論文
4-12-22

Improving the quality of Search engine by using the Intelligent agent technolo

Ha-Nam Nguyen[1], Gyoo-Seok Choi[2], Jong-Jin Park[3], Sung-Do Chi[4],

Abstract

The dynamic nature of the World Wide Web challenges Search engines to find relevant and recent pages. Obtaining important pages rapidly can be very useful when a crawler cannot visit the entire Web in a reasonable amount of time. In this paper we study way spiders that should visit the URLs in order to obtain more "important" pages first. We define and apply several metrics, ranking formula for improvingcrawling results. The comparison between our result and Breadth-first Search (BFS) method shows the efficiency of our experiment system.

Keywords

Crawling, spider, URL ordering, intelligent agent, machine learning.

1) 정회원 : Graduate school of Hankuk Hangkong University

2) 정회원 : Graduate School of Chungwoon University

3) 정회원 : Graduate School of Chungwoon University

4) 정회원 : Graduate school of Hankuk Hangkong University

※본논문은청운대학교학술연구조성비에의해연구되었음.

논문접수 : 2003. 12. 15.
심사완료 : 2003. 12. 23.

1. Introduction

The Web has plenty of useful resources, but their characteristics make them difficult to locate. These characteristics are huge and ubiquitous, mostly semi-structured or unstructured, diverse in quality, dynamic, distributed and autonomous. Web search engine helps us to find relevant documents. However, the dynamic, irregular nature and rapid proliferation of the Web space have made increasing difficulties in up-to-date search indexes and searching the useful information on Internet. Moreover, the increment of Web space makes the information overload upon query. When using general-purpose search engine such as *Google* (www.google.com) or *Teoma-search with authority* (www.teoma.com), we receive thousands of irrelevant results. Some specific search engines try to decrease size of Web space by keeping indexes only in specific domains. Some of them are *CiteSeer* (citeseer.nj.nec.com/cs), *SciSeek* (www.sciseek.com), etc. This kind of search engines focuses on the high- quality and some other criterions such as page cite, users' rank, and so on.

The most important part of search engines, called spider or crawler, is used to collect useful data from the web space. The quality of the data set dependson which methodology they used to collect data. To deal with this, we have proposed Information retrieval system that focuses on quality of data set. We try to collect high quality of data set by using intelligent technology. In this paper, we present the method to improve quality of results by learning from the good resource and then control the crawling process by using ranking formula.

2. Backgrounds

2.1.SearchEngine

2.1.1.Concept

A "search engine" is a very broad term, and you maybe get different definitions depending on whom you ask. In this paper, we deal with a relatively generic definition: Search engine is a resource that allows you to search for information on the Internet.

In ***www.dictionary.com***, the "search engine" term is considered that

> *__A software program that searches a database and gathers and reports information that contains or is related to specified terms.*
>
> *__A website whose primary function is providing a search engine for gathering and reporting information available on the Internet or a portion of the Internet.*

2.1.2.Generalstructure

In general, a search engine system comprises a controller component, an index database, and a crawler component. The interaction of these components is illustrated in figure 1.

The controller component consists of user-interface such as input and results returned screens, and utilities for searching process such as number of results, type of result, logical combination keywords, and so on. By using the *Controller* component, the corresponding queries are built. Then, they are sent to the *Index database* component based on users' request.

The *Index database* stores information about web documents. The structure of this database depends on the target of search engines. Based on properties of URL entity stored in *index database*, the *Controller* can look up suitable documents with users' request in the *Index database*.

The *Crawler* component like soft-robot or spider can roam through the Internet. The main target of this component is to collect and update the *Index database* with all needed information. Clearly, the quality of search results depends on the quality of the collected information of the *Crawler*component. In session 2.3, we will show the direction of current research to improve crawling techniques. In this paper, we try to improve the quality of search results by using our crawling-method.

