Design of an Agent Based Context Driven Focused Crawler

Naresh Chauhan[1] and **A.K. Sharma**[2]

Abstract - A focused crawler downloads web pages that are relevant to a user specified topic. Most of the existing focused crawlers are keyword driven and do not take into account the context associated with the keywords. This leads to retrieval of a large number of web pages irrespective of the fact whether they are logically related. Thus, the keyword based strategy alone is not sufficient for the design of a focused crawler as context relevance is more important as far as the user's requirement is concerned. This paper proposes the design of a context driven focused crawler (CDFC) that searches and downloads only highly related web pages, thereby reducing the network traffic. It also employs a category tree which is a flexible user interface showing the broad categories of the topics on the web. Since CDFC downloads only the relevant and credible documents, a very small number in comparison, the proposed design significantly reduces the storage space at the search engine side.

Index Terms - Search engine, Crawler, Hypertext Document System, Category Tree, Software Agents

1. INTRODUCTION

The World Wide Web (WWW) is a continuously expanding large collection of hypertext documents [1]. It represents a very large distributed hypertext system, involving hundreds of thousands of individual sites. It is a client-server based architecture that allows a user to initiate search by providing keywords to a search engine, which in turn collects and returns the required web pages from the Internet. Due to extremely large number of pages present on the web, the search engine depends upon crawlers for the collection of required pages. A Crawler [2] follows hyperlinks present in the documents to download and store web pages for the search engine.

Current commercial search engines maintain large number of web pages [3,7] and easily find several thousands of matches for an average query. Therefore, a search engine may present a list of thousands of web pages in response to user's particular keyword possibly consisting of irrelevant web pages also. The web search engines try to cover the whole web and serve queries concerning all possible topics [4]. In fact, from the user's point of view, it does not matter whether the search returned 10,000 or 50,000 hits because the number of matches becomes too large to sift, leading to the problem of *information overkill*.

[1]*Asst. Prof., Deptt. of Computer Engg., YMCA Institute of Engineering, Faridabad - 121006, India*
[2] *Professor & Head, Deptt. of Computer Engg., YMCA Institute of Engineering, Faridabad - 121006, India*
E-Mail: [1]nareshchauhan19@yahoo.com and 2ashokkale2@ rediffmail.com

The search quality of web pages can be improved by focused crawling [5,6,12] which aim to search and retrieve only the subset of the WWW that pertains to a specific topic of relevance. Focused crawler, therefore, offers a potential solution to the problem of information overkill. The existing focused crawlers [6,7] adopt different strategies for computing the words' frequency in the web documents. If higher frequency words match with the topic keyword, then the document is considered to be relevant. But the current crawlers are not able to analyze the context of the keyword in the web page before they download it. For instance, the word 'spider' has various interpretations. To a web programmer, it is the name of a software program used in search engines; to a general computer user it denotes a game of cards and to a layman it is simply name of an insect. Thus, the topical relevance is not the only issue for focused crawlers but context relevance should also be considered [10]. If the user issues one keyword then its relevant context must also be known.

In this paper, the design of a *Context Driven Focused Crawler* (CDFC) is being proposed that provides the context of the keywords to the user in a flexible and interactive category tree [5]. The agent-based design not only overcomes the complex time-consuming computations of existing focused crawlers but also reduces network traffic significantly. In line with the demands of a focused crawler that the relevant information should be collected and retrieved by the user in the least amount of time possible, the proposed architecture reduces the search time for a document and the information database on the search engine side becomes more easily manageable.

2. RELATED WORK

A similarity based crawler that orders URLs having target keyword in anchor text or URL, was probably one of the first efforts towards focused crawling [9]. The basic focus was to crawl more important pages first i.e. to look at various measures of importance for a page such as similarity to a driving query, number of pages pointing to this page (back links), page rank, location, etc. The Page Rank algorithm [11] computes a page's score by weighing each in-link to the page proportionally to the quality of the page containing the in-link. Thus, a web page will have a high page rank, if the page is linked from many other pages, and the scores will be even higher if these referring pages are also good pages, i.e. having high Page Rank scores. In the HITS (Hyper-link-induced- topic search) algorithm [8], an authority page is defined as a high quality page related to a particular topic or search query and a hub page is one that provides pointers to other authority pages. Based upon this, a web page is associated with an *Authority Score* and a Hub Score that is calculated to identify the web page context.

