CHAPTER-3

PROPOSED ARCHITECTURE FOR CONTEXT BASED
FOCUSED SEARCH ENGINE

Focused search engine is the information retrieval tool that narrows the search space by finding the documents relevant to user query only. The objective of a focused search engine is to return more specific results in response to a user query rather than large number of results.

In this chapter, the architecture for such a focused search engine has been proposed that works on the basis of the context of web pages and serve the user with contextually more related web pages.

3.1 INTRODUCTION

User searches for information from WWW, related to his topic of interest by using various information retrieval tools. The most popular and frequently used tool is a search engine. The major responsibility of a search engine is to maintain a repository of downloaded web pages and respond to the user search query with a list of matched web pages.

Currently there are approximately 50 billion web documents [12] on the Web, stored on thousands of web server worldwide. There may be large number of related documents from user’s query point of view, few more and few less relevant. A normal search engine returns thousands of matched documents in response to a query. The size of information provided by such a search engine is too large to go through and thus, leads to the problem of information overkill. This problem further aggravates in the case of inexperienced users, trying to search the information from the web. In line with the famous “8 second rule”, such users look at first few results and tend to turn away. This behaviour motivates the need to develop tools with high precision that can provide top ‘k’ relevant links to web documents, rendering search engines and related
technologies a potential area of research. Thus, the challenging task for a search engine is to find the relevant documents related to user search query.

The solution for it has come out in the form of focused search engine. Focused search engine is an information retrieval tool that retrieves the relevant web pages w.r.t. the user query. Moreover, it concentrates more on the quality or relevancy of the web pages rather on quantity.

In this chapter, a novel architecture for context based focused search engine is designed, which is able to display topic specific documents from the Web, catering to the need of a user to search highly relevant documents.

This chapter is organized as follows: Section 3.2 gives the brief introduction about the architecture of a general search engine. It covers various components and their overall functionalities. Section 3.3 highlights the gaps in the user requirement as well as the search results and presents the motivation to improve the basic architecture. The proposed modified architecture along with detailed working of its various components is discussed in section 3.4. It introduces the concept of contextual senses and their extraction and storage. To store the contextual senses, the context based index structure is also discussed. Section 3.5 describes the data flow in the various components of the proposed architecture. Section 3.6 concludes the chapter.

3.2 ARCHITECTURE OF A GENERAL SEARCH ENGINE

This section provides a brief review of general search engine architecture. A search engine generally comprises of the following components:

- User Interface
- Query Processor
- Index Builder
- Ranking Module
- Crawler
The detailed architecture of a search engine has already been discussed in chapter-2 (refer section 2.2.3). The architecture is composed of two layers back-end process and front-end process. Back-end process is composed of crawler, web graph builder, page rank calculator and index builder. The front-end process is composed of user interface, query processor and ranking module. The major responsibility of back-end process is to maintain the repository of downloaded web documents and build an inverted index for fast retrieval from the repository. The major responsibility of front-end process is to accept the user query and provide a list of matched web documents in descending order of their computed rank.

3.3 MOTIVATION FOR IMPROVEMENT

In order to search the information using search engine, generally users enter query keywords without giving the context. A single query keyword has various meanings depending upon its usage called contextual senses. This property of words having different meaning in different contexts is called polysemy. General search engine architecture does not filter and rank documents based on polysemy context. Thereafter, the search engine searches and produces the results without considering the user’s search context and as a result returns a large number of documents which may or may not fulfil their requirements. For example, for a query keyword ‘Server’, a general search engine responds with top results related to the computer server such as Server (Computing), File Server, Virtual Server, Server (Ubuntu) and Server Research Centre etc. Whereas, word ‘Server’ can also refer to a waiter, court games, a computer science host or a utensil.

General search engine does not provide an interface to capture the exact or actual context of query keywords. Information needs of a user thus cannot be decided purely on basis of query keywords. Therefore, improvement in existing architecture is required to get the specific context of user’s search and that of the web page.