2.2. AgentandIntelligentAgent

What is an agent? *"An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors."*[1]

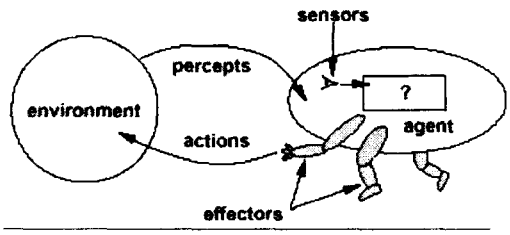


Figure 2:Agents interact with environments.

For Web searching agents, the environment is the World Wide Web for searching and the computer terminal for interacting with the user. The agent's percepts are the content of HTML documents retrieved from software sensors, which connect to the WWW (i.e., Internet) with HTTP Protocol. An agent's reasoning determines whether a web page contains targeted key words or phrases or not. If not, it goes to other pages and continues searching to complete the goal. It feed-backs to the environment by using output methods to return results to the user.

What exactly makes an agent "intelligent"is somewhat hard to define. It has been the subject of many discussions in the field of Artificial Intelligence, but a clear answer has been found yet. A workable definition of what makes an agent intelligent is given in [2]:

> *"Intelligence is the degree of reasoning and learned behavior: the agent's ability to accept the user's statement of goals and carry out the task delegated to it."*

2.3. Background on Intelligent techniques forcrawling

Crawling in Web space is a challenging task because of the performance and the reliability of issues. Most spiders use simple graph search algorithms to collect the relevant documents such as breadth-first search (BFS) [3]. Without controlling, the crawler will fetch pages for any topics. Thus, the developer must use some strategies for fetching pages. A good Crawler can improve the precision of search results by predicting whether a link points to a relevant Web page before downloading it. Current research in this area falls into two categories: content-based or link-based

2.3.1. Content-basedanalysis

In this case, spiders apply techniques for textual analysis [3][4] and keyword extraction to determine whether a page is relevant to user request. The spider can calculate the appearance of keyword in a page, and use that value to measure relevance of page. That is the right way of search engine Altavista [8] used to rank any page it collected.

2.3.2. Link-basedanalysis

Recently, Web link structure [5] is used to infer important information about pages. According to hypertext link structure, this type of research defines *authority* and *hub* terminology. All pages that are pointed to by page *u* are *authority* pages of *u*. And the other word, *hub* is alist of links to authority pages. The authority score of a page is sum of hub score of pages link to it. And conversely, the hub score is sum of corresponding authority scores of the pages, which it links to (figure 3). This method uses the number of hubs or authorities or both of them to measure the relevant of documents. The larger the score of them is, the higher a spider will rate a page.

Several link-based analysis algorithms have been developed over the past few years. The well-known and most widely used algorithms are PageRank and HIST algorithms, which are illustrated in [3][5][6][7]. The most population web search engine, Google, was developed on the basis of the PageRank algorithm [6][7].

3. Proposedapproach

In the principle of content-based method using the Vector-space model [3][4], it only computes the keyword rate to measure the relevance of any document. To rank any document using this method, we must compare the keywords rate of any document with whole set. It is impossible to rank web documents because we can't build the whole set of Web space.

In this paper, we improve this method based on some characteristics and structure of web documents. By using the intelligent agent technology [9], our system supports learning ability from the training set as well as making rational decision based on ranking formulas. The detail description of these formulas will be explained in section 3.2.

Our approach is divided into **training** part and **crawling**part. Figure 4 presents the framework of the proposed system.

3.1. Trainingstage

3.1.1. *Training-set*construction

Nowadays, there are hundreds of general web search engine available on Internet. The meta-search [10] is executed over a given query, concurrently a variety of web search engine on Internet. Then, results are merged and presented in a homogeneous ranking-based way to users.

