International Journal of Applied Engineering and Technology ISSN: 2277-212X (Online)
An Open Access, Online International Journal Available at http://www.cibtech.org/jet.htm
2017 Vol. 7 (1) January-March, pp.19-31/Rane and Pathak
Review Article

FOCUSED WEB CRAWLER WITH GENETIC ALGORITHM-AN APPROACH TO WEB MINING

***Devendra Tanaji Rane and Ganesh R. Pathak**
Department of Information Technology, Sinhgad College of Engineering, Pune, India
**Author for Correspondence*

ABSTRACT
Today internet is full of structured or unstructured information and this information influences people or society directly or indirectly. With the rapid growth of internet technologies, the web is considered as a world's largest repository of knowledge. Web data processing is the method of handling high volume of data which is not so easy. The rapid advent in internet technology has led the users to get easily confused in large hypertext structure. Fetching the relevant information from this huge web of structured data has become the need nowadays. In order to achieve this goal, we employ the concept of web mining. Focused Web Crawlers are getting developed to increase Precision of information Retrieval nowadays in search engines. Idea is to keep Crawler focused to the user interest towards the topic. Genetic Algorithm is the optimization technique, we have used in this paper to improve on relevance factor. It explains Genetic Algorithm's basic operations like Selection, Mutation and Crossover. It also explains how fitness calculations work in algorithm. Paper gives the detail architecture for the Focused Web Crawler along with the design details to explain how Genetic Algorithm can be used in focused crawler. Paper specifies the next steps about the actual implementation of "Focused Web Crawler with Genetic Algorithm-An Approach to Web Mining".

Keywords: *Web Mining, Focused Crawler, Search Engine, Genetic Algorithm, Precision, Selection, Mutation and Crossover, Fitness Calculation, Jaccard Similarity*

INTRODUCTION
The World Wide Web is a big set of data. The data keeps rising continuously round the clock. It is very vital to categorize data as important or unimportant in accordance with client query. Researchers are operational on techniques which would help to download related web pages.

a) Search Engines- Crawling & Indexing (Keole and Pardakhe, 2013)
Web Search Engine is a tool enabling document search, with respect to specified keywords, in the Web and returns a list of documents where the keywords were found. It processes textual and hyper-textual information in diverse ways, but the capability to quickly fetch a large number of Web pages into a local repository and to index them based on keywords is required by many applications. Large scale programs that fetch tens of thousands of web pages per second are called crawlers, spiders, web robots, or bots. Crawling is usually performed to subsequently index the documents fetched. Together, a crawler and an index form key components of a Web Search engines (Chakrabarti, 2012).
Components of Web Search Engine
1. User Interface
2. Parser
3. Web Crawler
4. Database
5. Ranking Engine

1) *User Interface*- It is the part of Web Search Engine interacting with the users and allowing them to query and view query results.
2) *Parser*- It is the component providing term (keyword) extraction for both sides. The parsers determine the keywords of the user query and all the terms of the Web documents which have been scanning by the crawler.
Term extraction procedure includes the following sub-procedures:

1. Tokenization
2. Normalization
3. Stemming
4. Stop word handling

3) *Web Crawler*- A web crawler is a relatively simple automated program, or script that methodically scans or "crawls" through Internet pages to create an index of the data it is looking for. Alternative names for a web crawler include web spider, web robot, crawler, and automatic indexer. When a web crawler visits a web page, it reads the visible text, the hyperlinks, and the content of the various tags used in the site, such as keyword rich meta tags. Using the information gathered from the crawler, a search engine will then determine what the site is about and index the information. Lastly, the website is included in the search engine's database and its page ranking process.

4) *Database*- It is the component that all the text and metadata specifying the web documents scanned by the crawler.

5) *Ranking Engine*- The component is mainly the ranking algorithm operating on the current data, which is indexed by the crawler, to be able to provide some order of relevance, for the web documents, with respect to the user query.

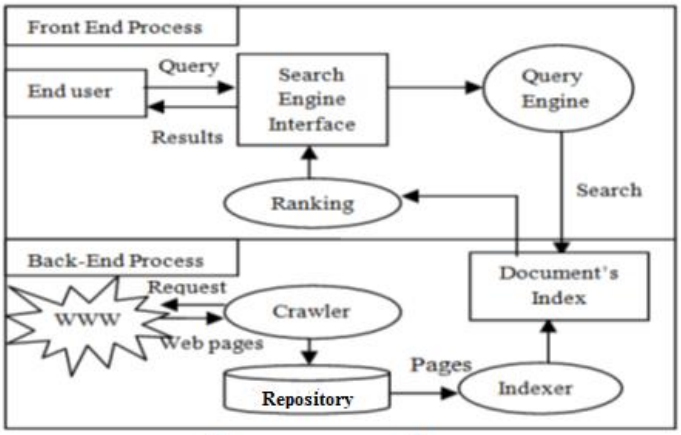

Figure 1: Web Crawler Architecture

b) Importance of Web Crawlers (Dahiwale et al., 2015)

