relevance of changes. The obvious approach would be to assess the magnitude or amount of change. Unfortunately, there is no computation method that provides a direct correlation between syntactic and semantic changes in a web page.

In this paper, we describe a topical crawler which would use the link structure of the web pages along with the anchor text, thereby giving weights to the web pages. These weights would be further used to index the web pages in decreasing order which would be the criteria for deciding upon the relevance of pages. Also we propose an algorithm for automated web page change detection. Our algorithm deals with the problem of weightage assignment to various changes on a web page.

In the next section, we describe the related work. In section 3 we discussed details of the proposed focused web crawler using link structure and weighted anchor text as well as web page change detection techniques. Section 4 describes the experiments performed and their observations. We analyze the results in Section 5. and finally section 6 concludes the paper with some directions for future work.

2. Related Work:

A focused crawler is designed to only collect web pages on a specified topic while traversing the web. The basic idea of a focused crawler is to optimize the priority of the unvisited URLs on the crawler frontier so that pages concerning a particular topic are retrieved earlier. Cho [1] introduces the usage of incremental crawler and proposes several techniques to build an effective incremental crawler. The incremental crawler can improve the “freshness" of the collection significantly and bring in new pages in a more timely manner. They perform various experiments to check how various web pages evolve with time. They conclude with the proposal of architecture of an incremental crawler using page rank.

Jamali [18] explains the topical crawler as a focused crawler aiming at selectively seeking out pages that are relevant to a pre-defined set of topics. The paper further states that besides specifying topics by some keywords, it is customary to use some exemplary documents to compute the similarity of a given web document to the topic. He finally proposes a new hybrid focused crawler, which uses link structure of

documents as well as similarity of pages to the topic to crawl the web pages.

Chakrabarti [3] describes focused crawler as a new hypertext resource discovery system. Rather than collecting and indexing all accessible web documents to be able to answer all possible ad-hoc queries, a focused crawler analyzes its crawl boundary to find the links that are likely to be most relevant for the crawl, and avoids irrelevant regions of the web. They designed two hypertext mining programs – a classifier and a distiller. The classifier evaluates the relevance of a hypertext document with respect to the focus topics whereas the distiller identifies hypertext nodes that are great access points to many relevant pages within a few links.

Pant [17] explains the reliance of topical crawling on link contexts. These crawlers automatically navigate the hyperlinked structure of the web while using link contexts. He further investigates the effects of various definitions of link contexts on the crawling performance by using topical crawlers that are guided by a Support Vector Machine. He proposes from results obtained that a crawler that uses the tag tree hierarchy within web pages provides effective coverage and exploits words both in the immediate vicinity of a hyperlink.

Chakrabarti [2] showed that there is a great deal of usable information on a HREF source page about the relevance of the target page. This information can be exploited by a supervised apprentice which takes online lessons from a traditional focused crawler by observing a carefully designed set of features and events associated with the crawler. They propose alternative focused crawler architecture where documents are modeled as tag trees using DOM (Document Object Model) in which two classifiers are used viz. baseline and apprentice. The baseline classifier refers to the module that navigates through the web to obtain the data for the apprentice classifier. Whereas the apprentice classifier is trained over the data collected and eventually guides the overall crawling.

Further Papapetrou [14] proposes IPMicra, a distributed location aware web crawler that utilizes an IP address hierarchy and allows crawling of links in a near optimal location aware manner.

Diligenti [4] introduced a method to overcome one of the main problem in focused crawling which is performing appropriate credit assignment to different documents along a crawl path. They suggested an algorithm that builds a model for the context within which topically relevant pages occur on the web. The algorithm also possesses the capability of partial reverse crawling.

Iwazume [13] proposes a system called Intelligent Information Collector and Analyzer. The system uses contexts in topical crawlers in a more granular form. They guided a crawler using the anchor text along with ontology. It specifies the common background knowledge shared by the user.

Cho [6] discussed in detail the use and advantage of parallel crawlers. By parallel crawling the work can be distributed thereby resulting into less bandwidth utilization and also increase in page quality.

In a subsequent work, Chung [5] proposes a topic-oriented approach, in which the web is partitioned into general subject areas with a crawler assigned to each. The main motive is to choose the strategy for partitioning the web amongst nodes in such a way that overlapping between the nodes is minimized.

Dalal [15] describes distributed collections of web materials to be common. He explains the Walden's Paths Path Manager tool to support the maintenance of distributed collections. Earlier efforts focused on recognizing the type and degree of change within web pages and identifying pages no longer accessible. The paper extends previous work done with algorithms for evaluating page changes based on context and to locate moved pages and apply the modified approach to suggesting page replacements when the original page cannot be found.

Gao [16] proposes several collaborative crawling strategies for the geographically focused crawling, whose goal is to collect web pages about specified geographic locations, by considering features like URL address of page, content of page, extended anchor text of link, and others. After careful study and experimentation he proposes that strategies like URL address of page and extended anchor text of link are shown

to yield the best overall performance for the geographically focused crawling.