Moreover according to the user search trends (presented in chapter-2 section 2.3), only 40% of the users go up-to 3rd page of displayed results. The major group of users either change the query, the search strategy or the information retrieval tool. It shows
that users are more interested to get relevant results either in top 10 or 20 positions on first page. Before discussing the proposed improvements, next subsection introduces the concept of Contextual sense, its scope in context of search engine and some motivating examples to show the need for improvement.

### 3.3.1 CONTEXTUAL SENSE: MEANING, SCOPE AND EXAMPLES

Generally, query keywords passed by users have multiple meanings (as Server); as a result the information needs of user remain ambiguous and cannot be clearly deduced from query keywords. This leads to many undesirable results in top positions. For example, in English ‘mouse’ is a pointing device in computing and rodent elsewhere. Current search engines cannot deduce the context on the basis of only query keywords, as a result, for a query keyword ‘mouse’ they respond with a list of web pages containing documents related to mouse in sense of pointing device as well as rodents. It may happen that sometime desired contents are not found even in top few results. Thus it’s a challenging task to make search engine identify different contextual senses of a word. The proposed context based focused search engine architecture has introduced a new module called contextual sense extractor (CSE) to remove ambiguity in query keywords due to polysemy property.

#### 3.3.1.1 Scope

The word ‘contextual sense’ has been given different definitions and scopes by different researchers. Context is referred to as information about location, identities of people and objects in close proximity of user and the changes in them [85]. Brown et al [86] has added the time of day, season or temperature etc. to location and identities to get the context. Another researcher [87] has added user’s emotional state, focus of attention to location and nearby identities, to consider as context. Dey and Aowd [88] has defined the context as any information that can be used to characterize the situation of a user. Anders and Marius [89] have considered the set of environmental states and its settings for the user, to get the context in user dependent applications and services such as information retrieval. In the area of information retrieval, the context is based on user’s recent activities such as visited pages and emails, previous
search queries etc. [65]. Some have considered user’s implicit behaviour with some metadata information [66] to identify the current search need. Another piece of work has provided the idea of automatic construction of user profile [67]. Work done in [69, 70] has tried to remove the ambiguity using query logs, click through history within a session, user profile etc.

Determining the context on the basis of user profile is not a good choice as user interest is not always related to the same topical area. Searching based on user profile may sometime leads to irrelevant document. For example, if a scientist provides ‘Mouse’ as a keyword, its profile based search will give preferences to documents related to ‘Mouse’ as ‘rodent’ assuming some research has been conducted on rodents. Whereas, scientist may be looking for ‘Mouse’ as a Computer Device in order to use computer to do some research work.

Therefore, in the proposed improvements we have considered the various polysemous meanings of a word as basis for different contextual senses. We determine the contextual sense of web pages based on polysemy meaning of query keywords.

3.3.1.2 Motivating Examples

Google is the most popular search engine which indexes maximum number of publically indexable documents from the web among all other available search engines [refer section 2.1.2]. It maintains the word lexicon and returns synonym words pair in alphabetical order in response to a given query keyword [31]. For instance, in response to the query keyword “Student”, the Google search engine returns a list of synonyms words pair as shown in Figure 3.1.
The results that Google returns in response to various query keywords are generally the synonyms word pairs and do not show the contextual senses of the keywords. It does not show the polysemy based contextual senses of the keywords. Similarly, various words that are having multiple meanings in different contexts when passed to Google, provides only the synonyms pair. Few such examples of words along with their different meanings and synonyms word pairs provided by Google are listed in Table 3.1. For instance, for a query word ‘Bat’ in row ‘2’ of table 3.1 Google provides options ‘Bata, Batman, Bata India and Batman Games’; whereas, the multiple meanings of ‘Bat’ are ‘a kind of mammal and cricket bat’. None of the different meanings of ‘Bat’ is covered by Google.

