ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

# Mobile Focused Crawler using K-Means Clustering, TF-IDF and BITMAP Index

Neetu Kumari, Gurpreet Singh Saini, Dr. Vivek Kumar

M.Tech (CSE) Student, Delhi College of Technology & Management, Palwal, Haryana, India

Assistant Professor, Dept. of CSE, Delhi College of Technology & Management, Palwal, Haryana, India

Director, Delhi College of Technology & Management, Palwal, Haryana, India

**ABSTRACT:** Search engines became vital tools for internet navigation however, even smart devices like smart phones, tablets and fablets are also a source of data repositories now a days, so as to supply powerful search facilities, search engines maintain comprehensive indices of documents offered on the online. The creation and maintenance of internet indices is completed by internet crawlers, that recursively traverse and transfer sites on behalf of search engines. Analysis of the collected info is performed once the information has been downloaded. In this analysis, we tend to propose an alternate, a lot of economical approach to assembling internet indices supported mobile crawlers which also will crawl in smart devices. Our projected mobile crawler's area unit transferred to the source(s) wherever information resides so as to separate out any unwanted data domestically before transferring it back to the computer programme for search on respective information. This reduces network load and hastens the compartmentalisation section within the computer programme. Our approach to internet travel is especially like minded for implementing supposed "smart" searching algorithms that verify economical navigation path on various resources web and devices supported the contents of sites and apps that are visited thus far, so as to demonstrate the viability of our approach we've designed a mobile focused crawler for mobile phone where will fetch the data from mobile phones and further using K-Means clustering will form a document cluster and TF-IDF for frequency formation and BITMAP indexing for searching the documents respectively on key phrases.

**KEYWORDS**: K-Means Clustering, TF-IDF, BITMAP index, Mobile Crawlers.

## I. INTRODUCTION

The World Wide net (Web) could be a giant distributed object-oriented database management system, consisting of associate in nursing calculable one.6 million sites as of Apr 2017 as a result of its distributed and localised structure, just about anybody with access to the online will add new documents, links, and even servers for instance, estimates that four-hundredth of the Web's content modification at intervals a month. However, the online conjointly lacks structure. Users navigate at intervals this massive data system by following machine-readable text links, which connect totally different resources with each other. One in every of the shortcomings of this navigation approach is that it needs the user to traverse a presumably significant slice of the online before finding a selected resource (e.g., a document, that matches sure criteria). Considering the expansion rate of the online, locating relevant data during a timely manner is changing into progressively harder.

The current approach to extract and look information from mobile and smart device repositories is to make and maintain indices for the respective extracted documents and very like indices for a catalogue or access methods for tuples during an information and data retrieval. However, before the pages are indexed they need to first be collected and came back to the categorization engine. This can be done by mobile crawlers that consistently traverse the documents which exists using algorithms based on (e.g.-Means Clustering, TF-IDF and BITMAP Index). The documents will be downloaded to a search engine using bots and crawlers, that parses the text and creates and stores the index. For samples of net search engines see Google, Altavista, Infoseek, etc.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

However, the above effective measure many issues related to this technique of categorization that evolve as a result of the document growing variety of smart devices that ought to be indexed on one hand, and also the comparatively show increase in network information over the mobile world on the opposite side of information retrieval. However, as an instance these issues, allow us to cross-check the subsequent statistics even at intervals the last few years, mobile usage had to scale dramatically to stay up with the growing quantity of knowledge out there on the network resource and on smart devices. Thus, so as to stay the indices of a search engine up to now, crawlers should perpetually retrieve information or focused information from mobile phones as quick as potentially like web crawlers.

## II. RELATED WORK

In this section we will present a lot of information about web crawlers and their literature review. In the first place, we will mention some historical data about them and we will give them their definition in order to make this diplomatic work more understandable, we will describe the existing crawling policies and analyse them. Crawling policies relate to how the crawler chooses the next page to visit, and logically, the policy that each crawler follows plays a catalytic role in its effectiveness.

Web crawlers - also known as robots, spiders, worms, walkers and wanderers - are as old as the Web itself [7]. The first crawler, by **Matthew Gray Wanderer**, was written in the spring of 1993. Several papers on web crawling were presented at the first two conferences that took place on the World Wide Web [8, 9, and 10]. However, at that time, the Web was much smaller than today, which made those systems not to experience "scaling" problems, as they are today. The web crawler is a program that searches the World Wide Web in a methodical and automated manner. Web crawlers are mainly used to construct a copy of the pages we've visited. This copy can then be further elaborated by a search engine that will categorize the downloaded pages to provide a very fast search.