Gravano [8-10] suggested that if a reasonable number of links pertinent to one particular geographic location point to a web resource and these links are smoothly distributed across the location, then this location is treated as one of the geographic scopes of the corresponding web resource. They also propose how to solve aliasing and ambiguity.

Exposto [7] evaluates distributed crawling by means of the geographical partition of the web. The method is based on existence of distributed crawlers for specified geographical locations. Further the evaluation results are compared with hash partitioning strategies.

Markowetz [11] proposed the design and the initial implementation of a geographic search engine prototype for Germany. The described prototype performs massive extraction of geographic features from crawled data, which are then mapped to coordinates and aggregated across link and site structure. The domain used for this is "de". This assigns to each web page a set of relevant locations called footprints, the data of which is then integrated into a high-performance query processor on a cluster-based architecture.

Hersovici [12] proposed SharkSearch which was an advance form of Fishsearch. The shark-search has been embodied into a dynamic web site mapping that enables users to tailor web maps to their interests. It also linearly combined the information gained from the smaller contexts with that from the entire parent pages.

It has been discussed in [23] how to estimate the change frequency of a web page by revisiting the page periodically. A study [19] has been done on how often a crawler should visit a page when it knows how often the page changes.

There has been work done on comparing HTML and XML documents. For example, [24] describes *HtmlDiff*, a tool for detecting and highlighting changes in web pages. More recently, [25] describe *X-Diff*, a change detection algorithm that allows for unordered subtrees. However, the application of differencing in this work differs in that it is not used to detect changes over time, but to extract information that varies among different web pages at the same time

Buttler[27] presented a new mechanism for detecting changes to HTML and XML documents based on the Page Digest encoding, focusing on providing standard change detection mechanisms, operators for more advanced change detection operators, techniques to monitor different aspects of a page, including content, tag types, attributes, and structure, and, finally, a set of unique optimization techniques based on the Page digest encoding that dramatically improves the performance of the system.

Tan [22] proposed a sampling-based algorithm to study a challenging problem in Web crawling: Given a sampled subset of the webpages in a local repository, how can a crawler exploit a fixed amount of available download resources to make the local repository as up-to-date as possible? He investigated in detail the parameter space in algorithm, especially the download granularities and adaptive download probabilities, to achieve the optimal performance in terms of change ratio.

Liu presented WebCQ [28], a prototype of a large-scale web information monitoring system, with an emphasis on general issues in designing and engineering a large-scale information change monitoring system on the Web. Features of WebCQ include the capabilities for monitoring and tracking various types of changes to static and dynamic web page changes, personalized delivery of page change notifications, and personalized summarization of web page changes.

The WebCQ system consists of four main components: a change detection robot that discovers and detects changes, a proxy cache service that reduces communication traffics to the original information servers, a personalized presentation tool that highlights changes detected by WebCQ sentinels, and a change notification service that delivers fresh information to the right users at right time. A salient feature of change detection robot is its ability to support various types of web page sentinels for detecting, presenting, and delivering interesting changes to web pages. This paper describes the WebCQ system with an emphasis on general issues in designing and engineering a large-scale information change monitoring system on the Web.

Some probabilistic models have been exploited to approximate the observed update history of a webpage and predicts its future changes [20], in which the fast-changing pages are more likely to be crawled. The limitation of these models is that they need to gather sufficient and accurate knowledge about each webpage's change history. To address this problem, Cho, et al., [26] have prescribed to randomly sample webpages from each website, and then a crawler should crawl an entire website once it detects that the website has more changed samples than other sites. However, they do not support this choice with any theoretical or empirical evidence.

[21]describes an efficient Web page change detection system based on tree optimizations that were implemented on top of the Hungarian Algorithm, which is employed to compare trees that correspond to HTML Web pages. The optimizations attempt to stop this $O(n^3)$ algorithm before it completes all its iterations based on criteria having to do with properties of HTML and heuristics.

D. Yadav [29] discussed algorithm for extracting web changes consists of three steps: document tree construction, document tree encoding and tree matching for the detection of two types of changes , structural and content changes. The authors have further mentioned that the proposed algorithm has linear time complexity and extracts effectively the changed content from different versions of a web pages. In [30] D Yadav has proposed an architecture for parallel crawler built on the lines of a client server model. Also he has discussed a three-step algorithm for page refreshment. They have used the algorithm to check whether the change is at structure level or content level or at image level in web pages.

3. Proposed Methods:

We have divided this section in two parts. In first part we discuss the relevance of anchor texts in topological web crawler and second part we discuss and present mechanism for detecting changes to HTML documents.

3.1 Relevance of anchor texts for topical web crawlers:

The objective of search engines are to provide highly relevant search results; this is where anchor text helps, as the tendency is to hyperlink words relevant to the landing page. Webmasters may use anchor text to procure high results in search engine results pages. But first off, what is anchor text? Anchor text refers to the text used in linking structure. And the keywords we use contributes to relevancy on the page it links to, will give a bit of a boost to that page when someone searches for that phrase. Now, keep in mind that we need to utilize other optimization techniques in order to rank for that phrase, unless it is a totally non-competitive phrase. But anchor text is definitely one of the factors used in the ranking algorithm to decide which pages rank higher than others. The inclusion of important keywords in the anchor text can make a big difference in the final ranking of web pages. All search engines that matter, give significant weight to the anchor text on web pages.