By using meta-search approach, our system [11] builds the *Training-set* by sending queries and receiving results from some web search engines. Our system sorts results in descending order of keyword appearance in the description of links. We consider that this set has closest relationship with users' request. By analyzing data in this set, our system collects the word-set that maybe relevant to user request. It will be illustrated in the next session.

3.1.2. Trainingbyusing *Training-set*

In this session, we present how our system can learn from the training-set. Based on the structure of aHTML page, we have to analyze the content of the document for making the *Relationship word set*. The structure of HTML page consists of a header part and abody part, which is depicted in Figure 5. The header part stores the information about the document and body part stores the document's content.

Intuitively, information stored in some META-tags [12] helps search engine to improve quality of results. Here, we look for META element that definesa comma-separated list of keywords/phrases, or gives a short description [13]. Our method uses information in *keywords* and

description META-tag for learning process (Figure 6).

```
<html>
        <head>
            <meta name=""
content="">
            <title>New Page
1</title>
        </head>

        <body>


        </body>
</html>
```

Figure 5: The basic structure of HTML

<META NAME="**description**"
CONTENT="The Java Boutique is a collection of java applets, games, scripts, and tutorials. Learn programming and download ">
<META NAME="**keywords**"
CONTENT="free java applets,applet archive download source code programmer tutorials webmaster articles object oriented programming IDE reviews jini watch news java enabled browser interpreter compiler internet.com javabeans educational audio games visual effect utilities navigation menu text effect parameters, javaboutique.internet.com java.internet.com javaboutique.com">

Figure 6: META-tag structure of a page when keyword is "java"

System first downloads and analyses the HEAD part of the web documents stored in the *training-set*. There are a lot of META data in the HEAD part[12], but we only analyze data within *keywords* and *description* META-tags among them for collecting relative words and expressions. We apply the term frequency method[3][4] to make the list of words with the number of their appearance while analysingHEAD part of documents in the *training-set*. Then, we compute a ratio of appreance of collected words. The number of those words may be hundreds or thousands, but most of them may be less-related or non-related. Thus, we should choose some highest frequencies among them to construct the *Relative words set*. In here, the relative words mean the words that are related to the keywords. For example, relative words when a keyword is "java" are "applets", "java script", "jdk", etc.(see table 1).

Let n be the number of relative words and fi be the term frequency of ith word, we can compute ratio of appearance of any word to all words as following

$$\alpha_j = \frac{f_j}{\sum_{i=1}^{n} f_i} \qquad (1)$$

3.2. Intelligencecrawlingstage

In this stage, we first download pages in *training-set* and compute the term frequency of appearance keywords and relative words in these pages. While calculating term frequency, our system constructs *crawling-set* for next crawling sessions. Next, we compute theaverage of term frequency by applying term frequency method [3]. In this paper, we have defined the following formula to compute theaverage of term frequency (TFave)

$$TF_{ave} = \beta \frac{\sum_{k=1}^{m} f_k}{m} + (1-\beta)\sum_{i=1}^{n}(f_i * \alpha_i) \qquad (2)$$

where

n: number of relative words

m: number of keywords

: weight value (01)

fk: kth term frequency

fi: ith word frequency

i: relative word rate

Then, a rank value of documents can be calculated by using following formula:

$$R = \begin{cases} \frac{TF_{ave}}{Maxkey} & if \quad TF_{ave} \le Maxkey \\ 1 & if \quad TF_{ave} > Maxkey \end{cases} \qquad (3)$$

Based on this rank value, we can make a rational decision whether links are good or not. And this rank value guidescrawling process how to continue fetching more pages in next stage. This crawling process iterates and finishes until the *Result set* is filled up.

Figure 7 presents framework of crawling iteration process. Based on R value in formula 3, we decide which node is sent to the *Result-set*. In our method, this value is used to guide the crawling process. The higher R value is, the sooner page is fetched. Any node with R values equal zero and aren't exist such as P1, 8 and P2, 11 nodes in figure 7 is rejected. At the end of any iteration, the page which has highest R value among all pages is selected, and the children of itare fetched in next iteration. This process is repeated until the number of pages reach to the defined maximum value.