Researchers say that the vast size of data outcomes in reduced coverage of total data while search is performed and it is anticipated that only 33% of the data gets listed into indexer. The web is so gigantic that even the number of important or appropriate web pages that get download is too huge to be discovered by the client. This scenario creates the requirement of downloading the most relevant and excellent pages first. Web search is presently creating near about 13% of the traffic to Web sites. Web crawler requires searching for information between web pages identified by URLs. If it can believe each web page as a node, then the WWW can be visualised as a data structure that looks like a Graph. To navigate a graph, crawler will require traversal mechanisms much nearly equal to those needed for traversing a graph like BFS or DFS, proposed Crawler follows a BFS approach.

The Crawler is the most vital part in the search engine. It can traverse the Web space by following Web page's hyperlinks and storing the downloaded Web documents in local repositories that will later be indexed and used to respond to the user's queries efficiently.

c) Crawler Types

The crawlers can be classified in to two types based upon the application. General Crawler and Focused (Topical) Crawler. The General Crawler serves as an entry point to web pages. It strives for coverage that is as broad as possible, whereas the Focused Crawler is built to retrieve the pages within a certain topic. In this case, the task of crawling could be constrained by programmer (Selvakumar and Vijaya, 2014). A

focused crawler or topical crawler is a web crawler that attempts to download only web pages that are relevant to a pre-defined topic or set of topics. Focused Crawler is kind of crawlers try to find high-quality information on a specific subject as soon as possible and try to avoid irrelevant pages in order to the results would be as accurate as possible. Thus, a focused crawler is a program which fetches as much as possible relevant pages.

It takes as input one or several related web pages and attempts to find similar pages on the web, typically by recursively following links in a best first manner. Ideally, the focused crawler should retrieve all similar pages while retrieving the fewest possible number of irrelevant documents. The goal of a focused crawler is to selectively seek out pages that are relevant to a pre-defined set of topics. The topics are specified not using keywords, but using exemplary documents. Rather than collecting and indexing all accessible web documents to be able to answer all possible 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.

d) What is Genetic Algorithm? (Dahiwale et al., 2015)

The focused crawling can be done by using Genetic Algorithm. The genetic algorithm is a kind of the searching algorithm. It penetrates the result place for the best possible solution to the tricky Problem. Genetic Algorithm uses the genetic operators which are selection, crossover and mutation. It produces the result for the succeeding generations. The process of genetic algorithm ended when the best possible solution is found. Applying the genetic algorithm for the web crawlers is possibly to produce an excellent outcome. It is one of the machine learning techniques which uses search heuristic that mimics the process of natural selection.

This paper proposes a focused crawling technique which makes use of genetic algorithm for optimum crawling. It uses Jaccard similarity measure to determine relatedness amongst searched result.

Literature Review

For search engines so many focused crawling programs are available, each of which are having advantages and disadvantages, some of them use the different probabilities for the particular input and they give good result by using the different mutation rates. The techniques presented in are applicable to any domain for which it is possible to generate term-based characterizations of a topic. It gives the total description about mutation rate. Chauhan and Kumar (2012) proposed the context model for the focused web search, it describes a Focused Crawler which looks for gain, make the index, and keep the collection of the pages on a particular area that represent a somewhat thin portion of the web. Thus, web substance can be handled by a scattered group of the focused web crawlers, each concentrating in one or a small number of areas. The focused crawler is directed by a category which discovers to be familiar with the relevance from the examples surrounded in a particular topic, Classification, and a distiller which discover relevant vantage points on the World Wide Web. Brin and Page (1998) said that it uses to grouping of the link structure analysis and the subject matter similarity while building their focused crawling. The idea behind that is based on the ordinary hyperlinks in pages is illustration to the authors' sight about additional pages. Also, the matter of the pages are the another source to relate them to the domain. In paper Deepa and Sivanandam, (2008) describe the whole genetic algorithm and gives the detail of genetic operators and working of the genetic algorithm. In paper, Mukhopadhyay *et al.*, (2009) is named as the Genetic Algorithm: A tutorial re-view it represent approximately all conservative technique search from a particular point. Genetic Algorithms all the times manage on an entire population. These add a lot of the toughness of the genetic algorithms. It decreases the risk of proper attentive in a local fixed place. In paper, Dahiwale *et al.,* (2015) have explained usage of genetic algorithm in focused crawler where they have used Jaccard similarity to calculate fitness for comparison. In paper, Selvakumar and Vijaya, (2014), has preferred SVM algorithm to do the hyperplane based classifications.

a) Focused Crawling

Crawlers are also known as spiders, web robots, bots etc. Habitually the crawling starts with a set of seed Uniform Resource Locator (URLs). These URLs are often relevant URLs. It is mandatory that these URLs must be as fine as possible, common practice is to search for the particular keywords on Google,

Yahoo, etc. and treat the first five to six URLs as the seed URL. Focused web crawling is a technique in web mining in which relevant information is mined using a general knowledge of interest and intelligent choice of links to follow or discard based on relevancy parameters. The task of crawling starts with a set of initial links known as seed pages (or seed links) that indicates the prior information about the content a user is interested in.