Another focused crawler [7] employs seed keywords which are used to find seed URLs from some standard search engine like

Google. The seed URLs are used to fetch seed pages with the help of TF.IDF algorithm, based on iteratively calculating word frequency. This algorithm is used to find out some more number of keywords from seed web pages to represent the topic. Afterwards, vector similarity is computed between web page and topic keywords to see whether the page is relevant to the topic.

Diligenti et al [6] uses a general search engine to get the web pages linking to a specific document and builds up a context graph for the page. The graph is then used to train a set of classifiers to assign documents to different categories based on their expected link distance to the target. In fact, graphs and classifiers are constructed for each seed document with layers being built up to a specified level. Thus, the crawler gains knowledge about topics that are directly or indirectly related to the target topic.

A critical look at the available focused crawlers [5-9,11] indicates that these crawlers suffer from the following drawbacks :

I. The problem of iterative computation of word frequency for every web document renders the search process expensive.
II. The relevance of web page is not known until it is downloaded.
III. Associated context of the web page is unknown prior to search initiation.
IV. The user interface of a search engine with keyword search is not flexible.

The proposed work paper effectively addresses the above-mentioned issues. A Context driven focused crawler has been designed, which uses augmented hypertext document structure coupled with a category tree for providing user interface at the search engine side.

1.1 Augmented Hypertext Documents

The information on WWW is organized in the form of a large, distributed and non-linear text system known as Hypertext Document system. HTTP and HTML provide a standard way of retrieving and presenting the hyper-linked documents. The XML offers more flexibility by allowing web page creators to use their own set of mark-up tags. This feature can be used to make augmentations in the hypertext documents for the suitability of web crawling [14]. The crawlers designed in PARCAHYD project [13] and [16,17] aim to enhance the performance and quality issues of crawlers using the concept of augmented hypertext documents. For instance, to manage the volatile information, variable information of a document is marked through volatile tags [15], which in turn are extracted out from the document along with their associated volatile information. The tags and their contents are then stored in a file having same name as document but different extension (.TVI). The hypertext documents that support .TVI and other related augmentations [14-17] are known as Augmented hypertext documents.

1.2 Category Tree

A category tree [5] is used as a graphical user interface in the search engine. It is a pre-defined canonical topic taxonomy with example keywords. To run a specific instance, initial input has to be provided in two forms. The user has to select and/or refine specific topic nodes in the taxonomy, and may also need to provide additional example keywords. The user, then, selects the example keywords of his interest in corresponding topic or category node. Subsequently, these selections are submitted to the search engine.

3. THE PROPOSED DESIGN OF CONTEXT DRIVEN FOCUSED CRAWLER(CDFC)

```
head>
<title>Crawler Information</title>
<meta name = "context" content = "General
Information about Crawler"/>
<meta name = "keywords" content = "Crawler, Web
pages, Search Engine, Spiders, Wanderers, Worms"/>
</head>
<body>
A Crawler is a program that retrieves web pages commonly
for use by a search engine. It traverses the web by
downloading the documents and following links from page
to page. Web crawlers are also known as spiders
</Keyword> or wanderers, or worms etc.
</body>
```

Figure 1: Sample Augmented XML cod

For the proposed work, the context of the required information has been augmented to the hypertext document wherein the tag names called '*keyword*' and '*context*' are explicitly marked at the time of creation of a hypertext document by the author. As an example, consider the sample XML code shown in Fig. 1.

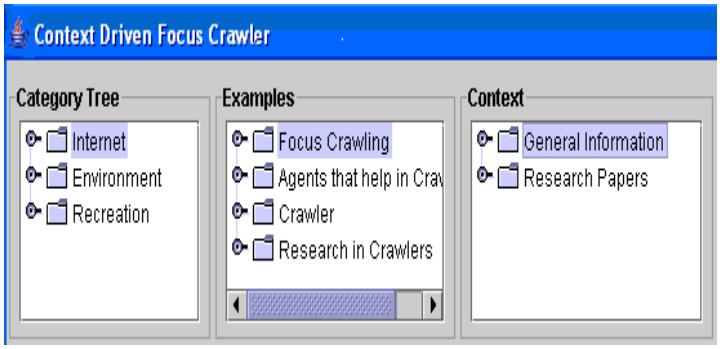

Figure 3: Modified Category tree for Search Engine

At the time of saving the document, all the keyword tags along with context and keyword tags are extracted out and stored separately in a file having same name but with different extension (say .TOC). The .TOC file extracted from the sample code of Fig. 1 is shown in Fig. 2. It may be noted that the TOC (Table of Contexts) is definitely much smaller in size as compared to the whole document.