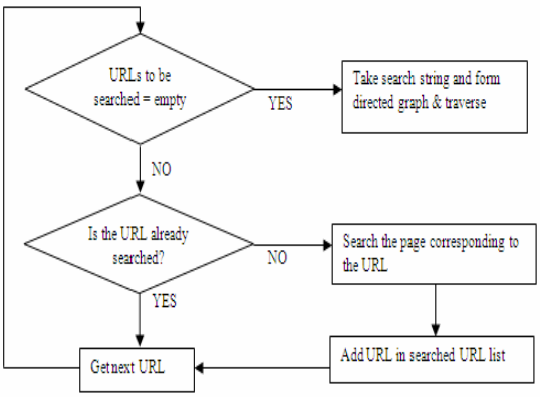


Figure 1 Extraction of Anchor texts and URLs from web pages

Since the relevance of anchor text is increasing highly these days, our technique is based on it. We thus introduce the concept of weighted anchor text wherein the every link has been given a weight. The method basically aims at forming a weighted tree structure where each node has been assigned a constant weight. The tree structure is a weighted directed graph where each incoming link leads to assignment of a constant weight to that particular node. Once the tree structure is formed, the next step is the indexing of the nodes. For arranging the nodes in an order of relevance these weights associated with nodes are of at most importance. We assume that the node

associated with the maximum weight is of most relevance in accordance with the given keyword because it has the maximum number of incoming links in the directed tree structure. Hence the indexing of the nodes in the proposed technique would be entirely based on the decreasing order of the weights associated with the nodes.

3.1.1 Detailed Analysis:

- The link structure of the web is used in this technique. The seed URL is taken as the root point.
- The Anchor Texts present on the URL page are extracted from the seed URL and the URLs corresponding to these anchor texts are also taken.

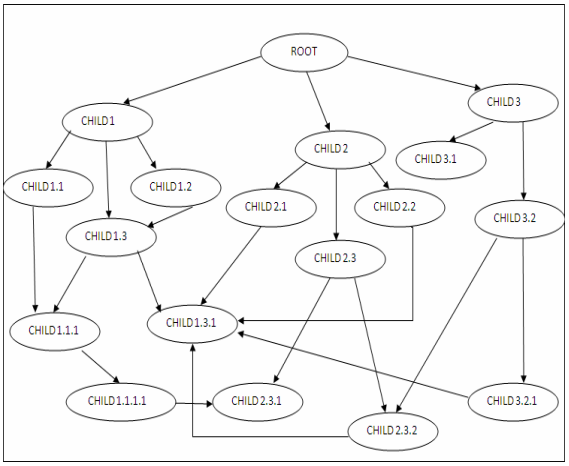


Figure 2. Directed Graph of links from Seed URL

- Now the child URLs of the seed URL are taken as the reference web pages and again anchor texts and URLs corresponding to them on the child pages are stored.
- When a search string is entered by the user, the crawler matches the string with stored anchor texts and takes all the anchor texts and corresponding links matching with the search string.
- Starting from the seed URL, the crawler makes a tree structure in a hierarchical form of all the matched anchor texts and corresponding links.
- In this manner we form a directed graph structure of the anchor texts and corresponding URLs .
- Since this is a topical web crawler basically intended to find the most relevant web pages corresponding to the user given search string, the technique uses weights as a criteria for sorting of these anchor texts and corresponding links.
- Weights are assigned to every node of the directed graph on the basis of incoming links form either parent links or other child links. For example in figure 2. Child 1.3 has a weight of 2 as there are two incoming links to Child 1.3 viz. Child 1 and Child 1.2.
- The crawler then traverses entire graph to find the node with the maximum weight. This node will be containing the link to the most relevant result corresponding to the user given search string. For example in figure.2. Child 1.3.1 has weight 5 hence will be of highest relevance.
- The other nodes of the directed graph will be sorted on the basis of decreasing weight corresponding to the anchor text nodes.
- In case of same weight the node lower in hierarchy in the directed graph will be more relevant.
- In this way we will get the sorted list of URLs and anchor texts on the basis of weighted anchor texts. This list will be in the decreasing order of relevance with respect to the given search string.

3.1.2 Advantages Over Existing Systems:

With the increasing usage of anchor text, more and more web sites are improving the quality of anchor texts on their web pages. Earlier most of the search engines used meta tags for searching and sorting of results. But many a times meta tags don't provide proper information about the actual contents of the page. Whereas Anchor Text if properly used can be a perfect information guider of the actual contents of a web page.

Our algorithm uses anchor text as the base for crawling and the assignment of weights to them on the basis of link structure of web pages. Thus it is one of the best possible ways of using anchor text for crawling purposes.