It is reasonable that all known search engines use crawlers. However, due to the competition that exists between search engine companies, the plans of these crawlers have not been publicly described. There are two notable exceptions: Google crawler and Internet Archive crawler. Of course for both crawlers, the descriptions in the bibliography are laconic, thus preventing their ability to replicate.

The Google search engine is a distributed system that uses multiple crawling machines [11, 12]. The crawler consists of five operating systems running in different processes. A URL server process reads URLs from a file and sends them to multiple crawler processes. Each crawler process runs on a different machine, is a single thread and uses asynchronous I / O to receive data from 300 servers in parallel. Crawlers transfer downloaded pages to a Store Serve process, which compresses the pages and stores them in the disk. The pages are then read from the disk by an indexer process, which extracts links from HTML pages and stores them in a different file on the disk. A resolver URL process reads the link file, finds the absolute URL of each one, stores it, and then reads it from the URL server. Typically, three to four crawlers are used, i.e. the entire system requires four with eight crawlers.

Internet Archive also uses multiple machines to crawl the Web [13, 14]. Each crawler process is assigned more than 64 sites to crawl, and no sites are assigned to more than one crawler. Each single-threaded process reads queues from the disk by URLs that correspond to the sites assigned to it. It then uses asynchronous I / O to extract parallel pages from these queues. Each time a page is downloaded, the crawler extracts the links it contains. If a link refers to the page it was found in, it is added to the appropriate queue; otherwise it is stored on the disk. Periodically, a batch process combines these URLs stored in the disk by deleting their double impressions [4].

Finally a web crawler is a kind of bot or software agent. Generally, it starts with a list of Urls to visit. As he visits each of these Urls, he finds all his links and adds them to the list of urls he will visit. It searches the Web repeatedly for a set of policies that we describe below.

There are two important web features that make web crawling very difficult: a) its large volume; and b) its frequency of change, as a huge percentage of pages are added, changed and subtracted every day. Also, network speed has improved less than processing speeds and storage capabilities.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

The large volume of the web means that the crawler can download only a percentage of pages within a given time, making it necessary to prioritize the download pages. Additionally, the high frequency of web page changes results in new pages added to a site or existing pages being refreshed or deleted until the crawler has downloaded the last pages of that site.

As Edwards and others have said, "Since the breadth of crawling is not infinite or free, it is necessary to crawl the Web efficiently in order to achieve a reasonable level of quality and validity of information." A crawler must carefully select the pages to be visited at each step [15].

The behavior of a web crawler is the result of a combination of policies:

- A selection policy that determines which pages will be downloaded.
- A re-visit policy that determines when to look for page changes.
- A politeness policy that determines how to avoid overloading Web sites.
- A parallelization policy that defines how to work distributed web crawlers.

Given the current web size, even large search engines only cover part of the publicly available hardware. A study by Lawrence and Giles (Lawrence and Giles, 2000) showed that no search engine classifies more than 16% of the web. As a crawler only downloads part of the pages, it is highly desirable that this part contains the most relevant pages and not just a random sample of the web.

This requires the existence of a metric that defines the importance of a page and allows the web pages to be categorized by priority. The importance of a page is a function of its quality, its popularity in terms of its links or visits as well as its Urls (Urls affect the importance of a website in the case of "vertical" search engines, limited to A single high-level domain or search engines limited to a particular Website). Designing a good policy choice has an added difficulty: it should work for partial information, since the complete set of Web pages is not known during crawling.

## III. PROPOSED ALGORITHM

A. *Design Considerations:*

- Crawl Documents from phones memory card.
- Bot/Spider will established connection using sockets with mobile phone.
- Store all fetched document in repository.
- Transform respected document into text format.
- Impose K-Means Clustering
- Pre-Processing on extracted Data
- Generate TF-IDF
- Perform bitmap index on documents for searching based on TF-IDF.