Table 3.1: Motivating Examples

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Word</th>
<th>Contextual Meaning not generally covered by Search Engines</th>
<th>Options provided by Google</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bark</td>
<td>Sailing ships, barking of a dog, part of tree trunk</td>
<td>Barkha Dutt, Barkatullah University, Barkha Bisht, Barka</td>
</tr>
<tr>
<td>2</td>
<td>Bat</td>
<td>Mouse like mammal, cricket bat</td>
<td>Bata, Batman, Bata India, Batman Games</td>
</tr>
<tr>
<td>3</td>
<td>Wood</td>
<td>United states actor, wood as jungle or forest, kind of golf club</td>
<td>Woodland shoe, Woodland sale, Woodstock school</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>4</td>
<td>Sail</td>
<td>Sheet used for sailing boats, an ocean trip,</td>
<td>Sail admit card, Sail recruitment</td>
</tr>
<tr>
<td>5</td>
<td>Pound</td>
<td>Currency, unit of weight measure</td>
<td>Pound to Rupee, Sign, Kg</td>
</tr>
<tr>
<td>6</td>
<td>Court</td>
<td>Judicial assembly as court, game area is also called court, courtyard near house, Margaret court player(name)</td>
<td>Court.nic.in, Court Marriage, Courtesy Meaning, Courteney cox</td>
</tr>
<tr>
<td>7</td>
<td>Crane</td>
<td>Mechanical object used to lift heavy things, crane bird, united state writer and poet names</td>
<td>Crane, Crane games, Cranex Ltd., Cranes software</td>
</tr>
<tr>
<td>8</td>
<td>Trunk</td>
<td>Elephant trunk, tree trunk, luggage carrying object</td>
<td>Trunking, trunk call, Trunk meaning (synonyms)</td>
</tr>
<tr>
<td>9</td>
<td>Lion</td>
<td>Lion animal, zodiac sign</td>
<td>Lion, Lionel Messi, Lion King, Lion vs. Tiger</td>
</tr>
<tr>
<td>10</td>
<td>Bear</td>
<td>Mammal animal, investor in share market</td>
<td>Bear, Bear Grylls, Beard, Bearing</td>
</tr>
<tr>
<td>11</td>
<td>Present</td>
<td>Time tense, gift</td>
<td>Presentation Skill, Present Perfect Tense, Present indefinite Tense</td>
</tr>
<tr>
<td>12</td>
<td>Desert</td>
<td>Dry land, abandon or leave someone in need</td>
<td>Desert Cooler, deserted meaning, desert meaning in Hindi</td>
</tr>
<tr>
<td>13</td>
<td>Pupil</td>
<td>Iris of eye, student</td>
<td>Pupil, Pupil dilate, papillary reflex</td>
</tr>
<tr>
<td>14</td>
<td>Table</td>
<td>Furniture, Table Array, Multiple</td>
<td>Tablet price in India, table no 21, Table no 21 Songs</td>
</tr>
<tr>
<td>15</td>
<td>Play</td>
<td>Drama, game playing</td>
<td>Play store, play temple run</td>
</tr>
<tr>
<td>16</td>
<td>Light</td>
<td>Weight, sunlight</td>
<td>Light box, light box jquery</td>
</tr>
<tr>
<td>17</td>
<td>Lock</td>
<td>Charge of a gun, lock of DOOR, wrestling hold, cluster of hair</td>
<td>Lock out, Lockheed Martin, Locked Out of Heaven</td>
</tr>
<tr>
<td>18</td>
<td>Band</td>
<td>Color band, frequency band, hair band, ring</td>
<td>Bandit queen, bandwidth, Band baja barat</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>--------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>19</td>
<td>Jam</td>
<td>Fruit jam, traffic jam, electronic jam</td>
<td>Jam Millia, Jamia Hamdard, Jam, J&amp;K, Jam Shipra</td>
</tr>
<tr>
<td>20</td>
<td>Lead</td>
<td>As metal, position in game</td>
<td>Leadership, Lead, Leadership qualities, Leadership styles</td>
</tr>
<tr>
<td>21</td>
<td>Rap</td>
<td>Hip-hop, knock</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>May</td>
<td>Month, whitethorn flower</td>
<td>Mayawati, Maybelline, Mayo College</td>
</tr>
<tr>
<td>23</td>
<td>School</td>
<td>Institution, large group of fish</td>
<td>School, school of open learning</td>
</tr>
<tr>
<td>24</td>
<td>Spade</td>
<td>Game of cards, hand shovel, offensive name for black person</td>
<td>Spademark, spade work</td>
</tr>
<tr>
<td>25</td>
<td>Bass</td>
<td>DEEP sound, type of fish</td>
<td>Bass in India, Bass booster</td>
</tr>
<tr>
<td>26</td>
<td>Buffet</td>
<td>Self-serve food, hit or punch</td>
<td>Buffet dinner in Noida, Lunch in Noida, lunch in Delhi</td>
</tr>
<tr>
<td>27</td>
<td>Navy</td>
<td>Colour, fleet of ships</td>
<td>Navy, Navya, Navyug</td>
</tr>
<tr>
<td>28</td>
<td>Flight</td>
<td>Air travelling, flock of birds</td>
<td>Booking, status, tickets</td>
</tr>
<tr>
<td>29</td>
<td>Moped</td>
<td>Motorbike, to be gloomy, sad</td>
<td>Moped prices, Mopeds in India</td>
</tr>
<tr>
<td>30</td>
<td>Dough</td>
<td>Mixture of flour, money</td>
<td>Doughnut, dough maker, doughnut recipe</td>
</tr>
<tr>
<td>31</td>
<td>Ham</td>
<td>Meat, radio operator, actor</td>
<td>Hamdard, Hamirpur, Hampi, Hamburg Sud</td>
</tr>
<tr>
<td>32</td>
<td>Port</td>
<td>Comp. Sc. Port, Airport, port wine, opening for firing through</td>
<td>Portal bsnl, port blair, portfolio</td>
</tr>
<tr>
<td>33</td>
<td>Bank</td>
<td>Bank financial, river etc.</td>
<td>Bank of Baroda, Bank of India, Bank Jobs, Banker Adda</td>
</tr>
<tr>
<td>34</td>
<td>Buffet</td>
<td>Furniture, meal</td>
<td>lunch, dinner</td>
</tr>
<tr>
<td>35</td>
<td>Colt</td>
<td>Horse, revolver</td>
<td>Colt Tech, pistol</td>
</tr>
<tr>
<td>36</td>
<td>Java</td>
<td>Java Lang, Island, coffee</td>
<td>Java Download, JavaScript, java games</td>
</tr>
<tr>
<td>37</td>
<td>Mouse</td>
<td>Rodent, computer device,</td>
<td>Mouse Breaker, Mouser, Mouse</td>
</tr>
</tbody>
</table>
Moreover, results of Google listing change according to the last browsing history of the searches that are carried out. One such scenario for query keywords ‘Student’ is shown in figure 3.1 (above) and figure 3.2. It can be seen that the result listing for ‘Student’ has been changed in figure 3.1 from options ‘Student Loan’, ‘Student portal’ to ‘Student of the year’ (Song, movie) after the release of the movie ‘Student of the year’ in Figure 3.2.

