An Enhanced Rule based Architecture for Exploring Topic Specific Web pages in WWW

J. Akilandeswari and N. P. Gopalan[1]

Sona College of Technology, Salem, Tamilnadu
[1]National Institute of Technology, Tiruchirappalli, Tamilnadu

ABSTRACT

Due to the rapid growth of World Wide Web the search engines gain greater importance nowadays. Every normal user relies on them to get the information they need. Topic-specific crawlers have become an important thrust area in the field of information retrieval in WWW. In this paper, a web spider architecture based on multi-agent system for cooperative information gathering is presented. The system collects Web pages that are on topic. The system employs two types of agents: retrieval and coordinator agents. Coordinator agents are responsible for disseminating crawling frontiers to individual retrieval agents. The URL frontier is built by a rule based engine which decides the next URL to be downloaded. The rule base is built dynamically. The coordinator agent is also built with fault tolerance mechanism to gracefully degrade in case of failure. The retrieval agents download the Web pages and return them along with the relevance score. The URLs are assigned dynamically so that the duplicate downloading of Web pages is avoided. The empirical results clearly depict the advantage of rule based design in terms of harvest rate and time.

1. INTRODUCTION

The Web is exploding in a very fast rate[1]. This explosion has provoked the research community to develop retrieval tools like search engines in getting the relevant information from World Wide Web (WWW). Web crawler is an important in search engines. It is a program which visits the pages in WWW in a methodically, automated manner to generate a copy of all the visited pages for later processing by the search engine [2]. Keeping in mind the currently available web pages and growth rate of Web, crawling 'all' pages requires enormous network and hardware resources. Also nowadays, the users are very much specific in getting the pages that are relevant to their topic of search. In general, there are three directions in which the web spider research can be carried out:

- Speed and efficiency: This direction contributes to the study of different ways to increase the harvest speed of a spider. Examples include Mercator [3], Internet Archive's crawler [4, 5], and Google's crawler [6].
- Spidering policy: The studies under this category reveal the behaviors of spiders and their impacts on other individuals and the Web as a whole. A well-designed, "polite" spider should avoid overloading Web servers. There are two standard ways. The first one, called the robot exclusion protocol, allows Web site administrators to indicate, by specifying a file named robots.txt in the Web site's root directory, mentioning which parts of their site should not be visited by a robot [7]. In the second method, usually known as the robots META tag,Web page authors can

indicate to visiting robots whether a document may be indexed, or used to extract more links.

- Information retrieval: Most Web spider research belongs to this category. These studies explore different spidering algorithms and heuristics that can be used by spiders to retrieve relevant information from the Web. Many of these studies apply to Web spider techniques that have been shown to be effective in traditional information retrieval applications, e.g., the vector space model [8].

Restricted bandwidth, storage, and computational resources, and to the dynamic nature of the Web, search engines cannot index every Web page, and even the indexed portion of the Web cannot be monitored continuously for changes. In fact an estimate done on March 2002 says that the visible Web is at around 7 billion static pages [9]. This estimate is more than triple the 2 billion pages that the largest search engine, Google, reports at its Web site [10]. The number of Web pages available now may be 10 to 20 times more than the number of Web pages in 2002. Looking for the related information in such a large portion of Web is quite cumbersome and challenging.

Topic specific crawlers are developed to gather pages that are relevant to the triggering topic. Generally, implementing a crawler with single process to gather all pages is certainly difficult. Therefore, many search engines often employ multiple processes in parallel to perform the task. The implementation of this design paradigm is referred as parallel spider which greatly improves the collection efficiency. There are certain issues to be taken care while designing a parallel crawler[11]:

- Overlap: There are possibilities that different processes download the same page multiple times. One process may not be aware that another process has already downloaded the page. This redundancy should be minimized to save network bandwidth and increase the crawler's effectiveness.
- Quality: Each process makes its own decision in downloading the pages. There should be some metric to make sure that the quality of the downloaded pages is as good for a parallel crawler as for a centralized one.
- Communication bandwidth: To improve the crawling effectiveness, crawling processes need to periodically communicate to coordinate with each other. This increases the communication overhead.