If this algorithm is used with the existing algorithms used by the search engines then the quality of results corresponding to search strings would surely increase to higher levels.

Time complexity with this algorithm is also not a problem as only sorting of the relevant anchor text takes the time and rest of the process is a continual process and extraction of anchor texts certainly takes less time as compared to meta tag generation and then search string from the meta tag.

3.2 Web page change detection:

Here we present a mechanism for detecting changes to HTML documents based on creating a Simplified hierarchy tree of the HTML document. In brief we sum up our research in the following steps

- Focusing on providing standard change detection mechanisms.
- Operators for more advanced change detection.
- Techniques to monitor different aspects of a page, including content, tag types, attributes, and structure.
- A set of unique optimization techniques that dramatically improves the performance of the system.
- After creating a hierarchical tree form of the web page, we establish weight age criterion for each level. We have considered an arbitrary value as weight age on basis of font size.
- The weightage is used to calculate checksum of various levels of the document as well as of the entire document, which help in determining the changes in the web page.
- The hierarchical tree format of the web pages are compared to one another to map structural changes.

3.2.1 Creation of a hierarchical tree structure:

A HTML document consists of various logical sections. For example in figure 3 we have identified levels based on a general view of the page.

Everything in an HTML document is a node. Nodes have a hierarchical relationship to each other. All nodes in an HTML document form a document tree (or node tree) as shown in figure 4. The tree starts at the document node and continues to branch out until it has reached all text nodes at the lowest level of the tree.

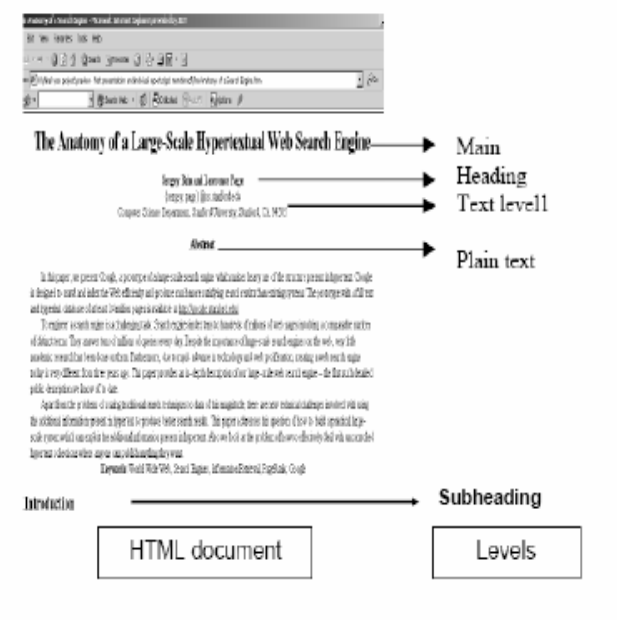


Figure 3-Identifying levels in a web page

In order to explain the creation of a hierarchical tree structure we consider the web page shown in figure 5 along with its html source code shown in figure 6. In the example a web page document starts with a title tag which contains the title of the page. Then we have a body tag which contains the entire content of the page. Certain tag types are used while creating a document tree from the corresponding web page while certain are rejected. Each node in the tree representing an HTML document has a type (e.g., paragraph, section, etc.) and possibly a number of attributes. Each node is also associated with a list of chunks of text interspersed with sub-nodes. The nodes are selected from category 1 tag types as shown in table 1.

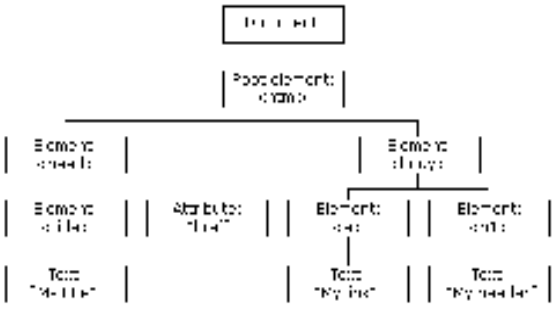


Figure 4 – Hierarchical tree structure

Tags category 1	Tag category 2
Title, Meta, H1, H2, H3 H4, H5, H6, pre, center Blockquote, dir, menu, Dt, dl, dd, ul, ol, li, table, Caption, thead, tbody, tr, Td, b, i	Isindex, base, link, script, style HR, DIV, form, big, small,map input, select, textarea, strike, code, samp, kbd, var, cite, applet, font, basefont

Table 1 HTML Tags classification

In figure 5 we can see the snapshot of the web page and determine the basic layout structure in terms of the html tags. The source code of the same page in figure 6 gives us a clearer view of the tag structure and hence we can verify the document tree created in figure 7.

The document tree of the html document starts with creation of a document root note to which head and body are attached. From table1, we use category 1 to create the tree structure as shown in figure 7.

The Anatomy of a Large-Scale Hypertextual Web Search Engine

Sergey Brin and Lawrence Page
{sergey, page}@cs.stanford.edu
Computer Science Department, Stanford University, Stanford, CA 94305

Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/

To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large scale search engines on the web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and web proliferation, creating a web search engine today is very different from three years ago. This paper provides an in-depth description of our large scale web search engine -- the first such detailed public description we know of to date.

Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results. This paper addresses this question of how to build a practical large scale system which can exploit the additional information present in hypertext. Also we look at the problem of how to effectively deal with uncontrolled hypertext collections where anyone can publish anything they want.

Keywords: World Wide Web, Search Engines, Information Retrieval, PageRank, Google

1. Introduction

Figure 5 – web page example

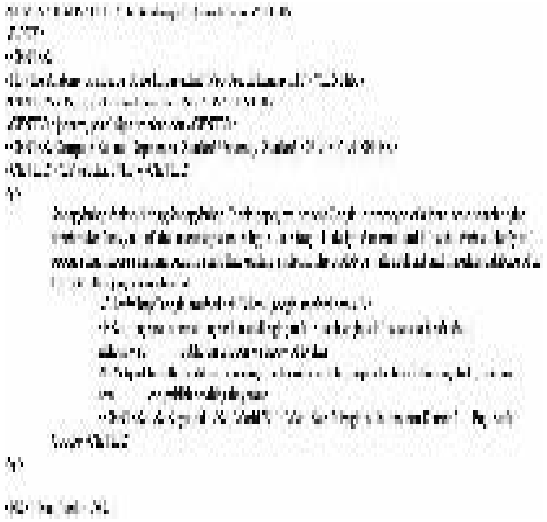


Figure 6 – source code view

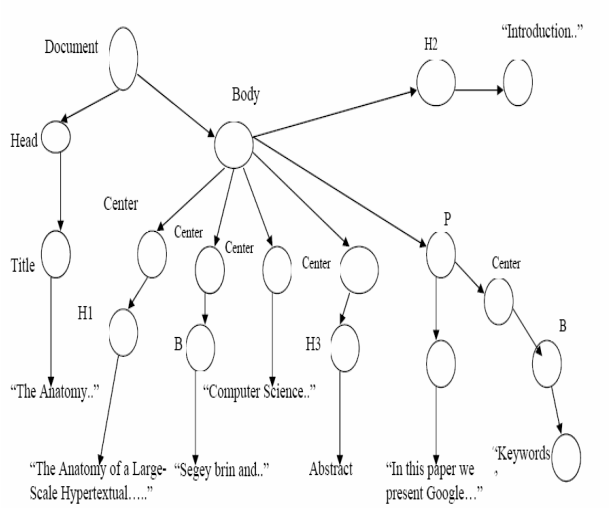


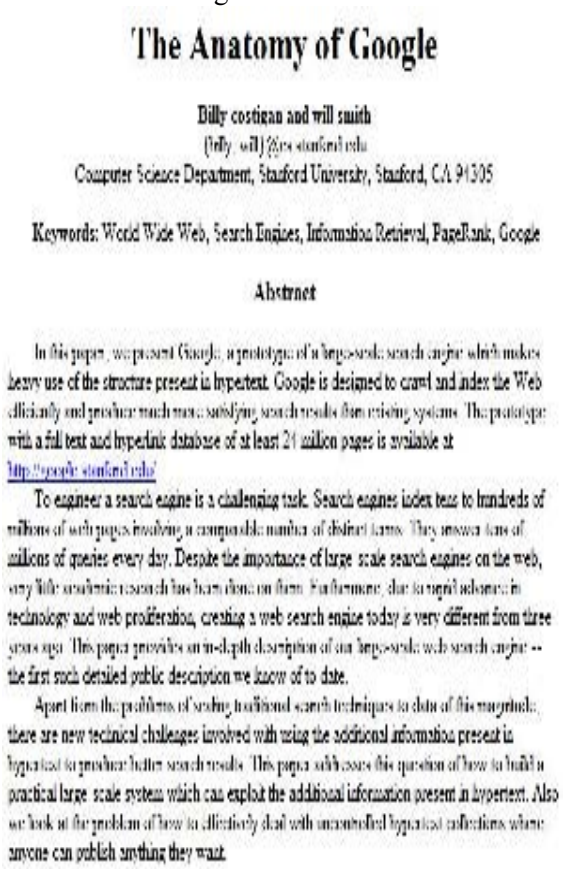
Figure 7 document tree-A

3.2.2 Tree comparison

After learning how to create a document tree of an html web page document, we proceed to map structural changes in a tree by recursive comparison techniques. Starting at the root, we compare nodes in two or more document trees recursively. If the type attributes, or number of sub-nodes differs between corresponding nodes in the document tree, they are considered a "different". Otherwise, we compare the text chunks between sub-nodes in sequence using a checksum evaluation algorithm which is discussed later in section 3.3. If any of them

differ, again, the entire node is also considered a "different".

In the figure 8 we have the changed instance of the web page document presented in figure 5. The document has undergone structural changes as well as textual changes. The document tree of this changed instance is also constructed as in figure 9.



