Proceedings of the 5th WSEAS Int. Conf. on APPLIED INFORMATICS and COMMUNICATIONS, Malta, September 15-17, 2005 (pp384-390)

A Distributed, Fault-tolerant Multi-agent Web Mining System for Scalable Web Search

N.P. GOPALAN, J. AKILANDESWARI
Department of Computer Science and Engineering
National Institute of Technology
Tiruchirappalli – 620 015, Tamil Nadu,+91 (431) 2501801
INDIA

Abstract:- With the increasing amount of information available over Internet, there is a need for a more specialized search tool to retrieve quality web pages relevant to the topic of search. The traditional search engines index only the static collections of web and there is a large gap between the current contents of web and the image of WWW stored in search engine databases. An attempt has been made to reduce this shortcoming by mining the web online at query time with adaptive learning multi-agents. The proposed system is scalable in the sense that it downloads several hundreds of relevant web documents per second and is resilient against the system crashes with graceful degradation. It is found that the present technique, as a complementary tool to any of the existing search engines provides a better performance with respect to recency, recall, precision, and coverage.

Key-words:- Web mining, multi-agents, precision, recall, recency, coverage, information retrieval.

1 Introduction

With abundant information, World Wide Web (WWW) has become the important source of building intelligence for businesses and search engines. There are various techniques for locating information like web search engines, meta searching, post retrieval analysis, and enhanced web collection visualization [1].

Mostly, the traditional search engines do not give serious attention to the dynamic nature of web. Lot of computations are done at web servers and the result is the dynamic generation of web pages which contributes to the potential search space.

Search engines use indexing algorithms to index web pages using a program called *web crawler* or *robot* that visit web pages regularly and index them. They can be distinguished on two aspects: crawling and ranking strategy [4]. Crawling strategy decides which page to download next and the ranking strategy contributes to the retrieval of search results based on rank. The above two strategies decides the parameters like *recall* (fraction of relevant pages that are retrieved), *recency* (fraction of retrieved pages that are current) of a search engine and *precision* (fraction of retrieved pages that are relevant) of a search engine.

The conventional standard approaches are best suited for static collections of documents and are not scalable. But in the case of WWW the sets of relevant pages are highly dynamic. For example, thousands of web pages are getting uploaded in the web everyday. The enormous and dynamic nature of web makes it difficult to maintain updated and exhaustive collections on any given topic.

Even though there are quite a few techniques to crawl intelligently in the web, they lack scalability and coverage. This is due to the fact that the static "index snapshot" approach does not scale with the growing nature of the web. Discrepancies between the current web contents and their snapshots represented by search engines' indices are the root cause affecting scalability. The study on coverage among six most popular search engines reveals that it varies approximately between 3% and 34% of the web's indexable pages [10]. In order to keep indices up-to-date, crawlers have to revisit documents to see if they have been changed, moved, or deleted. Further, crawlers have to try to exhaustively visit every new document to keep indices as complete as possible.

Despite its huge size, the web forms a small-world network, characterizing social and biological systems, in which any two documents on the web are on average 19 clicks away from each other [9]. This result suggests that "intelligent" agents mining the hyperlinks and their corresponding anchor text might be a novel approach to find useful information. With this approach, the relative load on the network can be minimized. The online mining agents used in the present work search only the current environment

and therefore, will not return the stale information and will have better chance to improve the recency of the visited documents.

Conventionally, agent based web mining systems can be classified under three categories [8]: intelligent search agents, information filtering / categorization and personalized web agents. Several intelligent web agents like Harvest[3], ShopBot[13], iJADE web miner[5], WebWatcher[1], Infospiders[7], myspiders[6], have been developed to search for relevant information using domain characteristics and user profiles to organize and interpret the discovered information.

Harvest[3] relies on pre-specified domain information about the types of documents or hard coded models of the information sources to retrieve and interpret documents. ShopBot[13] is a domain independent shopping agent that searches for the product information from variety of vendors. It learns to visit vendor home pages. IJADE web miner[5] makes use of fuzzy neural networks and fuzzy techniques to identify and produce comparative price lists for products.