The term agent describes a software abstraction, an idea, or a concept, similar to Object Oriented Programming (OOP) terms such as methods, functions, and objects. The concept of an agent provides a convenient and powerful way to describe a complex software entity that is capable of acting with a certain degree of autonomy in order to accomplish tasks on behalf of the user. But unlike objects, which are defined in terms of methods and attributes, an agent is defined in terms of its behavior. Agents itself have several characteristics that makes researchers interested to explore the agent technology. The characteristics are as follows [12]:

- Autonomous
- Goal-directed
- Task able
- Cooperative with other agents to accomplish its tasks.
- Communicative with other agents
- Adaptive

A multi-agent system (MAS) is a system composed of multiple agents acting collectively to reach the goals that are difficult to achieve by an individual agent or monolithic system. Multi agents are:

- Heterogeneous agents having expertise in different areas.
- Self-motivated
- Act to fulfill internal goals
- Share tasks with others
- Communicate and collaborate
- No global or centralized control mechanism

In this paper, a crawling architecture is presented to collect Web pages that are on topic of user's interest.

The design consists of multiple, parallel crawling agents and a coordinator agent to synchronize them. In this prototype, the crawling agents are included with certain intelligence in guiding them to download the next correct URL. This feature enables the crawler in avoiding visiting unproductive Web pages.

The remainder of this paper is organized as follows: section 2 discusses on the work related to Web crawlers. In section 3, the architectural framework of the crawler is described. In section 4, experimentation and evaluation details are discussed. Finally in section 5, future directions and conclusions are presented.

2. RELATED WORK

Web crawlers have been studied since the advent of the Web. In [22] a crawler was described that orders the URLs according to the back link count. Since the objective of topic specific crawler is to download only a small subset of the Web, they need to decide what page to download next. A crawler has to may improve the "quality" of the downloaded pages by retrieving "important" or "relevant" pages. The works described in [21, 23, 24, 25] proposed different algorithms to identify important pages early.

Focused Crawler was introduced by Chakrabarti in 1999 [23]. In [20], the authors evaluated a topic-Driven web crawler and compared their crawler with different crawling strategies. The paper [27] explains the use of Reinforcement Learning in the design of a spider. They computed the probability that a hyperlink in a page is likely to be linked to a relevant page, according to the texts in the hyperlink neighborhood and as a result their crawler follows the link with the highest probability value. The work described in [28] identified the URL to be crawled next by evaluating a learning mechanism in order to increase the efficiency in topic specific web resource discovery.

There are many works in the literature that uses agent's framework to deploy the functionalities of a crawler. Agent-based computing has long been suggested as a promising technique for application domains that are distributed, complex, and heterogeneous. Agent based technologies are being applied to a wide range of complex problems from interface agents [13, 14] to recommender systems [15] and autonomous and comparative shopping agents [16, 17]. There are agents to retrieval relevant Web pages in response to user generated topics [18, 19, 20]. In [11], the design of a parallel crawler has been studied extensively. The paper concentrated on eliminating the downloading of same web pages by multiple crawlers. They have not employed multi-agent systems which can be great help in parallel computing environment.

3. ARCHITECTURE

Fig. 1 portrays the architecture of the crawler employing multiple agents. There are three problems to consider while designing a parallel crawler:

- Deciding whether a page is on topic or not
- Choosing the next page to download
- Avoiding multiple agents downloading the same Web page

The retrieval agents are responsible for the first issue. The next two issues are considered by the coordinator agent. Our system focuses on employing intelligent multiple agents that mine the information contained in both hyperlinks and content. The advantages in this system are two fold: one is to reduce