The Anatomy of Google

Billy costigan and will smith

Computer Science Department, Stanford University, Stanford, CA 94305

Keywords: World Wide Web, Search Engines, Information Retrieval, PageRank, Google

Abstract

1. Introduction

Figure 8 – Changed instance of a web page

The document tree is constructed exactly as described earlier; the changed have been denoted in red. We can see that certain text has changed as well as the order of text has changed i.e. structural change. The "keywords" which appear later in figure 5 appear before the paragraph in figure 8.This structural change has been mapped in the document tree and also denoted by red as in figure 9. Thus the algorithm can easily map structural changes, for evaluation of textual change we use a checksum evaluation algorithm which calculates checksum at each node irrespective of any structural change or not. If there's no structural change for a node, its checksum value is calculated using algorithm described in section 3.2.3.

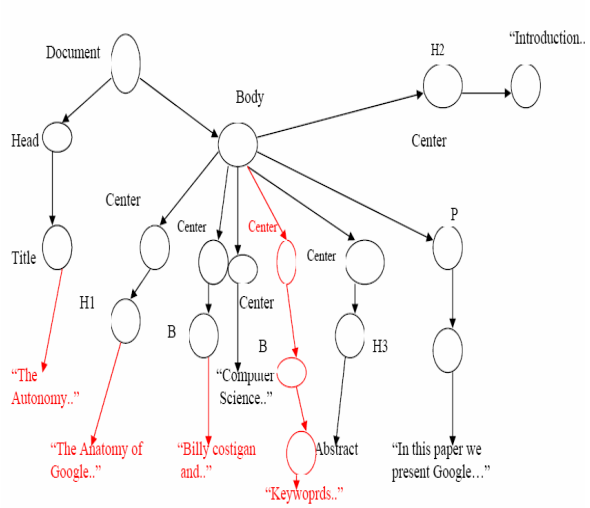


Figure 9– document tree – B

3.2.3 Checksum evaluation

- The value of checksum at each node of document tree is evaluated by using formula which contains parameters for attaching importance to text change, scale factor and ASCII code of characters.

- The checksum value of each node give us a detailed view of the "change" and help in comparing change difference with respect to various documents.

- The final checksum of the entire document is summation of values at different levels. This final checksum value, give us a summarized view of changes when we compare documents.

- Checksum calculation = $\sum$ i parameter * ASCII code * k-factor

  The value is calculated for each node by traversing the entire document tree and maintaining the checksum value of the nodes and the total value in a table.

Where,

- i parameter = importance to change parameter,
- ASCII code = ASCII code of each character,
- K-factor = scaling factor (which we have randomly initialized to be .1)

For our example, we have used font size as the I parameter i.e. we have taken significance of text as its font size, as the text size is large we take it to be more important and ergo, change to it is more important than smaller font size text.

Node name	Checksum value		
Document (total value)	6956.42		
Head	200.23		
Body	Center 1	303.34	
	Center 2	235.24	
	Center 3	174.66	
	Center 4	41	
	2 5	2 7	5748.4
		Center 6	414.35
	22 5	148.34	

Figure 10 - Checksum table value - A

In the above figure we have a table created on the basis of document tree structure drawn in figure 7 for the web page document in figure 5.The document node gives the total value of the checksum for the entire web page which gives us a summarized view of the overall change in when we compare various documents. An instance of the same page is taken which has a few textual and structural changes. The amount of change has been limited as to offer clear explanation of change on a certain page. Figure 11 gives the checksum table value for figure 6 web page document which has figure 9 as the respective document tree.

Node name	Checksum value	
Document (total value)	6941.57	
Head	170.5	
Body	Center 1	303.34
	Center 2	300.34
	Center 3	414.35
	Center 4	174.66
	Center 5	41
	2 5	5748.04
	22 7	148.34

Figure 11 - Checksum table value - B

From the document row of the tables we can get the value of overall change and compare it with various instance of the same document to map the change whether textual or structural.

4. Experiments and Observations :

According to various research papers and web material, we found that proper use of anchor text can help in showing the relevancy of web pages to key search engines to help rank for desired keywords. The usage of anchor text has significantly increased in searching algorithms. Thus many web sites use anchor text optimization for better search results. Anchor text optimization is the process of optimizing anchor texts using keywords to get higher rankings in search engines. Basically any anchor text optimization campaign would involve using keywords to link to pages having related content. All major search engines place huge level of importance to anchor texts in comparison to mere links.

To check the relevance of anchor text, we first made a search engine which we used over our local university server. We provided various search strings related to various subjects. 8 out of 10 times we got expected results. This particular experiment and observation proved that the anchor text is the best way to get most relevant data out of a searching operation.

Our next experiment was to check whether our algorithm was providing the desired results or not. To check this we made a crawler based on the given algorithm and checked it over our university website www.jiit.ac.in. We provided the name of our department head as our search string. There were in total 5 anchor texts which matched with the given search string. With the searched anchor texts, the crawler made a directed graph starting from the root URL. After sorting the crawler gave its results and the page that was assigned highest weight was apparently the most relevant page, which we confirmed manually. We repeated the process with various other search strings and got similar results. Hence we found the algorithm to be providing relevant results which we counter checked manually.