B. *Description of the Proposed Algorithm vide K-Means Clustering*

**Step 1: K-Means Clustering**

1. Initialize the center of the clusters = $\mu_i$= some value, i=1, k
2. Attribute the closest cluster to each data point $c_i=\{j:d(x_j, \mu i)\leqslant d(x_j, \mu l), l\neq i, j=1,...,n\}$
3. Set the position of each cluster to the mean of all data points belonging to that cluster $\mu_i=1|c_i|\sum_j\in_{cixj},\forall^i$
4. Repeat steps 2-3 until convergence

Notation |c|= number of elements in c

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

**Step 2: TF-IDF formation:**

TF-IDF, short for term frequency–inverse document frequency, is a numeric measure that is use to score the importance of a word in a document based on how often did it appear in that document and a given collection of documents. The intuition for this measure is: If a word appears frequently in a document, then it should be important and we should give that word a high score. But if a word appears in too many other documents, it's probably not a unique identifier; therefore we should assign a lower score to that word. The math formula for this measure:

1. tfidf(t,d,D)=tf(t,d)×idf(t,D)tfidf(t,d,D)=tf(t,d)×idf(t,D)
2. idf( t, D ) = log \frac{ \text{| } D \text{ |} }{ 1 + \text{| } \{ d \in D : t \in d \} \text{ |} }

The numerator: D is infering to our document space. It can also be seen as D= d1,d2,…,d n d1,d2,…,dn where n is the number of documents in your collection. Thus for our example | D || D |, the size of our document space is 4, since we're only using 4 documents.

The denominator: | {d∈D:t∈d} || {d∈D:t∈d} | implies the total number of times in which term t appeared in your entire document d (the d∈Dd∈D restricts the document to be in your current document space). Note that this implies it doesn't matter if a term appeared 1 time or 100 times in a document, it will still be counted as 1, since it simply did appear in the document. As for the plus 1, it is there to avoid zero division.

**Step 3: Bitmap index integration**

1. $B_{map=}$[x][y] create 2Darray and set val=1 element to zero
2. For i=1:x{j=(vi mod x) set to $B_{map}$[i][j]=1
3. Return map;

## IV. PSEUDO CODE

**Step 1: Crawl Document from Memory**

1. Initialize a queue of URLs (seeds)
2. Repeat until no more URLs in queue:
3. Get one URL from the queue
4. If the page can be crawled, fetch associated page
5. Store representation of page
6. Extract URLs from page and add them to the queue
7. Queue = "frontier"

**Step 2: Establishing Sockets**

1. Create socket
2. Bind socket to a specific port where clients can contact you
3. Loop
4. (Receive UDP Message from client x) + (Send UDP Reply to client x)
5. Close Socket
6. Pseudo code UDP client
7. Create socket
8. Loop
9. (Send Message To Well-known port of server)+(Receive Message From Server)

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

10. Close Socket

**Step 3: K-Mean Clustering**

kNN (dataset, sample) {

1. Go through each item in my dataset, and calculate the "distance" from that data item to my specific sample.
2. Classify the sample as the majority class between K samples in the dataset having minimum distance to the sample.
3. Compute dataset containing indices for the K smallest distances d ($X_i$, x).
4. return majority label for ($Y_i$ where i ∈ I)

}

**Step 4: Bitmap index Search Model.**

Bitmap--Search(x, k) returns (y, i) such that keyi[z][y]
keyi[z][y] = k or nil
i ← 1
while i ⩽ n[x] and k > keyi[z][y]
do i ← i + 1
if i ⩽ n[x] and k = keyi[z][y]
then return (x, i)
if leaf[x]
then return nil
else Disk-Read(ci[x])
return B-Map-Search(ci[x], k[z][y]).

## V. SIMULATION RESULTS

The proposed mobile focused crawler was implemented on the following hardware and software specifications: OS Name- Microsoft Windows 7 Ultimate, Version-6.1.7600 Build 7600, Processor- Intel(R) Core(TM) i5-3230M CPU @ 2.60 GHz, 2601.and compared its performance with intelligent smart phone crawler which is link based crawler , after running both the crawlers in same hardware and software environment with same input parameters, observed following rate of precisions for both the crawlers as shown in Table 1. Evaluate the performance of proposed framework by comparing it with intelligent web crawler by applying both the crawlers to the same problem domains such as Book Show, Book to read, Cricket match and Match making. The experiments are conducted on different social aspects database (repositories of smart phones) having same words with different meaning as book meaning reserving some resource for the domain Book show while as book meaning a physical entity made up by binding pages for the domain book to read. From the above test cases it is vibrant that if correct searchable documents are provided according to domain sense of the word then: 1. the precision range comes out to be 20 to 70% which is fairly acceptable for the crawl. 2. The test results show that the search is now narrowed down and very specific results are obtained in sorted manner. 3. The sorting of the results cause the most relevant link to be displayed first to the user so as to save his valuable search time