Context	General Information about Crawler
Keywords	Crawler, Web pages, Search engine, Spiders, Wanderers, Worms

Figure 2: TOC file for sample code of Fig. 1

The category tree has also been suitably modified for the proposed design such that the context is also displayed with category examples. As shown in Fig. 3, the user selects a Category node (say Internet), and then its related examples are displayed. When the user selects an example (say Crawler), the two associated contexts are shown and finally, the user selects the context (say General information). In fact, the modified Category tree is a pre-specified collection of various categories in a graphical interface showing the various examples under these categories with their *contexts*. The user can choose any of the associated contexts by selecting *Category* → *Examples* → *Contexts* in the order. Nevertheless, if needed, new examples can be inserted by the user, which may later on linked to the contexts by the crawler.

For the purpose of crawling the web, CDFC employs three agents namely User agent, Matcher agent and Dbase agent as listed in Table 1. A brief discussion on these agents and their related components is given below:

Agent	Responsibilities
User agent	Acts as interface between the user and the system. Accepts user selections of keywords and context and displays documents to the user
Matcher Agent	Matches the user keyword with the keywords in the database, retrieves their contexts & URLs and sends them to user agent
Dbase Agent	Acts as interface between database and the external world. Stores and updates the TOC files and documents downloaded by the crawler in the database

Table 1: Agents and their Responsibilities 1. User Agent:

The user agent is responsible for the following activities:

1. It accepts the user selection of category node and related keywords from the category tree. It sends this information to the matcher agent to retrieve the associated contexts and their links from the database.
2. On getting associated contexts and their links from matcher agent, the user agent displays the list of contexts to the user in category tree.
3. The user agent accepts the context selected by the user and displays its all corresponding links.
4. On selection of a link by the user, user agent sends this link to Retrieve_Doc_Process to retrieve the document from the database. If the document is not present in the database, it passes the link to crawler to download the document.