Infospiders[7], myspiders[6] are multi-agent based systems, which use artificial intelligence techniques to learn and adapt itself to the current document visited. Clever [14] is a search engine that uncovers two types of pages: authorities, which provide the best source of information on a given topic, and hubs, which provide collections of links to authorities.

This paper attempts to find a solution for scalability problem by incorporating agent based systems for mining the hyperlinks on a web page to find more quality web pages. The solution is designed in such a way that it narrows down the gap between the current web content and the one found at the previous crawls. A significant characteristic of a web page is “quality”, which is the relevance of web pages that are to be retrieved as per the user’s request. In all other agent based information retrieval systems, the agent is made to crawl according to specified predefined information. The prototype system called *LACrawler* mines the hyperlink information and learns itself from the previous crawls and enhances the quality of web pages during retrieval.

The remainder of this paper is organized as follows: Section 2 describes the issues that are taken into consideration while designing the system architecture. Section 3 describes the architecture of the mining system, and Section 4 presents the implementation details. Section 5 presents the evaluation of the current system and its comparison with the other systems. Finally Section 7 offers concluding remarks.

2. Research Issues

Every search engine has its own crawling and ranking algorithms that influence the parameters like recall, recency, and precision. It is to be noted that the success of Google is attributed to its ranking algorithms.

The following are the issues which influence the design of intelligent agent based crawlers:

i. Incorporating a technique to calculate the relevance of a page.
ii. Automatic extraction of information from unfamiliar web sites.
iii. Incorporating learning ability.
iv. To make agents share their experience.
v. To make agents fault tolerant.

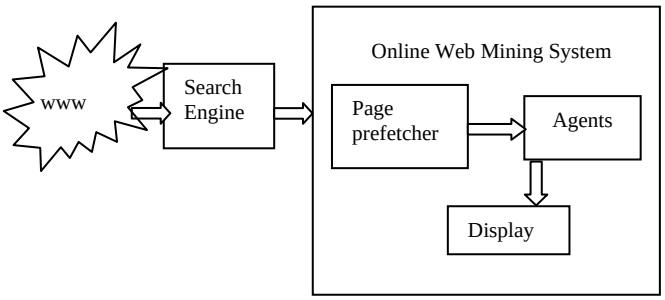
When employing multiple agents, care should be taken to make the overall system manageable, flexible and robust. The main question is how to incorporate scalability and in what terms. In this article, scalability is defined as the number of relevant pages downloaded per second. Another design issue is employing the learning capability in the system. It is known that feed-forward neural network is best suited for pattern identification problem [15].

As per the definition, the autonomous agents act continuously until it is eaten or otherwise dies automatically. To incorporate graceful degradation, the workload of the dead agent should be transferred to newly spanned agent. To identify the agent that is destroyed due to system crash or network interruptions, either timeout or message passing scheme or a combination of both may be employed.

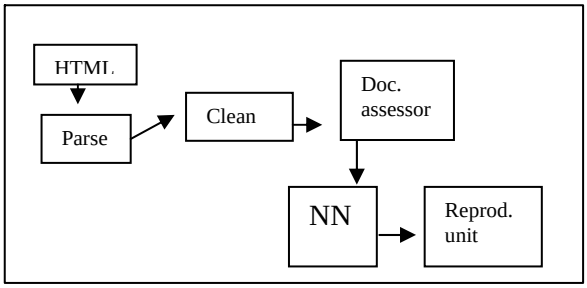
3. Architecture

As the hyperlink structure contains an enormous amount of information, which can help automatically infer relevance among documents, the present intelligent agent based system concentrates on mining the hyperlinks. When a hyperlink is created, it represents an implicit endorsement of the page being pointed to. These structures become an important source for mining and give a better understanding of the relevance and quality of the web’s contents.

The basic architecture of the online web mining system for searching the web for relevant documents is shown in figure 1.