![Figure 3.2: Changed Result from Google for keyword ‘Student’](image)

It is clear from these motivating scenarios that large number of words are having different context as per their usage, but a search engine interface does not cover all of them. The search is carried out without considering user specific interest for such queries. Thus, users get some undesired results. Hence, there is a need to modify
search engine basic architecture that gives search results after resolving the Polysemy based contextual senses of query keywords.

3.3.2 IMPROVEMENT IN EXISTING ARCHITECTURE

Thus the basic architecture of a search engine needs to be improved to determine and resolve the Polysemy based context of the query keywords and retrieve more relevant documents at higher positions. In order to achieve the same, some changes are proposed in the general architecture of search engine. Modifications are proposed in the query processor, ranking module and user interface of the front-end process layer. The back-end process layer has been divided into two sub-layers: middle layer and bottom layer. No modifications are proposed for the bottom sub layer whereas the changes have been done in index builder module and one extra module named back-link extractor has been added at middle sub layer. The detailed proposed modified architecture of context based focused search engine is discussed in the following section.

3.4 PROPOSED MODIFIED ARCHITECTURE OF SEARCH ENGINE

The proposed modified architecture of search engine is shown in Figure 3.3. It is referred as Context Based Focused Search Engine. It works on different contextual interpretation of the keywords, entered by the user in form of query and the words present in the web documents. The architecture consists of the three layers:

- Top Layer
- Middle Layer
- Bottom Layer
Figure 3.3: High level architecture of proposed context based focused search engine
Detailed working of various components of different layers is explained in the following sections.