Our next step was to check the algorithm over some dynamic web sites like www.rediff.com, www.yahoo.com etc. We

extracted all the links of www.rediff.com up to 5 levels and stored it into the file. We had approximately 10000 anchor texts and corresponding URLs. Again we provided search strings in our searching application whose crawler was implemented using the above given algorithm. With 10000 anchor texts to search for and multiple links to same pages, when we entered search string, directed graph was formed and sorting was done according to the weights assigned to the anchor texts. The final result was mostly giving relevant pages related to the user provided search string.

The problem we faced was during the downloading and extraction of anchor texts and corresponding URLs as we had hardware and network constraints. Hence the database formation process took ample amount of time.

With the above results from the experiments performed we can conclude that usage of anchor text in a proper manner can surely help in better searching. Also, the algorithm proposed, if incorporated with existing systems can help in getting better results.

5. Results :

We found that the proposed algorithm is working properly if the anchor text of the web page is the accurate information provider of the contents of the web page. We also found that usage of anchor text as a basis for searching algorithm is best and most accurate technique if the anchor text is properly allocated to web pages.

The weights assigned to nodes of the directed graph are purely based on the link structure of the web pages. The accuracy claimed in the paper about results obtained after searching is based on the experiments performed on the various web sites as well as our university server. We have assumed that all the anchor texts are accurate descriptor of the contents of the web page to which they point. We have not taken into consideration enormous data as in that time complexity would be large.

Also we described work in applying techniques and ideas to efficiently map changes in web pages. The web page is modeled as a document tree following an algorithm as described in the paper. This document tree is constructed for all changed instances of a web page and compared recursively, thus giving the change in structure of the web page. The checksum evaluation algorithm gives us textual change value for each node and for the entire document which is represented in a tabular form. Focusing on the structure of the document and its textual content in a balanced manner, we obtain change detection methods that are both simple to implement and appear to work on a large variety of documents.

6 Conclusion and Future Work

From the above discussions it is clear that one cannot ignore the importance of anchor text in order to increase the page rank in any search engine. So for that matter our methodology can be of a great use in any searching algorithm. Knowing the increasing relevance of anchor text, we decided to work upon it and came up with a technique in which searching is purely based on anchor text. Weights are associated for each incoming link to a node and thus we have a weighted directed graph of anchor texts. Searching is done on this graph and consequently we have a list of anchor texts in descending order of their weights.

We have observed that this technique is giving relevant results most of the times but it might not be the case if we have a large data base or if searching is to be done on entire web. This is so because graph formation and its traversal becomes a tedious task. Also its time complexity is quite high but surely it is going to give commendable results if these factors are taken care of.

As for web change detection is concerned we presented an efficient technique which maps structural and textual changes in an html web page document. The algorithm works in two sections, the first section involves steps regarding the creation of a document tree for the respective html document. The second section deals with the checksum evaluation algorithm which maps the change in format of a numerical value which is tabulated on the basis of the node structure of the document tree. The checksum table gives the value of overall change as well as change at various node levels hence giving us a clear view of the change in the documents.

The tree-based comparison and checksum evaluation technique described in our paper provides us a convenient distance measure to determine value of change in web pages. Comparison could use the total checksum difference, but a simpler similarity measure is just the minimum of the fraction of each tree whose nodes are matched by corresponding nodes in the other tree (without taking into account the textual content). This similarity measure can then be used in a standard clustering algorithm, and clusters of pages correspond to groups of pages from which variables can be meaningfully extracted by the above procedures. Of course, structural document clustering of web pages may have other applications as well.

Another source of variability among web pages is the se of mark-up like bold face, italics, and sub- or superscripts. Those tags generally do not represent structural differences between text documents, but rather should be simply considered part of the textual content. If they are used as tags during the tree-based comparison, most of the time, the results will still be correct, but occasionally, an accidental alignment between those tags may cause only part of a semantically meaningful field to be returned. This can be addressed by treating such tags as part of the text and not considering them to be part of the page structure.

References:

[1] J. Cho, H.G. Molina, The Evolution of the Web and Implications for an Incremental Crawler, Proceedings of the 26th International Conference on Very Large Data Bases, Dec. 1999, p:200-209.

[2] S. Chakrabarti, K. Punera, and M. Subramanian, Accelerated Focused Crawling through Online Relevance Feedback, Proc. 11th Int'l World Wide Web Conference, Honolulu, USA, May 2002, p:148-159.

[3] S. Chakrabarti, M. van den Berg, and B. Dom., Focused crawling: A new approach to topic-specific web resource discovery. Computer Networks, Elsevier Science, 1999, p:545-562.

[4] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. Focused crawling using context graphs, In VLDB 2000, p: 527-534.

[5] C. Chung and C. L. A. Clarke, Topic-oriented collaborative crawling, In CIKM, 2002 pages 34-42,.

[6] J. Cho, H.G.Molina, "*Parallel Crawlers*", Proceedings of the 11th international conference on World Wide Web, May 2002, p : 362 – 363.