A focused crawler can be implemented in various ways (Patni *et al.*, 2014). Some of the approaches are shown below:

A. Priority based focused crawler: In a priority based focused crawler, the retrieved pages are stored in a priority queue instead of a normal queue. The priority is assigned to each page based on a function which uses various factors to score a page. Thus, in every iteration, a more relevant page is returned. This is mainly useful in distinguishing between important and unimportant information, wherein priority is given to a more important page.

B. Structure based focused crawler: Structure base focused crawlers take in account the web page structure when evaluating the page relevance. Its strategy is to compute the relevance score of the page with a predefined formula, then predict the relevance score of the link, and compute the authority-score of URLs in the queue to be crawled and determine their priority according to the comprehensive value of relevance-score and authority-score namely first crawl relevant and quality page.

C. Context based focused crawler: Many a times, when a user searches a particular topic on the web, the search system is unaware of the user's needs. For e.g.: If a user is looking for a college university, the search results may include references of that university even on a news portal. Such information becomes irrelevant to the user. This increases the work for the user to filter out unwanted data. To avoid this, we implement a context based focused crawler, which tries to understand the context of the user's needs by interacting with user and comprehending the user profile. The crawler then gets adapted to such contexts and uses them for future search requests.

D. Learning based focused crawler: Learning based focused crawler is a new learning based approach to improve relevance prediction in focused web crawler. Firstly, training set is built to train the system. Training set contains value of four relevance attributes:

a. URL word Relevancy
b. Anchor text relevancy
c. Parent page relevancy
d. Surrounding text relevancy

Secondly, they train the classifier (NB) using training set. After that trained classifier is used to predict the relevancy of unvisited URL.

b) General Architecture

A focused crawler has the following main components: (a) A way to determine if a particular web page is relevant to the given topic, and (b) a way to determine how to proceed from a known set of pages. An early search engine which deployed the focused crawling strategy was proposed based on the intuition that relevant pages often contain relevant links. It searches deeper when relevant pages are found, and stops searching at pages not as relevant to the topic. Unfortunately, the above crawlers show an important drawback when the pages about a topic are not directly connected in which case the crawling might stop pre-maturely.

A topical crawler ideally would like to download only web pages that are relevant to a particular topic and avoid downloading all others. A possible predictor is the anchor text of links; this was the approach taken by Pinkerton in a crawler developed in the early days of the Web (Bhatia *et al.*, 2009).

Figure 2 gives the high level overview of a simple focused crawler (Jayaratne and Samarawickrama, 2013). URL downloader downloads web pages from WWW initiated with the seed URLs and sends them to the classifier. Classifier which is trained with the help of seed set makes relevance judgments on the pages it receives. URLs extracted from these relevant pages will be added to the crawler frontier thus, to continue with the crawling process. URL downloader maintains a database (URL DB) of crawled pages. Upon retrieving a URL from the crawler frontier it will first check the URL DB to see whether that page

has already been downloaded or not. URL DB may contain a local copy of the downloaded pages and will also serve the indexing process.