3.4.1 THE TOP LAYER

The top layer is composed of user interface and query processor at broader level. As compared to the architecture of a general search engine, we have modified the query processor which is now composed of contextual sense extractor, search module, context based relevance calculator and ranking module. Contextual sense extractor and Context based relevance calculator are two new modules added in the proposed improvements. Contextual sense extractor deal with various meanings/contexts of query keywords and words present in web pages. Context based relevance calculator judge the relevance of web documents in respect to various contextual senses of words and assign context score. In the proposed architecture, ranking module makes use of this context score as a ranking heuristic.

Processing at the query interface operates in two modes one for end user who makes use of search engine as an information retrieval tool. The second mode is for the administrator who built up the back end repository for search. The details of two modes are as follows:

3.4.1.1 End User Mode

The end user mode elaborates the process of resolving the user query with list of relevant documents. The query processor in this mode starts with the user query. It searches the indexed database pre-processed using various contextual senses of words and allows the user to choose the context of the search topic from the various contextual senses of query keyword. Thereafter, based on the selected context, its ranking sub module orders the matched results based on context scores and present more relevant web documents to the user. Thus, in end user mode only three sub modules of query processor i.e. the user interface, search module and ranking module of query processing are used. Some modifications have also been done in the user
interface to display various contexts of the query and also allow the user to choose one specific context. The brief working of user interface is as follows:

3.4.1.1 User Interface

The basic component of top layer is user interface. Following activities are been performed in this interface:

1. It acts as an interface between user and the search system.
2. It accepts the user's query keywords from the console. The query keywords are then passed to the query processor.
3. It displays various contextual senses of the query keyword returned by the query processor.
4. It allows the user to select one specific context of interest and pass this information to the query processor.
5. It finally displays the search results retrieved by query processor in descending order of computed context score in selected context.

3.4.1.2 Administrator Mode

The administrator mode elaborates the process of repository construction. For this, the query processor gets the list of downloaded URLs from repository and passed them to contextual sense extractor (CSE). The CSE sub module gets the various contextual senses of top ‘20’ highest weighted words present in web documents from WordNet dictionary. The context based relevance calculator computes the context score for each downloaded URL in respect to various extracted contextual senses of top ‘20’ highest weighted words by CSE sub module. Thereafter, update the indexed database for each URL entry consisting of computed highest context score and its corresponding contextual sense. The detailed working of contextual sense extractor and context based relevance calculator is discussed in chapter-4. The ranking module uses this updated indexed database to return the list of web pages in order of their context score, in response to a specific query context. Chapter-5 elaborates the detailed working of ranking module.
Therefore, in the proposed architecture the major work or modification has been done at query processor.

### 3.4.2 THE MIDDLE LAYER

The Middle layer indexes the local database and enriches the database for different topics and contexts. It consists of following two components:

- Indexer
- Back-link extractor

#### 3.4.2.1 Indexer

The job of the indexer is to receive the web documents from the local database and to create the final repository which is further used by the query processor to produce the result to the user against a given query.

There are many models available for the indexing such as: inverted index model, PAT array model, signature file model etc. The inverted file index is a widespread popular indexing technique. This is basically a collection of lists, one per term, recording the identifiers of the documents containing that term.

Navarro et al. [90] has discussed the construction and inquiry of inverted model. Gonnet et al. [91] has conversed that pat array or pat trees are suitable for more complex inquiry. In 1980’s signature file model was very popular [92, 93] and now has been replaced by inverted file model. The retrieval using inverted index is done on the basis of document frequency and keyword frequency as Donald et al. [94] and Justin Zobel et al. [95] conversed. Some modification to pat array model has been proposed by Xunwen Liu et al. [96] to improve the efficiency of space and time. Charles et al. [97] & Ricardo et al. [98] has discussed indexing technology that is static in nature as they have considered that the changing period of documents is comparatively long. Sergey Brin et al. [29] discussed that Google use keyword-based
technology under the hyperlink data environment, the data structure used by the Google as Index.