4. Experimentandresults

As an example, we took an experiment with some keywords relative tocomputer fields to collect results. We design and executetwo experiments for choosing the best parameters in training stage and comparing results in crawling stage.

This experiments is done under PC with 933Mhz CPU and 256Mb RAM. The software of experiments is developed by using Visual C 6.0 environment.

4.1. Experimentintrainingstage

In this experiment, we try to find optimal parameters in making the *Relative words set*. Based on the distribution of *Relative words set* results when changing parameters, we tried to compute and find out the bestparameter.

Firstly, our system builds a set of ULRs by sending queries to search engines and receiving results from them. We assume that this set of URLs contain corresponding pages to users' request. In our experiments, we created the *training set* with approximately200 independent web pages (see figure 8). We use all of those pages for looking up the relative words of the users' keyword (ex. "java").

http://java.sun.com
http://javaboutique.internet.com
http://www.javasoft.com
http://javascript.internet.com
http://www.javaworld.com
http://www.december.com/works/java.html
http://www.arcytech.org/java
http://gee.cs.oswego.edu/dl/html/javaCodingStd.html
http://www.first.gmd.de/persons/leo/java/Telnet
http://javassh.org
...

Figure 8: Training set when keyword is "java"

Relative word	Appearance raate(∂)
Applets	22.45%
Java programming	22.45%
Free	20.41%
Jdk	18.37%
Java Script	16.33%

Table 1: Example of relative words set when keyword is "java".

Next, our system downloads and analyzes header part of all pages in *training set*. As we mentioned earlier, *description* and *keyword* META-tags are the most relative with request of users. We first calculate the term frequency (fi) of all words' appearance and then calculate the appearance rates () of all words. The problem here is a number of less-relative andnon-relative words. Thus, we must cut off these words. We assume that the word with higher fi value is more relative with keywords than the others. In this test, we choose the number of *relative word set* is confine to five (n=5). Table 1 is example of relative word set when a keyword is "java". In this table, we only keep first five words and calculate their appearance rates () by using formula 1.

Generally speaking, the words in *keyword* META-tags are morerelative with users' request than ones in *description*META-tags. We introduce d/k (description/keyword) ratio for comparing the degree of importance between two parts. For example, the ratio 1/2 means that the *keyword*part is two times important than *description* part. In our experiments, we introduce error word rate (E%) for looking up the best d/k ratio through a lot of experiments. Here, we define error words are any non-relative or less-relative words in *relative word set*. And also we define E% as follows,

$$E_{\%} = \frac{\sum_{k=1}^{m} \alpha_k}{\sum_{i=1}^{n} \alpha_i + \sum_{k=1}^{m} \alpha_k} \times 100$$

where

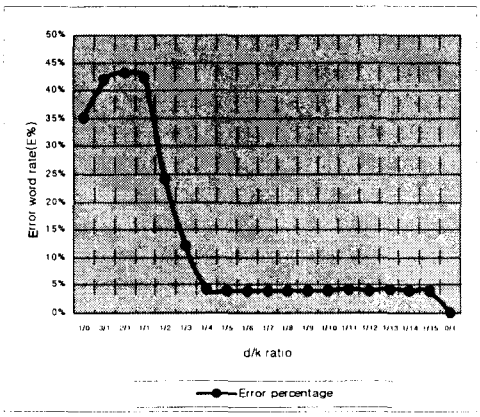
n: number of words in *relative word set*

m: number of error words

k: corresponding of error words (0km)

i: corresponding relative words (0in)

The Graph 1 shows us E%distribution according to the d/k ratio. We change the d/k ratio such as horizontal axis of Graph 1 and calculate E%. Then the ratio having lowest E% is chosen as a result.