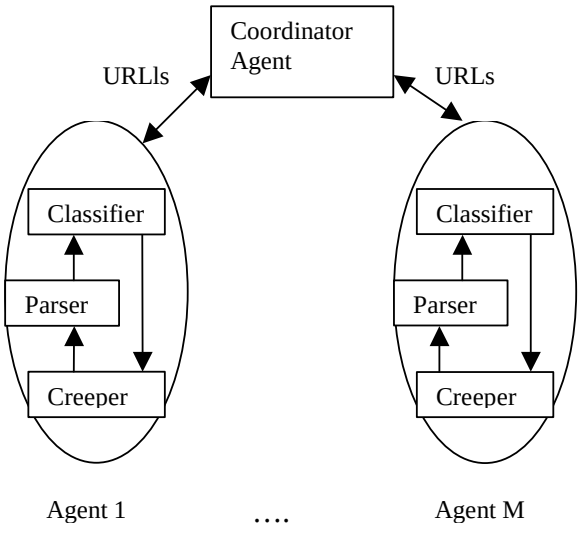

Fig. 1: Architecture of the Web crawler

the network traffic load and another is to parallelize the computation. The intelligence is built on both the agents. During the start the crawler should have knowledge such as seed URLs, topic specific keywords and URL prediction. For seed URLs the crawler needs some good URLs, which point to many relevant web pages. Also it needs some relevant keywords of specific topic. The crawler has to predict the downloaded pages. These knowledge bases form the experience for the crawler to learn.

3.1 Coordinator Agent

Coordinator agent is responsible for synchronizing retrieval agents. The user interface inbuilt in the agent collects the starting URLs to initiate the crawl. To improve the quality of the result sets, the crawl is started from pages that are assumed to be relevant. These relevant seeds are obtained from a trustable search engine i.e. from Google. The agent engenders number of retrieval agents and places them in each seed URL. The retrieval agents in turn return the classified hyperlinks with appropriate relevance scores. The agent performs the URL seen test and inserts the hyperlinks in the frontier along with the values in descending order. Since the URLs are assigned by the coordinator agent, the retrieval agents will never be consigned with already visited Web pages. This dynamic assignment avoids visiting the same page multiple times by different crawling agents.

Since a single agent coordinates the activities of the retrieval agents, there is a disadvantage of being a bottleneck. But this weakness is overcome by having a standby master agent called mirror agent which is periodically updated with the computational paradigms of the coordinator agent. If there are any problems detected in the coordinator agent, then the computations are gracefully handed over to the mirror agent and finally it takes control over the whole operations.

The crawling agents are monitored by this control agent for proper synchronization. Even if one crawling agent is crashed due to problems such network error, or logical error, the master agent detects the downfall, transfers the computations to another newly spawned agent and kills the old one.

3.2 Retrieval agent

These agents start the crawl by downloading the Web page. The page is parsed and is given as input to the classifier. The hyperlinks in the page are given to the coordinator agent. The Naïve-Bayes classifier is trained with the help of topic taxonomy like Yahoo!. The creeper gets the relevance score of the downloaded page. According to radius-1 hypothesis, the hyperlink from an off-topic page is not all visited. There may be some off topic pages with hyperlinks pointing to topic of search. In [25], a rule based system was developed to exploit the inter-class relationships. The relevance scores are computed based on associations among the classes. In this paper, we have built rules and computed normalized relevance scores for each paged downloaded as follows:

$$R[p] = NHypi \rightarrow i \,/\, NHypi \rightarrow i \; NHypi \rightarrow j \qquad ...(1)$$

where $NHypi \rightarrow i$ is number of hyperlinks in a page that belong to the same class and $NHypi \rightarrow j$ is number of hyperlinks that belong to other classes. Instead of considering the probabilities, the counts are taken into care. This computation gives 20% more efficiency in finding quality Web pages. The hyperlinks with relevance scores are sent to the coordinator agent in batches.

4. DISCUSSION

Our work is implemented using JADE, the latest platform for multi agent system. JADE is one of the most used and promising agent development frameworks. JADE also support the development of multi agent system through the predefined programmable and extensible agent model and a set of management and testing tools. The environment allows each agent to dynamically discover other agents and to communicate with them according to the peer-to-peer paradigm [26]. Communication is the main part in multi-agent architecture. The agent communication language FIPA ACL is used. The main features of FIPA ACL are the possibility of using different content language and the management of

conversations through predefined interaction protocols.

