REVIEW ON SEARCH ENGINE OPTIMIZATION

Sabresh.V[#1], P.Usha[*2]

[#1,2]*Department of Computer Science, Dr N.G.P Arts and Science College
Coimbatore-641048, Tamil Nadu, India*

---***---

ABSTRACT— Search engines are answer machines. They exist to urge , understand, and organize the internet's content so on supply the foremost relevant results to the questions searchers are asking. In order to means up in search results, your content must first be visible to look engines. This paper introduces SEO(Search Engine Optimization), working of it. As we all know main point of SEO is to rank websites using various techniques of SEO like Keyword Research, Link Building, Social Media Strategy so on enhance the Page Ranking of internet sites .The next optimization technique used is crawling and it's the invention process during which search engines send a team of robots (known as crawlers or spiders) to seek out new and updated content. The next optimization technique used is indexing. Search engines process and store information they find in an index, a huge database of all the content they've discovered and deem okay to serve to searchers.

Keywords—Search Engine optimization,Crawler,Page Ranking,Indexing.

1. INTRODUCTION

Search Engine (SE) may be a tool that permits users to locate information on the planet Wide Web. Search engines use keywords entered by users to hunt out websites which contain the knowledge sought. When someone performs an enquiry, search engines scour their index for highly relevant content then orders that content within the hopes of solving the searcher's query. This ordering of search results by relevance is understood as ranking. generally, you'll assume that the upper an online site is ranked, the more relevant the program believes that site is to the query. Crawling and indexing may be a prerequisite to exposure within the SERPs. If you've got already got an online site, it would be an honest idea to start out out off by seeing what percentage of your pages are within the index. this may yield some great insights into whether Google is crawling and finding all the pages you'd love it to, and none that you simply don't.

2. VIEW ON PROGRAM OPTIMIZATION

A web program could also be a software that's designed to seem for information on the earth Wide Web. The search results are generally presented during a line of results often mentioned as program results pages (SERPs). The knowledge could even be a mix of web sites , images, and other kinds of files. Some search engines also mine data available in databases or open directories. Unlike web directories, which are maintained only by human editors, search engines also maintain real-time information by running an algorithm on an online crawler.

Search engine optimization (SEO) is that the tactic of affecting the visibility of a web site or a web page during a search engine's "natural" or un-paid ("organic") search results. generally, the earlier (or higher ranked on the search results page), and more frequently a site appears within the search results list, the more visitors it'll receive from the search engine's users. SEO may target differing types of search, including image search, local search, video search, academic search, news search and industry-specific vertical search engines.

3. SEARCH ENGINE ARCHITECTURE OVERVIEW

In this section, we'll provides a high level overview of how the entire system works as pictured the above Figure . Further sections will discuss the applications and data structures not mentioned during this section. Most of Google is implemented in C or C++ for efficiency and will run in either Solaris or Linux. In Google, the web crawling (downloading of web pages) is completed by several distributed crawlers. there's a URL server that sends lists of URLs to be fetched to the crawlers. the online pages that are fetched are then sent to the storeserver. The storeserver then compresses and stores the online pages into a repository. Every website has an associated ID number called a docID which is assigned whenever a replacement URL is parsed out of an online page. The indexing function is performed by the indexer and thus the sorter. The indexer performs sort of functions. It read the repository, uncompresses the documents, and parses them. Each document is converted into a gaggle of word occurrences called hits. The hits record the word, position in document, an approximation of font size, and capitalization. The indexer distributes these hits into a gaggle of "barrels", creating a partially sorted forward index. The indexer performs another important function. It parses out all the links in every website and stores important information about them in an anchors file. This file contains enough information to figure out where each link points from and to, and thus the text of the link. The URLresolver reads the anchors file and converts relative URLs into absolute URLs and successively into docIDs. It puts the anchor text into the forward index, related to the docID that the anchor points to. It also generates a database of links which are pairs of docIDs. The links database is employed to compute PageRanks for all the documents.

The sorter takes the barrels, which are sorted by docID and resorts them by wordID to urge the inverted index. this is often exhausted place so as that tiny temporary space is required for this operation. The sorter also produces a listing of wordIDs and offsets into the inverted index. A program called DumpLexicon takes this list in conjunction with the lexicon produced by the indexer and generates a replacement lexicon to be employed by the searcher. The searcher is pass by an online server and uses the lexicon built by DumpLexicon in conjunction with the inverted index and thus the PageRanks to answer the queries.

4. CRAWLING THE WEB

Running a web crawler is a challenging task. Crawling is that the foremost delicate application since it involves interacting with many thousands of web servers and various name servers which are all beyond the control of the system.