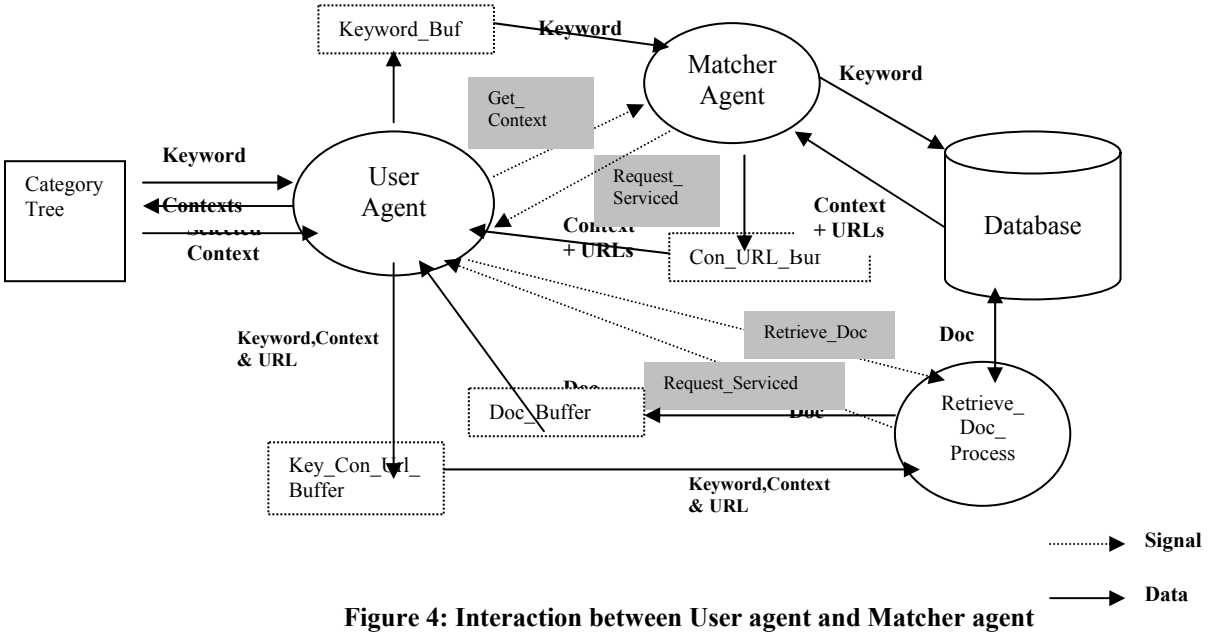

Figure 4: Interaction between User agent and Matcher agent

2. Matcher Agent: The matcher agent is responsible for the following activities:

1. It gets the keyword from the user agent and searches the keyword in the database to retrieve the corresponding contexts and their URLs.
2. The contexts and their associated URLs are sent to the user agent.

3. Retrieve_Doc_Process: It is responsible to search and retrieve the document in the database corresponding to a URL.

The interaction of user agent with matcher agent and Retrieve_Doc_Process (shown in Fig. 4) is described as follows:

1. It accepts the user selection of category node and related keywords from the category tree, stores the keyword in *Keyword_Buffer* and sends the message *Get_Context* to Matcher agent.
2. Matcher agent extracts the keyword from the Keyword_Buf and matches it with the keywords stored in the database. If the keyword is found, it retrieves its related contexts and URLs, stores them in *Con_URL_Buffer* and sends the signal *Request_Serviced* to the User agent.
3. User agent extracts the contexts and their URLs from the buffer and displays them to the user in category tree.
4. The user selects one of the contexts in the category tree. The user agent stores the keyword, its selected context and corresponding URL in *Key_Con_Buffer* and sends the message *Retrieve_Doc* to the Retrieve_Doc_Process.
5. Retrieve_Doc_Process extracts the keyword, context and URL from the buffer and searches the database for the document corresponding to URL. If the document is found, it stores the document in *Doc_Buffer* and sends the message *Request_Serviced* to the user agent.
6. The user agent extracts the document from the buffer and displays it to the user.

4. Dbase Agent: It is responsible for storing and updating the database whenever a new document or TOC is downloaded by the crawler.