[7] J. Exposto, J. Macedo, A. Pina, A. Alves, and J. Ru_no., Geographical partition for distributed web crawling, In GIR, 2005, p: 55-60.

[8] O. Buyukkokten, J. Cho, H. Garcia-Molina, L. Gravano, and N. Shivakumar, Exploiting geographical location information of web pages, In WebDB (Informal Proceedings), 1999 p: 91-96.

[9] J. Ding, L. Gravano, and N. Shivakumar, Computing geographical scopes of web resources, Proceeding of the 26th VLDB conference,2000. p: 545-556.

[10]L.Gravano, V. Hatzivassiloglou, and R. Lichtenstein, Categorizing web queries according to geographical locality, in CIKM, 2003, p: 325-333.

[11]A. Markowetz, Y.-Y. Chen, T. Suel, X. Long, and B. Seeger, Design and implementation of a geographic search engine, in WebDB, 2005, p: 19-24.

[12]M. Hersovici, M. Jacovi, Y.S. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur, The Shark-Search Algorithm—An Application: Tailored Web Site Mapping, Proc. Seventh Int'l World Wide Web Conf., 1998.

[13] M. Iwazume, K. Shirakami, K. Hatadani, H. Takeda, and T. Nishida, IICA: An ontology Based Internet Navigation System, Proc. AAAI-96 Workshop Internet Based Information Systems, 1996.

[14]Odysseas Papapetrou, George Samaras, *Distributed Location Aware Web Crawlin*", WWW (Alternate Track Papers & Posters) 2004, p: 468-469.

[15] Z. Dalal, S. Dash, P. Dave, L. Francisco-Revilla, R. Furuta, U. Karadkar, F. Shipman, Managing Distributed Collections: Evaluating Web Page Changes, Movement, and Replacement, Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries, June 2004, p :160-168.

[16]W. Gao, H.C. Lee, Y. Miao, *Geographically Focused Collaborative Crawling*,

Proceedings of the 15th international conference on World Wide Web, May,2006 p : 287 - 296.

[17]G. Pant, P. Srinivasan, *Link Contexts in Classifier-Guided Topical Crawlers*, IEEE Transactions on Knowledge and Data Engineering, 18(1), January 2005, p : 107-122.

[18]M. Jamali, H. Sayyadi, B. B. Hariri and H. Abolhassani, A Method for Focused Crawling Using Combination of Link Structure and Content Similarity, Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'O6), p : 753-756 .

[19]J. Cho, H.G.-Molina, Department of Computer Science, Stanford, The Evolution of the Web and Implications for an Incremental Crawler, CA 94305, December 2, 1999.

[20]J. Cho, H.G.Molina, Department of Computer Science, Stanford, Effective page refresh policies for web crawlers. *ACM TODS 28*, 4 (2003), p:390–426.

[21]Khoury, Imad El-Mawas, Rami M. El-Rawas, Oussama Mounayar, Elias F. Artail, Hassan, An Efficient Web Page Change Detection System Based on an Optimized Hungarian Algorithm. Presented at: Knowledge and Data Engineering, IEEE Transactions Volume: 19, Issue: 5 On p: 599-613.

[22]Q. Tan, Z. Zhuang, P. Mitra and C. Lee Giles, Designing Efficient Sampling Techniques to Detect Webpage Updates, Proceedings of the 16th international conference on World Wide Web table of contents Banff, Alberta, Canada.

[23]J. Cho, University of California, Los Angeles and Hector Garcia- Molina, Stanford University, Estimating Frequency of Change, ACM transaction on internet technology, vol3, issue 3, Aug 2003. P:256-290.

[24]F. Douglis and T. Ball, Tracking and viewing changes on the web, In USENIX Annual Technical Conference 1996, p: 165– 176.

[25]Y. Wang, David J. DeWitt, Jin-Yi Cai, X-Diff: An Effective Change Detection Algorithm for XML Documents, 19th International Conference on Data Engineering (ICDE'03), 2003.

[26]CHO, J., and Ntoulas, A., Effective change detection using sampling, In *VLDB '02* (2002).

[27]D. Buttler, D. Rocco and L. Liu, Efficient Web Change Monitoring with Page Digest, World Wide Web Conference, New York, NY, United States, May 17 - May 22, 2004.

[28]L. Liu, Calton Pu, W. Tang, WebCQ–Detecting and Delivering Information Changes on the Web, IEEE Conference on Information and Knowledge Management, Washington, DC, November 2000.

[29] Divakar Yadav, A.K. Sharma & J.P.Gupta, "Change Detection in Web pages", IEEE Proceeding of 10th International Conference on IT, Dec 17-20, 07, Rourkela (India). ISBN: 0-7695-3068-0, Page(s) 265-270.

[30] Divakar Yadav, A.K. Sharma & J.P.Gupta, "Architecture for parallel crawler and algorithm for web page change detection", IEEE Proceeding of 10th International Conference on IT, Dec 17-20, 07, Rourkela (India). ISBN: 0-7695-3068-0 Page(s) 258-264.