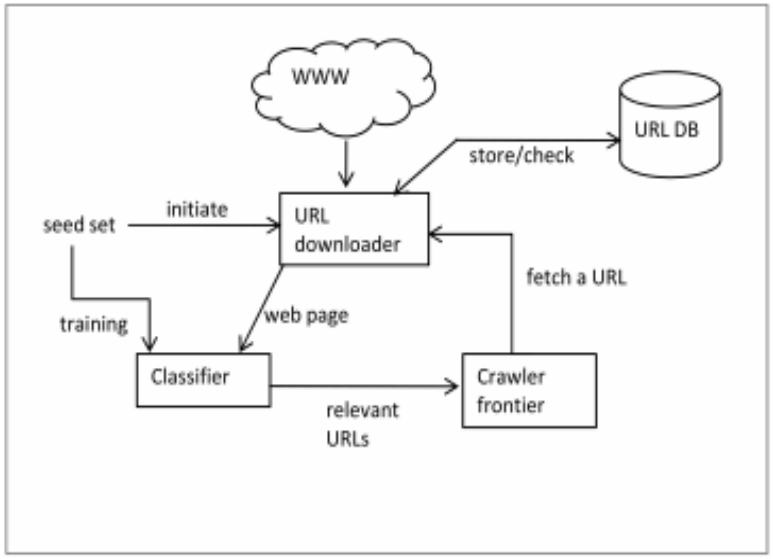

Figure 2: Architecture of a Simple Focused Crawler

Early work on focused crawling was based on simple keyword matching, regular expression matching or binary classifiers. De Bra proposed the Fish-search algorithm in which, crawling is simulated by a group of fish migrating the web. Each URL corresponds to a fish whose survivability is dependent on visited page relevance and remote server speed. Page relevance is estimated using a binary classification by using simple keyword or regular expression matches. After traversing a certain number of irrelevant pages, fish dies. Hersovici improves this algorithm into Shark-search in which the page relevance is calculated as a similarity between document and query in Vector Space Model (VSM). Cho proposed calculating the PageRank score on the graph induced by crawling the web and then using this score as a priority of URLs for the next crawl (about prioritizing the crawl frontier).

The two main approaches to modern focused crawling are based on content analysis and link structure analysis. Focused crawling based on content analysis is heavily dependent of automatic text classification techniques to determine the relevance of a retrieved web page to the crawling domain. Since web pages are noisy, these crawlers employ different noise removal techniques to extract only the useful textual content. Link analysis based approaches build a web graph and this graph is used to identify potential URLs that can lead to topic relevant pages.

Chakrabarti (2012) was the first to utilize machine learning into focused crawling. They used existing document taxonomy (Yahoo!) to train a Naive Bayes classifier and to classify retrieved web pages into categories. Use of a taxonomy helps in better modelling of irrelevant pages (the negative class). Distiller, another component of their crawler, identifies hub pages pointing to many topic relevant pages. Naive Bayes classifier has been used in other topical crawlers as well to name a few. Assis uses genre related information as well as the content related information. Crawler analyses a web page to identify the terms that correspond to genre and the topic separately, in which a traditional crawler doesn't identify as separate.

Bazarganigilani proposes a novel approach which uses genetic programming (GP) to efficiently discover the best similarity function which is a combination of Bag-of-words, Cosine, Okapi similarity measures. This is achieved with the fitness function used by the genetic algorithm. FOCUS is based on link structure analysis which uses link distance (how close a particular URL to the seed URLs) to rank pages directly rather than using a classifier to determine page relevance.

c) Genetic Algorithm

Genetic Algorithms were invented to mimic some of the processes observed in natural evolution. Many people, biologists included, are astonished that life at the level of complexity that we observe could have evolved in the relatively short time suggested by the fossil record. The idea with GA is to use this power of evolution to solve optimization problems. The father of the original Genetic Algorithm was John Holland who invented it in the early 1970's ("Genetic Algorithm," Online, 1996).

It is better than conventional AI in that it is more robust. Unlike older AI systems, they do not break easily even if the inputs changed slightly, or in the presence of reasonable noise. Also, in searching a large state-space, multi-modal state-space, or n-dimensional surface, a genetic algorithm may offer significant benefits over more typical search of optimization techniques (Linear programming, heuristic, depth-first, breath-first, and praxis) ("Genetic Algorithm," Online, 1996).

GAs simulates the survival of the fittest among individuals over consecutive generation for solving a problem. Each generation consists of a population of character strings that are analogous to the chromosome that we see in our DNA.

Each individual represents a point in a search space and a possible solution. The individuals in the population are then made to go through a process of evolution.

GAs are based on an analogy with the genetic structure and behaviour of chromosomes within a population of individuals using the following foundations ("Genetic Algorithm," Online, 1996):

- Individuals in a population compete for resources and mates.
- Those individuals most successful in each 'competition' will produce more offspring than those individuals that perform poorly.
- Genes from `good' individuals propagate throughout the population so that two good parents will sometimes produce offspring that are better than either parent.
- Thus each successive generation will become more suited to their environment.

After an initial population is randomly generated, the algorithm evolves through three operators:

1. selection which equates to survival of the fittest;
2. crossover which represents mating between individuals;
3. mutation which introduces random modifications.