Soumyadeb Mitra [99] has proposed a technique to update an inverted index based on merging of the posting lists of terms. Ndapandula Nakashole [100] has pointed to the issue of indexing the web documents in digital libraries so as to help the search engine to result with relevant documents much ahead of irrelevant documents in response to user query. Joseph Sack [101] has presented full text indexes that are compressed index structure comprised of tokens from the indexed textual data. Frederik Transier [102] has indicated that the most popular data structure for text search engines is the inverted index.

A critical review of above discussed existing indexing techniques indicates several drawbacks in indexing structures. Static Indexing consider that the indexed documents are seldom changed and hence not suitable for the documents over Internet that has frequent updates. Keyword based technology breaks through the restriction of enquiry topic. Inverted index keeps the information only about the each distinct term in the document.

Thus, none of the Index structure considers the context of the term in the index. Indexes are only based on the occurrence of the term in the document not related to their usage/sense behind its usage. The resultant set of documents is obtained after a query only matches the dictionary word (alphabets). So, none of the index structures for search engines have a provision to store contextual sense of query keyword.

In the proposed CBFSE architecture the modification in the index structure is done so that the contextual information of words can also be stored in the inverted index. We have proposed a context based index structure that stores the information related to the various contextual senses of the words with the list of pointers pointing to the relevant document in that contextual sense. The detail of the index structure is discussed in the following section.
3.4.2.1.1 Context Based Index Structure

The proposed data structure for storage of indexed keywords and their associated contexts is shown in the figure 3.4

![Context Based Index Structure](image)

**Figure 3.4: Context Based Index Structure**

At highest level there is an array of pointers, consisting of 26 locations and each location pertains to an alphabet ranging from ‘A’ to ‘Z’. Each pointer points to a binary search tree of nodes where each node starts with a specific alphabet. For example, consider the pointer corresponding to alphabet ‘M’, it is pointing to a binary search tree consisting of nodes starting with alphabet ‘M’ i.e. Mouse, Monitor, Mastermind etc. Similarly the pointer pertaining to alphabet ‘S’ points to a binary search tree consisting of nodes such as Student, Scholar, Skillet, Server, Sneak etc.
The detail of the node structure is presented in Figure 3.5. The structure of the node has a provision to store the list of various contextual senses of a keyword. For instance, the contextual senses $CS_1$, $CS_2$, $CS_i$ etc. are shown in the figure. For each contextual sense $CS_i$, a list of pointers to relevant documents in that contextual sense is also associated. For example, for contextual sense $CS_1$ the list of pointers pointing to documents $D_{11}$, $D_{12}$ etc. is associated.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Various Contextual Senses of keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$CS_1$</td>
</tr>
<tr>
<td></td>
<td>$CS_2$</td>
</tr>
<tr>
<td></td>
<td>$CS_i$</td>
</tr>
</tbody>
</table>

![Figure 3.5: Structure of the node](image)

Where,

Keyword: the word contained in one or more documents

Various Contextual senses of keyword: list of all different contextual senses of the keyword

$CS_i$: $i^{th}$ contextual sense

$D_{ij}$: $j^{th}$ document of the $i^{th}$ context

This structure is designed keeping in mind the quick response to the user query. As the query comes in the form of keyword, first the index is searched by the beginning letter of the keyword. After that, corresponding BST is searched for a match of the keyword, if found the list of contextual senses is displayed to the user. The user selects a specific contextual sense for its information need. The search progress again in the index structure only for the matched documents related to the user specified contextual sense of the keyword.

The index structure is updated for each new downloaded page after evaluated using CBR calculator.
3.4.2.2 Back-Link Extractor

There are two types of hyperlinks for web documents, in-links and out-links. The links pointed to a web page called its in-links or back-links (the in-links and back-links are used interchangeably). The links that are emerging from a web page to other pages are called out-links or forward-links. Generally, the forward links are considered by search engines as pages that could be displayed to the user in response to a query. Back-links are only considered for the purpose of analysing the web documents to which they point to. Guang [103] proposed a level-based link analysis that computes the rank of a web page by assigning weight to each hyperlink according to its level properties. S. Ganesh [104] recommended a link analysis Page Rank algorithm that works on back-link count and association metric to evaluate relevancy of a link in a web page. Chen Ding [105] has proposed a mechanism to locate referral parent (back-link) for a given fragment of a web document from the client side. All these existing techniques do not consider the back-link to be as candidate page displayed in response to a query. However, S. Chakarbari [72] found that if back-link information will be provided to web surfer; the process of information sighting will be much more effective. It does not differentiate between various incoming links i.e. all back-links are considered of equal importance.