Overall Architecture



Individual Agent's architecture

Figure 1. Architecture

The inputs to the system are: seed URLs, query terms or keyword list, and maximum number of pages to crawl. The seed URLs serve as the starting point of search. They can be obtained either from a domain expert or from a trusted search engine like Google. As the complete information regarding relevant pages is not known apriori, getting input from domain expert is a tedious task, if not impossible. Due to this difficulty, the input is obtained from the Google. The starting documents are pre-fetched and each agent is positioned at one of these documents and given an initial amount of energy to crawl.

Each agent analyzes the document on which it is currently placed with necessary stemming and noise removal activities. The outgoing links of the document are then examined using feedback neural network with reinforcement learning. The artificial intelligence technique is employed to estimate the relevance of the document to be fetched and displayed. The probability of relevance is calculated with the help of estimated link relevance. The computation of link relevance and its corresponding probability is described in later section. If that value is above a given threshold, the document is considered to be relevant to topic of search and the agent is made to crawl further by updating its energy or else the agent is killed. If the document to be fetched is relevant, the agent will reproduce placing the off-springs on those relevant documents. The documents fetched by the agents are displayed to the user and then stored in the data repository for further indexing using the relevance measure and probability.

Since the crawling process is continuous, the agents should tolerate the crashes and network disturbances without losing the data. For maintaining robustness, timeout scheme is employed. Every agent is monitored for the specified time limit within which they have to deliver the results. If not, they are considered as dead. To reassure and to avoid duplications, a message is also sent and if there is no answer, it is confirmed that a new agent should replace that agent's work. The newly spanned agent utilizes the main data structures that are stored periodically, for re-crawl. If the agent is killed explicitly, then the agent itself passes a 'killed' message. This hybrid scheme helps in monitoring the number of agents, which are active at any instant of time.

4. Implementation details

The web crawler design exploits computational parallelism to achieve efficiency. Hence, multithreading is expected to provide better utilization of resources. The distributed agent is implemented using java as it has built-in support for threads. To keep the interface as simple as possible, the users are kept transparent about the parameters used. The query terms and maximum number of pages to be crawled are given as input. Once the search is started, the system obtains the seed URLs from a trusted search engine like Google and initiates crawlers. A crawled page is displayed to the user if and only if its probability of estimated relevance is above a given threshold.

LACrawler Algorithm:

```
User_input(query_terms, MAX); // inputting
        keywords for search and  maximum
        of pages to be crawled
Seed_URLs = search-engine(query_terms);
// getting the seed URLs
Prefetch_document(seed_URLs);//fetching
        the documents of those URLs
Initialize_dagent ( query_terms, MAX); //
        Initiate the crawling process
for each dagent {
        position (dagent, document); //
                place the agent in the pre-
                fetched document
        dagent_energy=initial value;  }
        for each dagent {
          while visited < MAX {
            get_the_hyperlink( );
            for each hyperlink {
dagent_doc := fetch_new_document( );
```

```
estimate_link_relevance(query_terms, dagent_doc);
        // calculate relevance    using (1)
estimate_prob(link_relevance); //calculate using (2)
learning (dagent, link_relevance);
if estimate_prob>threshold
{
 display();
 store();
 dagent_energy+= link_relevance – latency( ); //
     updating energy
  reproduce( );
}
else
{
 Send ("killed");
 Kill;
}
}
}
        /* End of LACrawler */
```

The relevance of a link is estimated as follows:

estimate_link_relevance(q,p)=

$$\frac{\sum_{k\in p\cap q} f_{kq}\ f_{kp}}{\min\left(\sum_{k\in q} f_{kq}^{\ 2}\right)\left(\sum_{k\in p} f_{kp}^{\ 2}\right)} \qquad (1)$$

where q is the query, p is the page and f_{kp} is the frequency of term k in page p.

After calculating the link relevance, the probability of relevance is also calculated to confirm that following the link under consideration is truly beneficial. A document is considered relevant if and only if the probability of relevance is greater than the given threshold.