Design of Focused Crawler Using Genetic Algorithm

To use genetic algorithm, design of the standard focused crawler needs to be modified to fit it with genetic algorithms' operations like selection, crossover and mutation. This section explains the propose work along with the design of the focus crawler.

a) Problem statement

It is important for any focused crawler to choose the most promising links in order and try to maximize the relevancy of new, unvisited URLs. Focused crawler should provide flexibility while calculating relevance of URLs by having built in artificial intelligence within the system. Machine learning techniques provide best results in this field. By applying the concepts of genetic algorithm for topic based focused crawling, more optimal and accurate results can be achieved.

b) Proposed Architecture of Focused Crawler

The main aim of proposed work is to choose the most promising links in order and try to maximize the relevancy of a new, unvisited URL by applying the concepts of genetic algorithm for focused crawling. This algorithm will try to produce more optimal result, and it will also help to improve the accuracy. The proposed method yields promising results. Fitness function and similarity approach need to be adjusted in such a way that it gives us maximum precision.

Our system consists of mainly 4 modules. Web pages are downloaded and their content is extracted. The extracted text and URLs are then fed to a classifier which determines if a page is relevant or not using genetic algorithm. If it is considered relevant, all the outlinks from that page is extracted and the crawling is continued. Figure 3 demonstrates the architecture. We are planning to use crawler4j library, which is a Java based open source web crawler as the baseline crawler and custom modules will be added to make it a focused crawler. In upcoming subsections we briefly discuss the functionality of these modules.

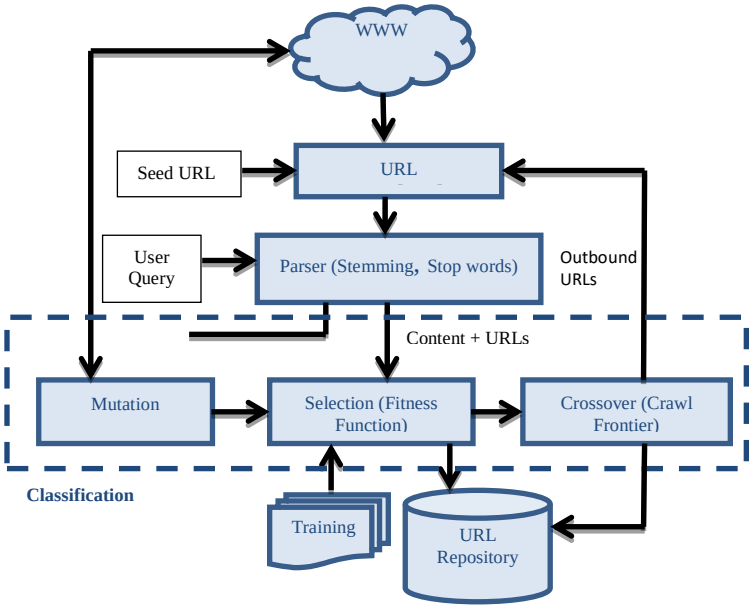

Figure 2: Architecture of Proposed Focused Crawler

URL Downloader: Downloads html pages from the web. Initially, the downloading is started with the user provided seed URLs and later refers the Crawler Frontier to get URLs for crawling. This can be implemented as a multi threaded application, which will make web page downloading faster. URL Downloader should also be responsible for enforcing certain ethics that prevents sending requests to the same server too frequently, thus, it will have some wait between 2 requests. It should also respects the Robots Exclusion Protocol which allows site owners to give specific instructions to web crawlers whether the site is crawlable or only some parts.

Parser: Web pages are noisy as they contain banners, advertisements, and other sources of unwanted entities. So, extracting useful textual content out of them is challenging. Parser extracts the textual content and URLs out of web pages. Parser can deal with typos in html, inconsistent html tags, comments and variety of other html errors. Parser ignores the content inside SCRIPT and STYLE tags in html pages, which do not contain any rich information useful for the classifier. Html META tags contain useful information about a particular page; we consider META title tag and the META description tag. This will also have class to perform stemming and stop words techniques on user's query to get refined user query.

Classification: Classification phase is a very important stage in the system. We have chosen Genetic Algorithm as classifier, which applies Selection, Mutation and Crossover techniques. It uses training data set in classifier, which is used to determine similarity of a new instance. Fitness function used to compare similarity is explained in next section. It updates final score of URL in database.

Crawl Frontier: It is the data structure which contains the URLs to be crawled and is implemented as a priority queue. All the URLs of a page will be added to the Crawl Frontier if that page is considered relevant at the classification stage. URL is assigned a relevance score which is the same score of its parent page (page where the URL is extracted) as assigned by the classifier. So, the URLs extracted from highly relevant pages are given priority and crawled first. Crawl Frontier maintains a database of crawled URLs to make sure that same URL is not crawled twice.

c) Process Flow and Genetic Algorithm Implementation Plan

Crawling flow will be as shown below:

Step 1- URL frontier is initialized with seed URLs and also placed in relevant links log.

Step 2- Communicate the User Query. Perform Stop words removal technique and stemming technique on User Query to get refined User Query.

Step 3- Process URL in URL frontier one by one to apply GA for optimized crawling result.