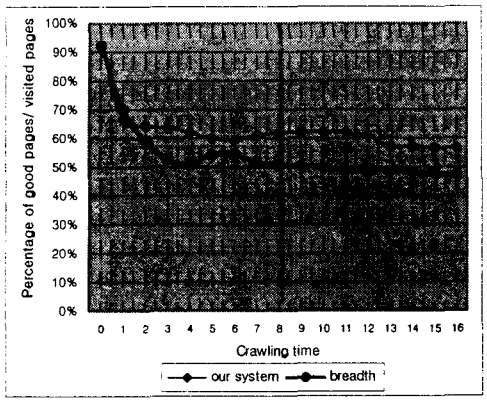


Graph 1: Histogram of error rate when keyword is "java".

By Graph 1, the error rate shows the lowest value when the d/k ratio is 0/1 and n is five. So we experientially chose the d/k ratio is 0/1 and number of relative words is five (n =5) for the further experiment.

4.2. Experiment in intelligent crawling stage

In this second experiment, we try to crawl about 10000 pages by using the proposed method (session 3.2) and Breadth First Search method. We define *good pages* and *visited pages* as new terms of our experiment. Good pages mean that they have rank value (*R* value) greater than zero. And visited pages are downloaded pages from the Internet during crawling process.



Graph 2: Percentage of good pages/visited pages when keyword is "java"

The Breadth-first search system uses a basic graph-traversal algorithm. The BFS approach assumes that URL relevant to a target contains other relevant web pages. Many commercial search engines have used this crawler technique. The BFS crawls the web pages in aqueue of URL (FIFO) [3].

Our crawling method (Figure 7) and BFS approach are very different. In the beginning of an iterative process, we build the *crawling-set* based on previous *result-set*. In here, the *result set* means set of URLs of which rank value (*R*) is already computed and greater than zero. And *crawling-set* means set of URLs that are linked by a page with the best *R* value among URLs in the current *result set*. Then, URLs *in crawling-set* are downloaded and each *R* value is calculated. Though repetition of this process the good pages are updated into the *result-set* in increasing order until the *result set* is filled up.

The Graph 2 shows us the comparison of good pages/visited pages rate between our system and BFS method through crawling iteration until the result-set reach to 10000. This ratio manifests the efficiency of crawling process.

In our experiment, we only focus on the quality (good *R* value) of whole collected pages, not quantity of them. High quality pages are pages relevant to user request (it normally has high R value). So quality is more important than quantity in web search method. We introduce some value, G%, for evaluating the performance of above two methods (BFS and our method). Here, we define G% as following

$$G_{\%} = \frac{number\ of\ good\ pages\ between\ m\ and\ n}{Whole\ good\ pages} \times 100$$

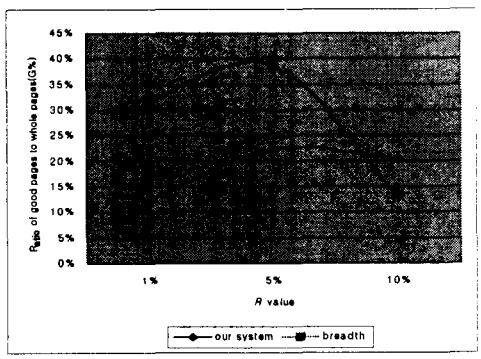
where

m, n indicate some *R* value range

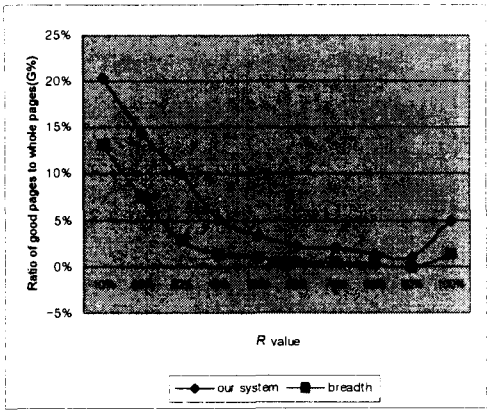
Graph 3 performs the distribution of good pages (G%) according to rank value. The horizontal axis shows *R*value range and vertical axis shows the ratio of good pages within corresponding *R* value range to entire collected pages. For example, In graph 3.a, when *R* value from 0% to 1%, we get G% are 12.62% (our system) and 32.32% (BFS).