Let C be the event that current page is relevant satisfying the user's search term with P(C) as the associated probability. Let R_L be the event that document following a link is also relevant. The probability of relevance P, for the link to be followed can be computed as follows:

$$P = P(C) * P(R_L \mid C) + (1 - P(C)) * G) \qquad (2)$$

where P(C) is the probability that current page is relevant, $P(R_L \mid C)$ is the conditional probability that following R_L from C is also relevant, and G is the probability that any document is relevant.

The similarity measure described in (1) is used to calculate the probability measure. It can be substantiated that the probability measure helps in locating the relevant documents. The equation (2) can be rewritten as

$$P = P(C) * \frac{P(C \cap R_L)}{P(C)} + (1 - P(C)) * G \qquad (3)$$

The second term in the above equation is negligible as G usually tends to be less in many cases. Using (2), if P is calculated as significant figure then

$$P(R_L) > P(C) \qquad (4)$$

The above fact shows that following a link from more relevant current document is always beneficial, since

$$P(R_L \mid C) > P(C \mid R_L). \qquad (5)$$

To calculate the energy given to the agent, latency is used, which is the time spent on downloading the page of the corresponding hyperlink and is given by

$$\text{Latency} = \frac{no\_of\_seed\_URLs}{MAX} * \frac{time(p)}{TIMEOUT} \qquad (6)$$

In the above, *time (p)* is the time taken to download the page *p*, and *TIMEOUT* is the expiration time of the socket connection.

5. Evaluation and Experimental Results

The evaluation of the online search system is based on different characteristics like set of seed URLs, combination of query terms, crawling strategy, indexing strategy, recall, recency, and precision.

The system is assessed and compared with the breadth first search, a baseline heuristic in which the links to be followed are ranked according to the similarity between the query and the page containing links. It is found that the system performs well if the set of seed URLs and query terms match the set of relevant documents to be retrieved (see Figure 2).

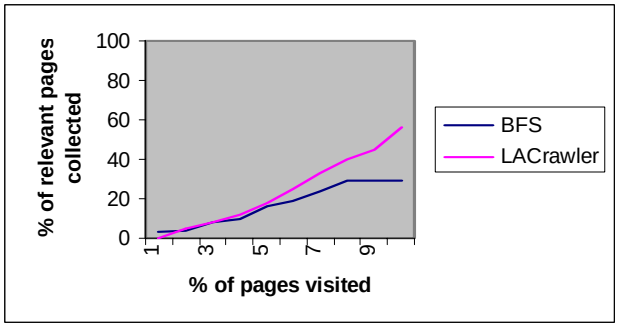


Figure 2. Performance of the system compared with heuristic technique

The above graph clearly depicts the learning advantage of the system. Initially the performance of both the systems are equal. Due to the learning process in each run, the percentage of relevant documents collected increases. Up to 10% of the total pages collected, both the crawlers collect same number of relevant documents. After that, LACrawler collects more quality pages. This is due to the mining activity embedded in the crawling

agents. After 80% of the total pages collected, BFS stops collecting relevant pages as its curve flattens, but the other continues to do its appreciable work.

An important observation is made with respect to the quality of seed URLs. If they are not chosen well, the performance is seemed to degrade. Even though parallelism is exploited, the total time spent on querying the web incurred due to post retrieval analysis and inter-agent communication, cannot be amortized. This time is not so considerable when considering relevancy as an important factor to achieve. Another factor influencing the quality is the proper choice of threshold value. In the experiments, the threshold value is chosen on trial and error basis. Initially the threshold value is started with 0.1. While experimenting, the value is gradually increased. It is studied that as the threshold value is increased to a maximum level, the documents that are at close proximity in relevance are not collected by the system, which is undesirable. So it is chosen as an optimum value (0.72) as shown in figure 3.