Step 4- Download page for URL.
Step 5- Compute score of page/URL by performing link and content analysis on page.
Step 6- Take out links from processed page and process all taken links for finding score of each link.
Step 7- Perform mutation.
Step 8- Select only few top score links from processed pages and send it to frontier.
Step 9- If selected link is not earlier visited then do crossover.
Step 10- Send the page to resultant repository.
Step 11- Repeat process for all selected pages recursively.
Step 12- Get the output.
Flowchart:

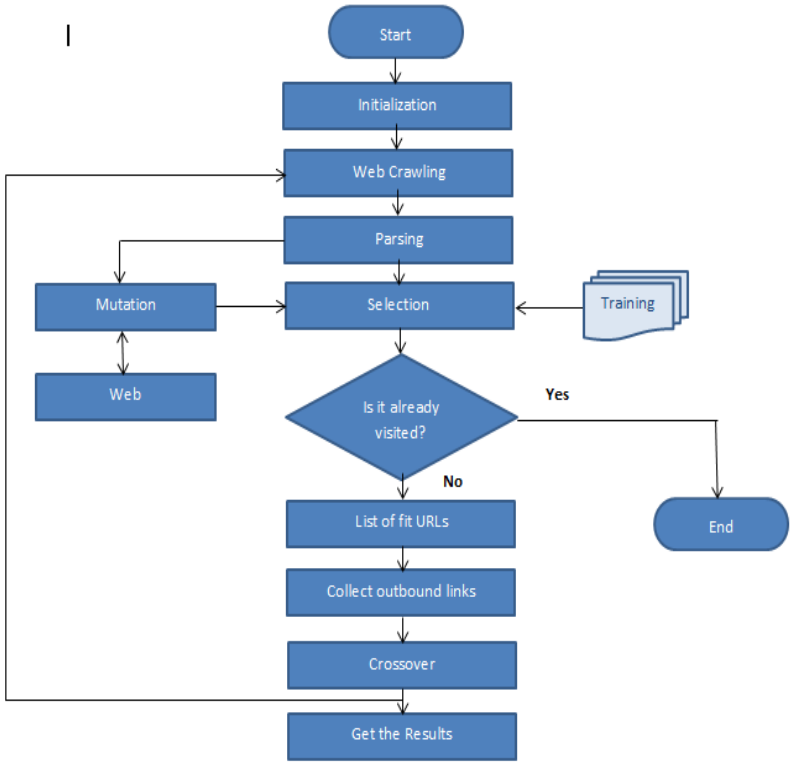

Figure 3: Process Flowchart

The brief description about stop word removal, stemming, special symbol removal, basic crawling, selection, mutation, crossover is considered as follows.
Remove special symbol, stop words, stem words: Special symbol removal algorithm will remove all the special symbol that are (. @ % & + - / # $! *, etc). Stop word algorithm will remove all the stop words. E.g.-the, are, is, about, all, by etc. Stem words algorithm will do stemming of all the words e.g.-summing, interested, like ing, est, ed, these words are removed. By using these three algorithms we get the re-fined query, that refine query is consist of keywords. By using these algorithms we can focus on only original refined keywords.
Web Crawler: The process of web crawling first it will start with the seed URL, then crawler will start downloading a group of seed pages, Parse through the download page and extract all the links. The links to pages that have positioned in a queue. Crawlers are designed for different purposes In the High performance crawlers, their goal is to improve the working of the crawler by downloading as a number of documents as feasible as in a definite time. By using this process the easiest algorithms is the Breadth First Search (BFS) Algorithm. This algorithm is the standardized search from one side to other side of the neighbor nodes. It starts by the source node and find all the neighbor nodes at the parity. If the goal is

found, then it is reported as the success and the search is ended. If the goal is not found, then it will go down to the next level far reaching the search from corner to corner the next neighbor nodes at that stage and so on till the objective is not found. When all nodes are searched, but objective is not found then it is reported as the failure. Breadth first is well performed when the objective is found on the upper level in a deeper tree. This strategy gives us more relevant result.

Genetic Algorithm:

Genetic algorithm is an iterative procedure which represents its applicant result as sequence of the genetic material called as the Chromosomes. When the folks come together, population form. Population is customized in the every iteration. Genetic Algorithm's process are continuously repeated are called the generation.