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

| S.No | Search Document | File Size | Percentage Precision |
|---|---|---|---|
| 1 | PDF Documents | 4 MB | 74% |
| 2 | Word Documents | 2MB | 83% |
| 3 | Spread Sheet Documents | 3 MB | 76% |
| 4 | WDL Documents | 10 MB | 79% |

The testing of the framework is carried out on live web, thus by precision reported accuracy of proposed Mobile focused crawler is promising.
Above test results completely depend on various Smart Phones. The respected requests provide the correlated documents for the search.

## VI. CONCLUSION AND FUTURE WORK

Mobile Because one can always buy larger and faster phones and design more and more smart software to co-ordinate them, ultimately the limiting factor will be the bandwidth of the internet, since there is so much that a router can gather a data from mobile devices. And as the size of the network grows, the breadth of the internet with smart phones is becoming increasingly important. Unfortunately, even if network speeds increase, it is expected that the amount of crucial information will increase (text data may not increase, for example, multimedia data, for example).

If internet speed actually becomes an obstacle for web crawlers or mobile focus crawlers, then there are two directions for research. The first direction, in which work is already being done, designs algorithms that find what page should be refreshed and when, so that the whole database is kept up-to-date with less work. The second direction for research is geared towards the possibility of wide-area distribution, which has an intuitive appeal: a web crawler that is in France will be more effective in crawling the French pages. Fortunately, the cost of collaboration between nodes is relatively low (for every 20KB web pages downloaded, only a few urls need to be transferred). However, this assumes that it is ok to leave the data on the site we retrieved. If all pages are ultimately sent back to a central storage (e.g. for categorization) then the distribution has not benefited from anything. One help is that all the documents can be compressed to 60% of the size and will be fetched by the spiders or bots quickly from smart devices. Additionally, if only text is needed, total compression may be a factor of 10, which can make wide range distribution marginally effective to the user.

## REFERENCES

1. Kleanthis Thraboulides. Object-oriented Java programming. Volume B '.
2. Ch. Makris, E. Theodoridis, I. Panagis, A. Perdikouri and E. Christopoulou. Retrieving information
3. D. Boswell. Distributed High-Performance Web Crawlers: A Survey of the State of the Art, December 10, 2003.
4. A. Heydon and M. Najork. Mercator: A Scalable, Extensible Web Crawler, Compaq Systems Research Center 130 Lytton Ave., Palo Alto, CA 94301.
5. J. Cho and H. Garcia-Molina. The Evolution of the Web and Implications for an Incremental Crawler. Department of Computer Science, Stanford, CA 94305, December 2, 1999.
6. WebCrawler Timeline Http://www.thinkpink.com/bp/WebCrawler/History.html
7. The Web Robots Pages. Http://info.webcrawler.com/mak/projects/robots/robots.html
8. David Eichmann. The RBSE Spider - Balancing Effective Search Against Web Load. In Proceedings of the First International World Wide Web Conference, pages 113--120, 1994.
9. Oliver A. McBryan. GENVL and WWW: Tools for Taming the Web. In Proceedings of the First International World Wide Web Conference, pages 79--90, 1994.
10. Brian Pinkerton. Finding What People Want: Experiences with WebCrawler. In Proceedings of the Second International World Wide Web Conference, 1994.
11. Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. In Proceedings of the Seventh International World Wide Web Conference, pages 107--117, April 1998.
12. Google! Search Engine Http://google.stanford.edu/
13. Mike Burner. Crawling towards Eternity: Building an archive of the World Wide Web. Web Techniques Magazine, 2 (5), May 1997.
14. The Internet Archive. Http://www.archive.org/
15. Web crawler. Http://en.wikipedia.org/wiki/Web_crawler
16. Larbin Multi-purpose web crawler Http://larbin.sourceforge.net/index-eng.html
17. WebSPHINX: A Personal, Customizable Web Crawler Http://www.cs.cmu.edu/~rcm/websphinx
18. The Stanford WebBase Project Http://www-diglib.stanford.edu/~testbed/doc2/WebBase/