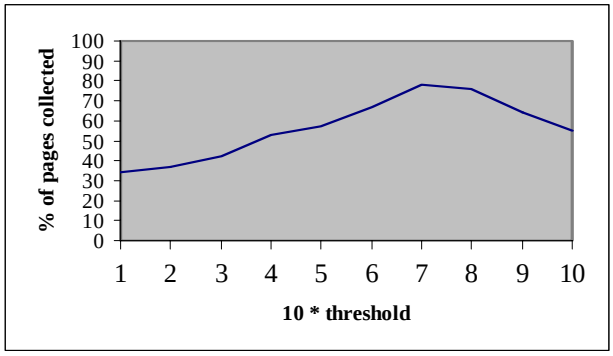


Figure 3. Threshold Vs % of pages collected

Let R be the pages retrieved using LACrawler. The precision of the retrieved results can be calculated as

$$P = \frac{1}{R} * (\sum_{p \in R} estimate\_link\_relevance(q, p)) \quad (7)$$

where p is the page and q is the query term. Recall and recency can be measured as

$$\text{Recall} = R * P \quad (8)$$

and

$$\text{Recency} = \frac{1}{R} * \frac{1}{\sum_{p \in R} \Delta t} \quad (9)$$

with $\Delta t = t_i(p) - t_j(p)$

$t_i(p)$ is the time when p is last modified and

$t_j(p)$ is the time when p was indexed by a search engine.

The three measures defined above (3 – 6) are used to evaluate the LACrawler against a search engine as web mining support tool. The current system is tested against Altavista, which is one of the search engines revealing the indexing date t_i, to evaluate recency.

LACrawler is made to visit 100 pages with 10 seed URLs and 10 query terms. figure 4 shows the plot for estimated precision

$$\overline{P} = \frac{1}{10} \sum_{q=0}^{9} P(R) \quad (10)$$

between number of pages returned by Altavista and LACrawler. At the beginning of the crawl Altavista provides a slight advantage over LACrawler. After visiting 10 pages the later outperforms the form

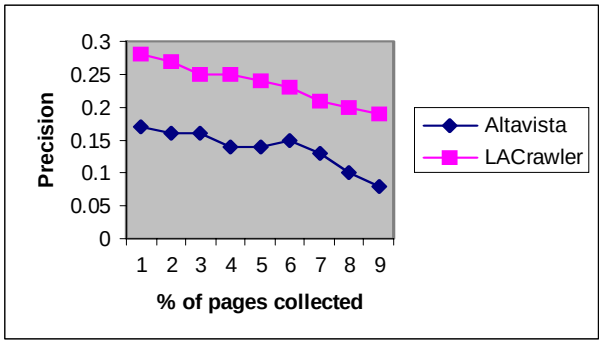


Figure 4. Estimated Precision

Figure 5 plots the estimated recall

$$\overline{R} = \frac{1}{10} \sum_{q=0}^{9} \text{Re}\,call(R) \quad (11)$$

versus the number of pages retrieved by Altavista and the current system. At the end of the crawl Altavista stops retrieving the relevant documents, while LACrawler continues to get quality pages.

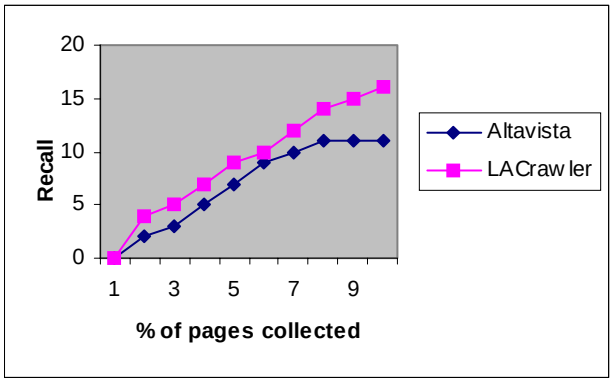


Figure 5. Estimated Recall

The estimated recency

$$\overline{REC} = \frac{1}{10} \sum_{q=0}^{9} \text{Re}\,cency(R) \quad (12)$$

is plotted between number of pages returned by

Altavista and LACrawler (see Figure 6). The former displays a very good recency advantage over Altavista.