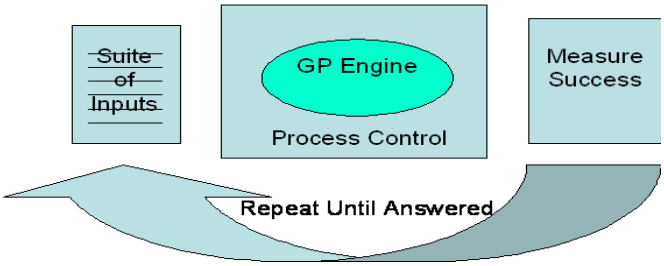

Figure 4: Typical Genetic Algorithm Processing

Genetic Algorithm used the genetic operative such as selection, crossover and mutation. It produces the solutions for the consecutive generations. It also used to optimize the Web crawling and it selects a more proper Web pages to be extracted by the web crawler. Genetic algorithm is used to calculate the relevancy of page by using fitness function. Link is extracted on the basis of fitness function. For fitness is calculated by using jaccard function. Operations are explained below:

Selection: Selection process finds out the similarity on web page on the basis of links and keywords. In selection process the formula for finding the score of links by using the jaccard function.

Jaccard function similarity can be calculated as

$$\text{Jsimilarity}(P,R)=(X\cap Y)/(X \cup Y) \quad (1)$$

P is a one web page and R is another web page. In web page P it contains a set of links called X and in web page R it contains a set of links called Y. Page P is constant but page R is vary. By using this formula it find out the score of the link by finding the similarity between the two pages on the basis of links.

In selection process formula to find the weight of keywords and data,

$$Dpr=\sqrt{(Mp+Nr+ CWpr)} / UWp \quad (2)$$

Dpr is the weighted term, p is a one web page and r is another web page. Mp is how much time keyword appeared in web page p. Nr is how much time keyword appeared in web page r. CWpr is common words in both web pages (p,r). UWpr is uncommon words in both web pages (p,r). By using these formulas we have to find out similarity between the two web pages on the basis of keywords.

Finally, for all the web page that has been visited by a web crawler, we have to make the addition for link and keyword score.

$$J(P,R) = \text{Jsimilarity (Jlink)} + \text{Dpr (Jkeyword)} \quad (3)$$

Crossover (Reproduction): Crossover generally combines two higher fitness value chromosomes, to gain a new offspring. These offspring are passed to the next iteration for further evaluations. After selecting the fittest individuals, we will perform the crossover operator to produce the children of the next generation. In crossover phase it selects the main link's sub-links after the selection process.

Mutation: Mutation phase provides the crawler capability to search widely various network area properly. Keywords are taken out after applying algorithm that has removed special symbol, stop words, stem word in parsing phase. The chosen keywords run like a query in the famous search engines, to get the links from the well-known search engine and pass to the selection phase.

d) Design Details & Implementation Plan

Java platform (JDK 1.8) will be used for the implementation of focussed crawler. RDBMS Postgres 9.3 will be used to store training data as well as testing data. Open source libraries like jsoup and Apache Lucene will be used for URL parsing and NLP purposes.

Pre-Requisites:

System Requirements:

- JDK 1.8
- Java open source libraries:
  - jsoup (Ver 1.10.1) – For URL download and parsing
  - Apache Lucene (Ver 6.3.0) – For NLP (Stemming, Stop words & Tokenization etc.)
  - Java-string-similarity – For jcard index calculations
- RDBMS Postgres 9.3 – For Training & Test databases

Design details of key modules are as follows:

URL Downloader & Parser:

jsoup library will be used to fetch & parse URL contents based on seed URL. jsoup is a Java library for working with real-world HTML. It provides a very convenient API for extracting and manipulating data, using the best of DOM, CSS, and jquery-like methods.

jsoup implements the WHATWG HTML5 specification, and parses HTML to the same DOM as modern browsers do.

- Scrape and parse HTML from a URL, file, or string
- Find and extract data, using DOM traversal or CSS selectors
- Manipulate the HTML elements, attributes, and text
- Clean user-submitted content against a safe white-list, to prevent XSS attacks
- Output tidy HTML

jsoup is designed to deal with all varieties of HTML found in the wild; from pristine and validating, to invalid tag-soup; jsoup will create a sensible parse tree.

*Example***:** To fetch the Wikipedia homepage, we can parse it to a DOM, and select the headlines from the "In the news section" into a list of Elements (online sample) as shown below:

```
Document doc = Jsoup.connect("http://en.wikipedia.org/").get();
Elements newsHeadlines = doc.select("#mp-itn b a");
```

Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform. It has classes for tokenization, stop word filtering, stemming and similarity. Here's an example using Lucene to remove stop words and stem an input string:

*Example***:**

```
public static String removeStopWordsAndStem(String input) throws IOException {
   Set<String> stopWords = new HashSet<String>();
   stopWords.add("a");
   stopWords.add("I");
   stopWords.add("the");

   TokenStream tokenStream = new StandardTokenizer(
         Version.LUCENE_30, new StringReader(input));
   tokenStream = new StopFilter(true, tokenStream, stopWords);
   tokenStream = new PorterStemFilter(tokenStream);

   StringBuilder sb = new StringBuilder();
   TermAttribute termAttr = tokenStream.getAttribute(TermAttribute.class);
   while (tokenStream.incrementToken()) {
      if (sb.length() > 0) {
```

```
            sb.append(" ");
        }
        sb.append(termAttr.term());
    }
    return sb.toString();
}
```