Thus, requirement is to include the back-links in final results but after differentiating between the various back-links to a web page. For this purpose, in the proposed CBFSE architecture (shown in Figure 3.3) back-link extractor module is introduced that recursively finds the back-links of the URLs stored in the indexed repository and sends them to crawl-manager for further downloading, if they are not already downloaded. These extracted back-links are then judged using the CBR calculator and only the relevant back-links are included in final results. Thus, enriches the repository with more number of web links. The detail of back-link extraction and their selective inclusion is discussed in chapter 6.
3.4.3 THE BOTTOM LAYER

The bottom layer, with the help of crawlers continuously search the web servers and keep downloading the web pages, storing them in a local database called repository [106]. The proposed architecture has not introduced any change or improvements in the bottom layer of search engine architecture. It uses the index of downloaded web documents to get the list of URLs and back-links and then process them using CBR calculator mechanism and maintain its own pre-processed indexed database. So, no change or modification is done at bottom layer. Generally, the bottom layer consists of three major components

- Crawl Manager
- Crawl Worker
- Local database

3.4.3.1 Crawl Manager

Crawl manager is one of the core components in the bottom layer of the architecture. It works on the basis of seed URLs provided to it. Once the seed URLs are initialized, it distributes them to multiple crawl workers for downloading the web documents from the WWW.
The crawl manager is the sole component which manages the entire working of all the crawl workers. The crawl worker activities are shown in figure 3.6.

The Crawl Manager consists of 3 sub processes whose brief working is as follows:

**URL_Mapper**

This process resolves IP addresses for URLs sent to the crawl manager for downloading the respective documents.

**Downloader**

This process first downloads the robot.txt for the URL and later downloads the document, if allowed in the robot.txt [132].

**Crawl worker (CW):**

This component of bottom layer works on the server-side and is coordinated by crawl manager. It downloads the web documents for the list of URLs assigned by Crawl manager by visiting the different web servers under its range of search.

Figure 3.6: Client Side Crawl Worker Activities
Local Database

The web documents after downloading from the CW are passed to Crawl Manager. Crawl Manager inserts these documents in a local database which further is utilized by the indexer to create the final repository.

The algorithm for Crawl Worker is shown in Figure 3.7

```
Function Crawl_Worker() {
    while (true) {
        Wait for message (Download_doc);
        // Message is from User agent to download the doc
        Pass signal (Map_Url) to URL_Mapper;
        // To resolve the URL
        Wait for message (Request_serviced);
        // This Message is from URL_Mapper
        Pass signal (Download_doc) to downloader process
        // for downloading the document
        Wait for message (request_serviced);
        // This message is from downloader
    }
    signal (update_database) // Message to update the database
}
```

**Figure 3.7:** Algorithm for Crawl_Worker

The components used for crawling process with crawl worker are discussed in detail as below:

**URL_Mapper**

This process gets a URL_IP set as input from crawl worker. It examines URL_IP pair and if IP is blank, then the URL is sent to DNS Resolver process. After the URL has been resolved for its IP, it is stored in resolved_URL_Buffer. Its algorithm is given in Figure 3.8.
DNS Resolver

Generally, the documents are known by the domain names of their servers. The name of the server must be translated into an IP address before crawler communicates with the server. The internet offers a service that translates domain names to corresponding IP addresses. The software that does this job is called Domain Name System (DNS). The DNS resolver uses this service to resolve the DNS addresses for a URL and returns it back to the calling URL Mapper. It then updates the database for the resolved IP address of URL.