For implementation, DMOZ topic taxonomy is chosen. 1000 classes are considered for testing. Our crawler has been tested with two baseline crawlers: one is a modified focused crawler without multiple agents and other is multi-agent based focused crawler without dynamic assignment of URLs. The crawlers are made to crawl upto 10000 pages, that is N=10000. The initial number of retrieval agents spawned by the coordinator agent is 25.

The crawler's performance is measured with the help of harvest ratio which is the average relevance of all pages retrieved on a particular topic.

$$\text{Harvest ratio} = \frac{\text{Summation of relevance score of each hyperlink on topic}}{N}$$

Fig.2 shows the comparison of three crawlers in terms of harvest rate. The performance of our design seems to be promising.

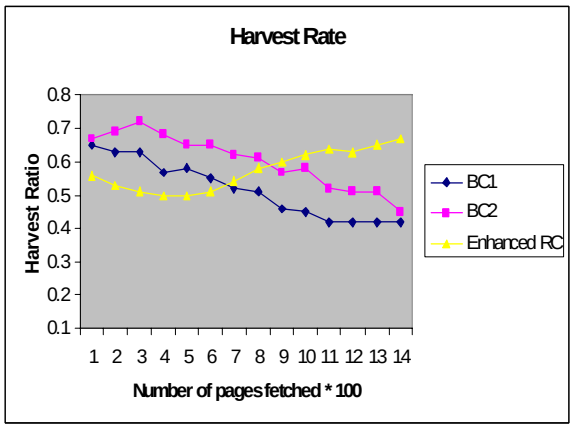

Fig. 2: Harvest Ratio

Another observation made on the crawler design is the time taken by the crawlers to download upto 7000 pages. The study is depicted in Fig. 3.

Two parameters are very crucial in deciding the usefulness of the spider. One is harvest rate and the other is the time taken to visit predefined number of pages. In both the measures our design performs well.

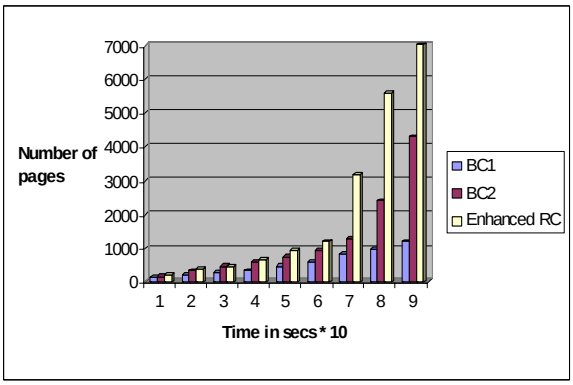

Fig. 3: Time taken to download pages

We made an important observation of memory utilization of both the agents: coordinator agent and retrieval agents. In our experimental setup, the system gets crashed down due the memory unavailability if the number of spawned crawling agents goes beyond 25. This problem must be alleviated by using appropriate data structure.

5. FUTURE DIRECTIONS AND CONCLUSIONS

The work can be extended in the following directions:

- With sophisticated rule discovery techniques
- Having coordination among the retrieval agents
- Choosing appropriate data structures to lessen the problems created due the memory unavailability.

This paper has discussions on the issues of designing a focused and parallel crawler. The fast improvements in the computational tools, the rapidly growing web needs a fresh approach. The contributions of this paper include: proposed an architecture which makes use of two types of agents: coordinator and retrieval agents. The main goal of finding quality results is achieved by employing multi-agent system coordinating with each other. The paper also suggested a technique to avoid duplicate downloading of Web pages by different agents. The experimental results advocate that the pages gathered from World Wide Web are of high quality.