Which if used on your strings like this:

```
public static void main(String[] args) throws IOException {
    String one = "I decided buy something from the shop.";
    String two = "Nevertheless I decidedly bought something from a shop.";
    System.out.println(removeStopWordsAndStem(one));
    System.out.println(removeStopWordsAndStem(two));
}
```

Yields this output:

```
decid bui someth from shop
Nevertheless decidedli bought someth from shop
```

Genetic Algorithm Implementation:

Outline of the Basic Genetic Algorithm:

1. ***[Start]*** Generate random population of n chromosomes (suitable solutions for the problem)
2. ***[Fitness]*** Evaluate the fitness f(x) of each chromosome x in the population
3. ***[New population]*** Create a new population by repeating following steps until the new population is complete

a. ***[Selection]*** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)

b. ***[Crossover]*** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.

c. ***[Mutation]*** With a mutation probability mutate new offspring at each locus (position in chromosome).

d. ***[Accepting]*** Place new offspring in a new population

4. ***[Replace]*** Use new generated population for a further run of algorithm
5. ***[Test]*** If the end condition is satisfied, stop, and return the best solution in current population
6. ***[Loop]*** Go to step 2

Genetic Algorithm Pseudocode:

```
Algorithm GA is
  // start with an initial time
  t := 0;

 // initialize a usually random population of individuals
  initpopulation P (t);

  // evaluate fitness of all initial individuals of population
  evaluate P (t);

  // test for termination criterion (time, fitness, etc.)
while not done do

        // increase the time counter
        t := t + 1;

        // select a sub-population for offspring production
```

```
        P' := selectparents P (t);

        // recombine the "genes" of selected parents
        recombine P' (t);

        // perturb the mated population stochastically
        mutate P' (t);

        // evaluate it's new fitness
        evaluate P' (t);

        // select the survivors from actual fitness
        P := survive P,P' (t);
        od
    end GA
```

Jaccard Similarity Library:

"Java-string-similarity" library (tdebatty) will be used to implement jcard index to compare calculated weighted links and keyword in selection process of genetic algorithm.

Conclusion

This paper described genetic algorithm and its application in focused crawler. Paper is detailing out the architecture and design strategy for the focused crawler using genetic algorithm. It covers UML class diagrams and sequence diagrams which will be used in implementation.

It suggests the methodology to calculate fitness of each individual. It also guides about further work which is required to complete Project stage-2 i.e. implementation and experimental setup required to be done to complete the project.

REFERENCES

Bhatia K, Gupta JP, Gupta P and Sharma A (2009). A Novel Framework for Context Based Distributed Focused Crawler (CBDFC). *International Journal of Computer Science and Information Technologies* **1**(1).

Brin S and Page L (1998). The Anatomy of a Large- Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems* **30** 107–117.

Chakrabarti S (2012). *Mining the Web: Discovering Knowledge from Hypertext Data*, (USA, Boston: Elsevier).

Chauhan N and Kumar S (2012). A Context Model For Focused Web Search. *International Journal of Computers & Technology* **2**(3).

Dahiwale P, Malik L and Raghuwanshi MM (2015). Design and Implementation of Focused Web Crawler Using Genetic Algorithm: An Approach to Web Mining. *International Journal of Scientific & Engineering Research* **6**(6).

Deepa SN and Sivanandam SN (2008). *Introduction to Genetic Algorithms*, (*Springer*-Verlag Berlin Heidelberg, Germany).

Genetic Algorithm [Online] (1996). Available: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html.

Jayaratne L and Samarawickrama S (2013). Focused web crawling using named entity recognition for narrow domains. *International Journal of Research in Engineering and Technology* **2**(3).

Keole RR and Pardakhe NV (2013). Analysis of Various Web Page Ranking Algorithms in Web Structure Mining. *International Journal of Advanced Research in Computer and Communication Engineering* **2**(12).

Mukhopadhyay DM, Balitanas MO, Farkhod AA, Joen S-H and Bhattacharyya D (2009). Genetic Algorithm: A Tutorial Review. *International Journal of Grid and Distributed Computing* **2**(3).

Patni R, Patani V, Shah R and Shah V (2014). Understanding Focused Crawler. *International Journal of Computer Science and Information Technologies* **5**(5).
Selvakumar M and Vijaya A (2014). Design and Development of a Domain Specific Focused Crawler Using Support Vector Learning Strategy. *International Journal of Innovative Research in Computer and Communication Engineering* **2**(5).