In order to scale to many many sites, Google features a fast distributed crawling system. A single URLserver serves lists of URLs to variety of crawlers. Both the URLserver and therefore the crawlers are implemented in Python. Each crawler keeps roughly 300 connections open directly. This is necessary to retrieve sites at a quick enough pace. At peak speeds, the system can crawl over 100 sites per second using four crawlers. This amounts to roughly 600K per second of knowledge. A major performance stress is DNS lookup. Each crawler maintains a its own DNS cache so it doesn't got to do a DNS lookup before crawling each document. Each of the hundreds of connections can be during a number of various states: looking up DNS, connecting to host, sending request, and receiving response. These factors make the crawler a posh component of the system. It uses asynchronous IO to manage events, and variety of queues to maneuver page fetches from state to state. It seems that running a crawler which connects to quite half 1,000,000 servers, and generates tens of many log entries generates a good amount of email and phone calls. Because of the vast number of individuals approaching line, there are always those that don't know what a crawler is, because this is often the primary one they need seen. Almost daily, we receive an email something like, "Wow, you verified many pages from my web site. How did you like it?" There are also some folks that do not realize the robots exclusion protocol, and think their page should be shielded from indexing by a handout like, "This page is copyrighted and will not be indexed", which needless to mention is difficult for web crawlers to know . Also, due to the large amount of knowledge involved, unexpected things will happen. For example, our system tried to crawl a web game. This resulted in many garbage messages within the middle of their game! It seems this was a simple problem to repair . But this problem had not come up until we had downloaded tens of many pages. Because of the immense variation in sites and servers, it's virtually impossible to see a crawler without running it on large a neighborhood of the online. Invariably, there are many obscure problems which can only occur on one page out of the entire web and cause the crawler to crash, or worse, cause unpredictable or incorrect behavior. Systems which access large parts of the web got to be designed to be very robust and punctiliously tested. Since large complex systems like crawlers will invariably cause problems, there must be significant resources dedicated to reading the e-mail and solving these problems as they're available up.

5. INDEXING THE WEB

Parsing -- Any parser which is meant to run on the whole Web must handle an enormous array of possible errors. These range from typos in HTML tags to kilobytes of zeros within the middle of a tag, non-ASCII characters, HTML tags nested hundreds deep, and an excellent sort of other errors that challenge anyone’s imagination to return up with equally creative ones. For maximum speed, rather than using YACC to get a CFG parser, we use flex to get a lexical analyzer which we outfit with its own stack. Developing this parser which runs at an inexpensive speed and is extremely robust involved a good amount of labor . Indexing Documents into Barrels -- After each document is parsed, it's encoded into sort of barrels. Every word is converted into a wordID by using an in-memory hash table. Once the words are converted into wordID’s, their occurrences within this document are translated into hit lists and are written into the forward barrels. The main difficulty with parallelization of the indexing phase is that the lexicon must be shared. Instead of sharing the lexicon, we took the approach of writing a log of all the extra words that weren't during a base lexicon, which we fixed at 14 million words. That way multiple indexers can run in parallel then the tiny log file of additional words are often processed by one final indexer. Sorting -- so as to get the inverted index, the sorter takes each of the forward barrels and sorts it by wordID to supply an inverted barrel for title. This process happens only one barrel at a time, thus requiring little temporary storage. Also, we parallelize the sorting phase to use as many machines as we've just by running multiple sorters, which may process different buckets at an equivalent time. Since the barrels don’t fit into main memory, the sorter further subdivides them into baskets which do fit into memory supported wordID and docID. Then the sorter, loads each basket into memory, sorts it and writes its contents into the short inverted barrel and therefore the full inverted barrel.

6. THE RANKING SYSTEM

Google maintains far more information about web documents than typical search engines. Every hitlist includes position, font, and capitalization information. Additionally, we think about hits from anchor text and therefore the PageRank of the document. Combining all of this information into a rank is difficult. We designed our ranking function in order that no particular factor can have an excessive amount of influence. First, consider the only case -- one word query. In order to rank a document with one word query, Google looks at that document's list for that word. Google considers each hit to be one of several differing kinds (title, anchor, URL, plain text large font, plain text small font, ...), each of which has its own type-weight. The type-weights structure a vector indexed by type. Google counts the amount of hits of every type within the list . Then every count is converted into a count-weight. Count-weights increase linearly with counts initially but quickly taper off in order that quite a particular count won't help. We take the scalar product of the vector of count-weights with the vector of type-weights to compute an IR score for the document. Finally, the IR score is combined with PageRank to offer a final rank to the document.

For a multi-word search, things is more complicated. Now multiple hit lists must be scanned through directly in order that hits occurring approximate during a document are weighted above hits occurring far apart. The hits from the multiple hit lists are matched up in order that nearby hits are matched together. For every matched set of hits, a proximity is computed. The proximity is predicated on how far apart the hits are within the document (or anchor) but is assessed into 10 different value "bins" ranging from a phrase match to "not even close". Counts are computed not just for every sort of hit except for every type and proximity. Every type and proximity pair has a type-prox-weight. The counts are converted into count-weights and that we take the scalar product of the count-weights and therefore the type-prox-weights to compute an IR score. All of those numbers and matrices can all be displayed with the search results employing a special debug mode. These displays are very helpful in developing the ranking system.

7. CONCLUSION

Search Engine is basically useful gizmo in present era of web. There are many of search engines available in market, but hottest program is Google. So for getting to pmost results in web, we have to use search engine optimization technique. Both on page and off page program optimization techniques are important for better search result.

8. REFERENCES

1. Ayush Jain,"The Role and Importance of Search Engine and Search Engine Optimization",june 2013,International journal of emerging trends & technology in computer science(IJETTCS)

2. Sergey Brin and Lawrence Page,"The Anatomy of a Large-Scale Hypertextual Web Search Engine",february 2018,International Journal of Engineering Science & Research Technology(IJESRT).

3. Vinit Kumar Gunjan,pooja,monika kumar,"search engine optimization with google",jan 2012,International Journal of Computer Science Issues,(IJCSI).

4. ranveet singh,ajay bansal,"impact of search engine optimization on marketing tool",march 2018,jindal journal of business rearch.

5. Shraddha Londhe , Dr. Hemant Deshmukh,"review paper on serach engine optimization,APRIL 2017,International Journal of Engineering Science and Computing,(IJESC).