Downloader

It extracts the resolved URL from Resolved_URL_Buffer and downloads the corresponding document after knowing the permission in robot.txt [109]. The algorithm for working of Downloader is given in figure 3.9:

```
Function URL_Mapper ( )
{
    Wait for message (Map_URL) from crawl_worker
    // wait for message from crawl worker to resolve the URL_IP
    Extract the URL and its IP pair from URL_IP_buffer
    If (IP is blank)
    {
        Called DNS_Resoler ( )
        Wait for the resolved URL
    }
    Store the resolved URL in Resolved_URL_Buffer
    Pass the signal (Request_Serviced) to Crawl_Worker
}
```
Figure 3.9: Algorithm for Downloader

3.5 DATA FLOW IN VARIOUS COMPONENTS OF THE ARCHITECTURE

The data flow between the various components of the architecture (figure 3.10) describes the communication between the components. The working starts from the downloading the web documents and ends with displaying the relevant documents in respect to the user query.
Figure 3.10: Data Flow among various architectural components
The data flow is described in the following steps:

1. Crawl manager gets the list of seed URLs.
2. Distribute the list in multiple crawl workers those search the web servers and download the documents corresponding to the URLs assigned to them and return back to the crawl manager with the downloaded pages.
3. Crawl manager stores the downloaded web documents/pages in the local database.
4. Back-link extractor recursively finds the back-links of the URLs stored in the local database.
5. It updates the local database with the extracted back-link information.
6. It forwards these in-link URLs to the crawl manager for further downloading if page has not been already downloaded.
7. Indexer parsed the web documents stored in the local database and extracts the keywords present in them.
8. It then prepares an inverted index of the local database on the basis of extracted keywords present in the web documents.
9. The search module pass the seed word to extract the list of URLs and back-links from inverted index of downloaded documents.
10. The List of URLs and back-links are passed to the search module.
11. This list of URLs and back-links are further passed as input to contextual sense extractor and CBR calculator modules of query processor.
12. The contextual sense extractor extracts the different contextual senses of words present in each document corresponding to each URL.
13. The extracted contextual senses are passed to CBR calculator that computes the context score for each URL in respect to various contextual senses of words and stores them in processed indexed database.
14. The user interface gets the end user query from the console.
15. Query keyword is passed to the query processor.
16. Query processor passes the query word to Contextual Sense Extractor module.
17. The contextual sense extractor extracts the different contextual senses of query keyword.
18. The query processor displays these different contextual senses to the user to enable him to select a specific context of the search.
19. Query processor gets the user selection for a particular context.
20. The query processor passes the selected context to search module.
21. The search module further gives the selected contextual sense as input to the ranking module.
22. The evaluated computed context score of each URL is passed to the ranking module with corresponding contextual sense.
23. The ranking module search/filter the results in respect to the selected context of the user query and rank the matched results in descending order of their computed context score.
24. The ranked results are then passed to the user interface to display to the user screen.
25. Relevant results are displayed to the user with highly ranked documents at top positions.

**Example: The working of the proposed Search Engine**

Thus, in the proposed architecture the documents are stored in a repository and an inverted index is maintained. The query processor takes the user query and searches the inverted index to get the list of matched documents. The various contextual senses of the query keywords are displayed to the user and asked to specify the context of the search. Then the matched documents are ranked in that contextual senses and corresponding documents from the repository are displayed to the user as the final results.

For instance, query for keyword ‘Server’ is shown in Figure3.11. The inverted index is maintained in alphabetical order. The various contextual senses of server are then displayed to the user such as, Waiter, Host, Server (Utensil), Server (Court Game) etc. where, user has selected the sense as ‘host’ and then the corresponding relevant ranked documents i.e. D1, D5, D6, D11 will be displayed as the results of search.
3.6 CONCLUSION

The modified architecture for focused search is proposed that considers the various contextual senses of the words to identify exact need of the user. The search is focused, as only the web documents related to the user selected sense are displayed in the result list. The various contextual senses of words are extracted. The suitable data structure, context based index is designed to store the contextual senses. The context based relevance evaluation and ranking of web documents enable to display only the relevant document at higher positions i.e. earlier in the result list. How the relevance of a web document can be judged based on the extracted contextual senses is covered in the next chapter.