REFERENCES

1. P. Lyman, H.R. Varian: How Much Information. [Online]. Available at http:// www.sims.berkeley.edu/how-much-info/ , 2000.
2. F.C. Cheong, *Internet Agents: Spiders, Wanderers, Brokers, and Bots*, New Riders Publishing, Indianapolis, Indiana, USA, 1996.
3. A. Heydon, M. Najork, "Mercator: A Scalable, Extensible Web Crawler", World-Wide Web, pp219-229, 1999.
4. M. Burner, " Crawling Towards Eternity: Building an Archive of the World-Wide Web", Web Techniques, 2 (5), 1997.
5. B. Kahle, "Preserving the Internet", Scientific America, 1997.
6. S. Brin, L. Page, "The Anatomy of a Large-scale Hypertextual Web Search Engine", Proc. Of the 7th International World-Wide Web Conference, 1998.
7. M. Koster, "A Standard for Robot Exclusion", Available at: http://www.robotstxt.org/wc/ norobots.html, 1994.
8. G. Salton, "Another Look at Automatic Text-retrieval Systems", Communications of the ACM, 29 (7) 648-656, 1986.
9. Cyveillance, "Sizing the internet", White paper, July 2000. http://www.cyveillance.com.
10. Google. http://www.google.com.
11. Junghoo Cho, Hetor Gasrcia-Molina, " Parallel Crawlers", WWW 2002.
12. F. Bellifemine, G. Caire, A. Poggi, D. Greenwood, *Developing Multi-Agent Systems*. Wiley Series in Agent Technology, 2007.
13. T. Helmy, S. Amamiya, and M. Amamiya, "User's ontology-based autonomous interface agents", In The 2001 International Conference on Intelligent Agent Technologies, Maebashi City, 2001.
14. H. Lieberman, "Autonomous interface agents", In Proc. ACM Conference on Computers and Human Interface, Atlanta, GA, 1997.
15. M. Balabanovi_c and Y. Shoham, "Content-based, collaborative recommendation", Communications of the ACM, 40(3), 1997.
16. R. Doorenbos, O. Etzioni, and D. Weld, "A scalable comparison-shopping agent for the World-Wide Web", In Proceedings of the First International Conference on Autonomous Agents, pages 39-48, 1997.
17. F. Menczer, W. Street, N. Vishwakarma, A. Monge, and M. Jakobsson, " IntelliShopper: A proactive, personal, private shopping assistant", In Proc. 1st ACM Int. Joint Conf. on Autonomous Agents and MultiAgent Systems (AAMAS 2002), 2002.
18. D. Eichmann, "The RBSE spider - Balancing effective search against Web load", Computer Networks, 4(2):281-288, 1994.
19. F. Menczer and R. Belew, "Adaptive retrieval agents: Internalizing local context and scaling up to the Web", Machine Learning, 39(2-3):203-242, 2000.
20. F. Menczer, G. Pant, M. Ruiz, and P. Srinivasan, "Evaluating topic-driven Web crawlers", In Proc. 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2001.
21. R. C. Miller and K. Bharat. SPHINX: a framework for creating personal, site-specific web crawlers. In Proceedings of the Seventh World-Wide Web Conference, 1998.
22. J. Cho, H. Garcia-Molina, and L. Page, "Efficient crawling through URL ordering", Computers networks and ISDN systems, 30:161–172, 1998.
23. S. Chakrabarti, M. van den Berg, and B. Dom, "Focused crawling: A new approach to topic-specific web resource discovery", In The 8th International World Wide Web Conference, 1999.
24. M. Diligenti, F. M. Coetzee, S. Lawrence, C. L. Giles, and M. Gori, "Focused crawling using context graphs", In Proceedings of the Twenty-sixth International Conference on Very Large Databases, 2000.
25. Ismail Sengör Altingövde and Özgür Ulusoy, "Exploiting Inter-Class rules for Focussed Crawling", IEEE Intelligent Systems, Volume 19 , Issue 6, pp 66-73, 2004.
26. M. Nikraz, G. Caire, and P. A. Bahri, "A Methodology for the Analysis and Design of Multi-Agent Systems using JADE", International Journal of Computer Systems Science & Engineering, special issue on "Software Engineering for Multi-Agent Systems", 2006.
27. J. Rennie and A. McCallum, "Efficient Web Spidering with Reinforcement Learning", Proceedings of the 16th international Conference on Machine Learning (ICML-99), 1999.
28. N. Angkawattanawit and A. Rungsawang, "Learnable Crawling: An Efficient Approach to Topic-Specific web Resource Discovery", 2nd international Symposium on communications and Information Technology (ISCIT 2002), October 2002.