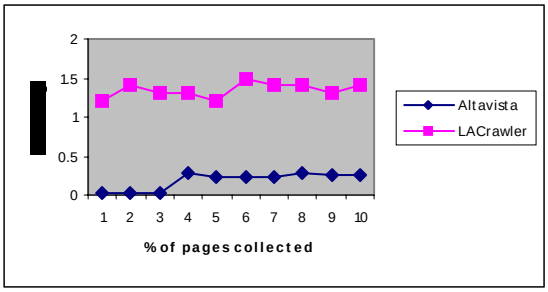


Figure 6. Estimated Recency

6. Conclusion

With the fast improvements in the computational tools, the rapidly growing web needs a novel approach. This article suggests such an approach with the agents mining the documents online. The shortcomings of traditional search engines in terms of recency, recall, precision and scalability are identified and discussed. A novel approach is suggested for using online web mining agents as complimentary tool to search engines to retrieve more relevant web pages. The performance of the system with different parameters is evaluated and the results are given empirically. The system design may be extended to include the search technique to explore the 'hidden web'. Also a collaborative crawling scheme may be employed where the parallely fired agents communicate their learning experience among themselves and to their offspring.

References:

[1] R. Armstrong, D. Freitag, T, Joachins and T. Mitchell, Webwatcher : A learning apprentice for the world wide web, *In proceedings AAAI Spring symposium on Information gathering from Heterogeneous, Distributed Environments*, 1995.

[2] K. Bharat and M. Henzinger, Improved algorithm for topic distillation in a hyperlinked environment, In proceedings of the 21st *ACM SIGIR conference on Research and Development in Information Retrieval*, 1998.

[3] C.M. Bowman, P.B. Danzig, U. Manber, M.F. Schwartz, scalable internet resource discovery: Research problems and approaches, *communications of the ACM*, 37(8), 1994, pp. 98-107.

[4] S. Brin and L.Page, The anatomy of Large scale hyperTextual web search Engine, *Proceedings of 7th World Wide Web Conference*, Elsevier Science, Amsterdam, 1998, pp. 107-117.

[5] R.B. Doorenbos, O.Etzioni and D.S. Weld, A scalable comparison shopping agent for the world wide web, *Technical report 96-01-03*, University of Washington, Department of CSE, 1996.

[6] Filippo Menczer, Complementing search engines with online web mining agents, *Decision Support Systems 992*, 2002, pp1-18.

[7] Filippo Menczer, A. Monge, Scalable web search by adaptive online agents: an Inforspiders case study, Intelligent Information Agents: Agent based Inormation Discovery and Management on the Internet, *Springer*, Berlin, 1999, pp. 337-365.

[8] Jaideep Srivatsava, B. Mobasher, R. Cooley, Web mining: Information and pattern discovery on the world wide web, *International conference on tools with Artificial Intelligence*, Newport beach, 1997, pp558-567.

[9] S. Lawrence, C. Giles, Accessibility of information on the web, *Nature 400*, 1999, pp. 107-109.

[10] S.R. Lawrence, C.L. Giles, Searching the World Wide Web, *Science 280*, 1998, pp. 98-100.

[11] Margaret H. Dunham, *Data Mining Introductory and Advanced Topics*, Pearson Education, 2003.

[12] Micheal Chau, Daniel Zeng, Hsinchun Chen, Micheal Huang, David Hendriawan, Design and evaluation of a multi agent collaborative web mining system, *Decision support systems*, Elsevier Science, 2002, pp. 1-17.

[13] Raymond S.T. Lee, James N.K.Liu, iJADE

web miner: An intelligent agent framework for Internet Shopping, *IEEE trans. on Knowledge and Data Engineering*, April 2004, pp. 401-413.

[14] Soumen Chakrabarti, ByronE.Dom, S. Ravikumar, Prabhakar Raghavan, Sridhar Rajagopalan, Andrew Tomkins, David Gibson, Jon Kleinberg, " Mining Web's Link Structure", *IEEE Computer*, 1999, pp. 60-67.