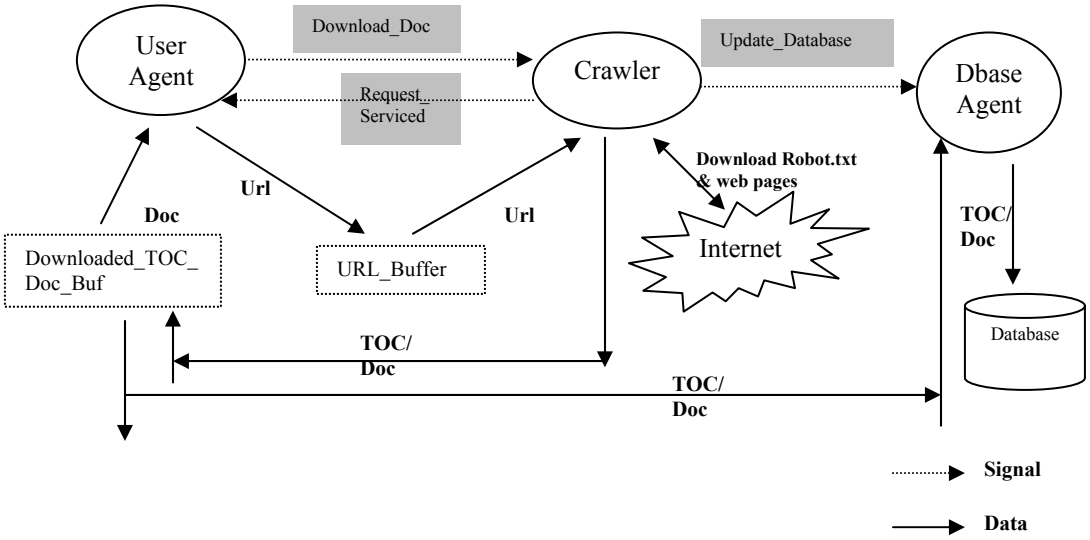

Figure 5: Interaction between User agent and Dbase agent & Crawler

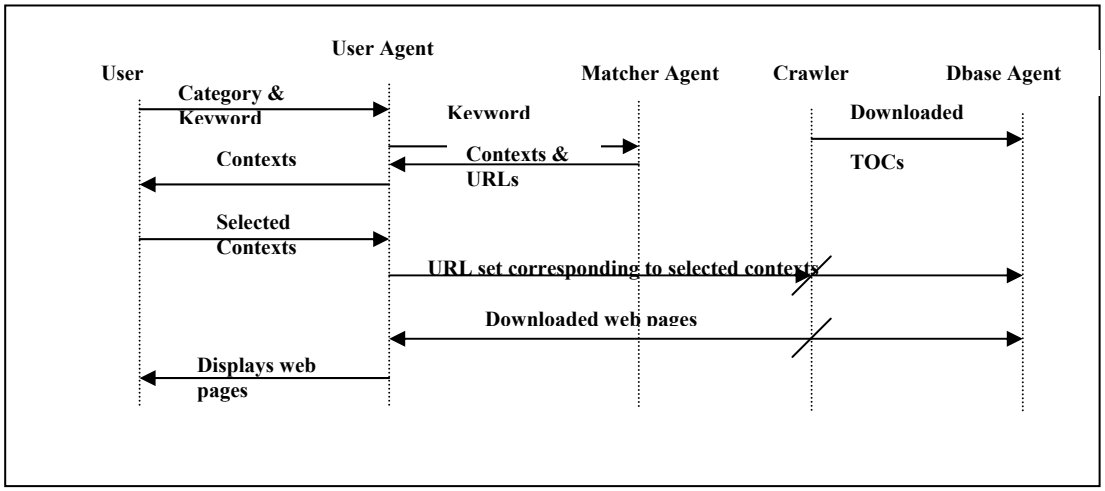

Figure 6: Sequence interaction diagram for various active components of CDFC

5. Crawler: The crawler continuously downloads the TOC files from the WWW in the background and stores them in the database. It also downloads the documents from the web on the request from the user agent and stores them in the database.

The interaction between the user agent, Dbase agent and crawler (shown in Fig. 5) is described as follows:

1. If the user agent needs a new document to be downloaded by the crawler, it stores the URL of that document in the *URL_Buffer* and sends the message *Download_Doc* to the crawler.
2. The crawler extracts the Url from the buffer and downloads the document from the web. Thereafter, it stores downloaded document in the *Downloaded_TOC_Doc_Buffer* and sends the message *Request_Serviced* to the user agent. Simultaneously it sends the message *Update_Database* to the Dbase agent to store the downloaded document in the database.
3. The user agent extracts the document from the buffer and displays it to the user.
4. Dbase agent also extracts the document from the buffer and stores it in the database.

The interactions between various active components of CDFC have been shown along the time line axis (see Fig. 6).

4. PERFORMANCE BENEFITS

The performance benefits of the proposed crawler are evaluated based on the following parameters:

1. **Harverst Ratio**: It is the rate at which relevant pages are acquired and how effectively irrelevant pages are filtered off. Since all web pages are retrieved according to the context selected by the user in CDFC, number of irrelevant pages is almost zero. Thus, the harvest ratio is high.
2. **Precision**: It is the ratio of number of relevant pages to the number of acquired pages. This is also high in CDFC as almost all pages are relevant to the user.
3. **Storage Requirements**: In CDFC, no document is downloaded if the user has not requested it. Therefore it does not index the documents which will never be used. Moreover the number of documents downloaded is very less in number as only related web pages are downloaded. Thus, storage requirement is very less as compared to other conventional crawlers.
4. **Search Time**: Since, the database size is very less in CDFC; it does not take much time to present the search results to the user.
5. **Network Traffic**: Since only highly related web pages are downloaded, which are very less in number and the size of TOC files being very less (5% of the original document), a significant amount of network traffic is reduced in CDFC.