As we mentioned earlier, the higher rank value is, the better quality of collected pages is. The Graph 3.a shows thatBFS method returns better results than our system in case of good pages with low quality (0%<R10%). Graph 3.b depicts the performance of these approaches in which good pages have high quality (R>10%). In this case, our system always returns better results than BFS method.

In almost all test cases with our method, our proposed method is better than BFS method (Graph 3) in the side of their quality.



(a)



(b)

Graph 3: Quality of result-set based on rank value with user's request is "java"

5. Conclusion

In this paper, we focus on the problem of ordering URLs for crawling.We have combined intelligent agent technique with rank algorithm to improve the performance of search engine. In other words, we are interested in not only keywords but also its relative words. Firstly, we present the way to calculate the term frequency for getting intelligent information from corresponding good pages. Secondly, we present the way to calculate *R*value for getting high quality pages as a result of web search engine. Based on the idea of link-based analysis, "a good page contains good links", Finally, our method predicts where good pages are before collecting by using *R*value. We designed a kind of lightweight crawler. Desirable results are built on the fly from receiving request to returning results. Based on the intelligent agent technique, our proposed method can learn from good pages and make a rational decision to choose better crawling direction than one in traditional BFS approach.

Our research focuses on how to make high quality results to user's requests. Therefore, our method is useful for developing assistant personal web search engine or interactive search engine to help web surfing. In the future, our work is addressed how to improve quality of search result by using heuristics.

Reference

[1] Stuart Russell, and Peter Norvig. "Artificial Intelligent - A model approach". Prentice hall, Upper Saddle River, N.J., 1995

[2] Gilbert, Aparicio, et al. "The Role of Intelligent Agents in the Information Infrastructure". IBM, United States, 1995.

[3] Soumen Chakrabarti, "Mining the Web Discovering knowledge from hypertext data", Morgan Kaufmann publishers an imprint of Elsevier Science, 2003.

[4] Bronson Trevor, Edgar Weippl, and Werner Winiwarter, "A Modern Approach to Searching the World Wide Web: Ranking Pages by Inference over Content", 2001.

[5] Allan Borodin, Gareth O. Roberts, Jeffrey S. Rosenthal, Panayiotis Tsaparas, "Finding Authorities and Hubs From Link Structures on the World Wide Web World Wide Web", 2001.

[6] S. Brin and L.Page, "The anatomy of Large-scale Hypertextual Web search engine", Proc. 7th WWW Conf. 1998; *www7.scu.edu.au/programme/fullpapers/1921/com1921.htm.*

[7] Jon M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment", Proc. 9th ACM-SIAM Symp. Discrete algorithms, Journal of the ACM, 1998.

[8] Dirk van Eylen, "Dirk van Eylen: AltaVista ranking of query results", 01/06/2002; *users.skynet.be/fa215763/avrank.html*

[9] Gilbert, Aparicio, et al. "The Role of Intelligent Agents in the Information Infrastructure". IBM, United States, 1995.

[10] Erik Selberg, Oren Etzioni, "Multi-Service Search and Comparison Using the MetaCrawler", Proceedings of the 4th International World-Wide Web

Conference, Oct. 9, 1995.

[11] Ha-Nam Nguyen, Gyoo-Seok Choi, Jong-Jin Park, "WebSearcher: A study on development of Information Retrieval system using Intelligent Agents technology", the 18th KIPS conference, Nov., 2002.

[12] "The global structure of an HTML document" *www.w3.org/TR/html4/struct/global.html*