Thus, the proposed crawler presents a flexible and interactive user interface in the form of category tree so that the user is guided in selecting the proper keywords along with their contexts for the web search. CDFC downloads only highly related documents, which are very less in number, thereby reducing the problem of information overkill faced by the user. Moreover, network traffic is reduced, as irrelevant web pages are not download

5. CONCLUSION

The proposed design of context driven focused crawler (CDFC) is based on the augmented hypertext document wherein the context of the keywords is stored in the form of TOC (Table of Contexts). The TOC coupled with a category tree provides context of the keywords. This design not only avoids the expensive complex computations for deriving the context of the user keywords but also reduces the network traffic significantly. Moreover, the quality of downloaded documents is in conformance with the topic and context of the user choice.

REFERENCES

[1] Raj Kamal -- *Internet and Web Technologies*; Tata McGraw Hill, 2003

[2] Junghoo Cho, Hector Garcia-Molina, " Parallel Crawlers", *Proceedings of the 11th International World Wide Web Conference*, Technical Report, UCLA Computer Science, 2002

[3] Mike Burner. , "Crawling towards eterneity: Building an archive of the worldwide web ", *Web Techniques Magazin*e, 2(5), May 1998.

[4] Martin Ester, Matthias Grob, Hans-Peter Kriegel, "Focused Web Crawling: A Generic Framework for specifying the user interest and for Adaptive crawling strategies*", Proc. of 27th International Conference on Very Large databases(VLDB '01*), 2001

[5] S. Chakrabarti, M. Van Den Berg, B. Dom, "Focused Crawling: A New Approach to Topic specific web resource discovery", *Proc. Of 8th International WWW conference*, Toronto, Canada, May,1999

[6] Diligenti M., Coetzee F.M., Lawrence S., Giles C.L., Gori M., "Focused Crawling using context graphs", *Proc. International Conference on Very Large Databases (VLDB '00*), 2000,pp. 527-534

[7] Yang Yongsheng, Wang Hui, "Implementation of Focused Crawler", *COMP630D Course Project Report*

[8] J.Kleinberg, "Authoritative Sources in a Hyperlinked Environment", *Proceedings of the 9th ACMSIAM Symposium on Discrete Algorithms*, San Francisco, California, USA, 1998.

[9] Junghoo Cho, Hector Garcia-Molina, L.Page, "Efficient crawling through URL ordering", *Proc. of 7th International WWW conference*, Brisbane, Australia, April, 1998

[10] Steve Lawrence, "Context in Web Search", *IEEE Data Engineering Bulletin*, Volume 23, Number 3, pp. 25-32, 2000

[11] S. Brin, L. Page, " The Anatomy of a Large-Scale Hypertextual Web Search Engine", *Proc. of the 7th International World wide web Conference*, Brisbane, Australia, 1998.

[12] F. Crimmins, "Focused Crawling review", 2001

[13] A.K. Sharma, J.P. Gupta, D.P. Agarwal, "PARCAHYD: An architecture of Parallel Crawler based on augmented hypertext documents" *Communicated to IASTED Journal of Computers and Applications*, June 2005

[14] A.K. Sharma, J.P. Gupta, D.P. Agarwal, "Augmented Hypertext Documents suitable for parallel crawlers", *Proc. Of WITSA-2003, a National workshop on Information Technology Services and Applications*, Feb'2003, New Delhi

[15] A.K. Sharma, J.P. Gupta, D.P. Agarwal, "A Novel Approach towards Efficient management of Volatile Information", *Journal of Computer Society of India* (CSI), July-Sep. 2003

[16] A.K. Sharma, Naresh Chauhan, Amit Goel, "An Agent based Crawler for Management of Volatile Information on World Wide Web*", In Proceedings of National Conference on Communication & Computational techniques (NCCT '06)*, Dehradun, Feb. 2006

[17] A.K. Sharma, Naresh Chauhan, "Demand Crawling based Efficient Web Search for Mobile Clients", *In Proceedings of National Conference on Information & emerging Technologies*, Ropar, Punjab, Feb. 2006

[18] Naresh Chauhan, A.K. Sharma, "A Comparative Analysis of Focused Crawling Techniques", *Proceedings of National Conference on IT*, Panipat, March 2006