EVOLVING STRATEGIES FOR WEB CRAWLER

KAMİL KÜÇÜK

IŞIK UNIVERSITY

2009

EVOLVING STRATEGIES FOR WEB CRAWLER

KAMİL KÜÇÜK

B.S., Information Technology, Işık University, 2009

Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

in

Information Technology

IŞIK UNIVERSITY

2009

IŞIK UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

EVOLVING STRATEGIES FOR WEB CRAWLER

KAMİL KÜÇÜK

APPROVED BY:

Assist. Prof. N. Ziya PERDAHÇI	Işık University	_____
Assist. Prof. Vedat Coşkun	Işık University	_____
Assoc. Prof. Seyhun Altunbay	Işık University	_____

APPROVAL DATE : 6/May/2009

EVOLVING STRATEGIES FOR WEB CRAWLER

Abstract

With the rapid growth of Internet and Internet-based information, it becomes the largest and publicly accessible data source in the world. Every day millions of information available so to achieve information becomes harder. To get the correct information trusted web sites and search engines are used. Trusted web sites have links between themselves, and users can reach correct and relevant information. Search engines are using crawler to follow links between pages. The context available to such crawlers can guide the navigation of links with the goal of efficiently locating highly relevant target pages. Crawler takes seed pages from search engines and follows these links using multi-agents. After first search, the results are inserted to database and they are used for seed pages for another search. The aim is the get access more reliable information using more seed pages in a short time.

EVOLVING STRATEGIES FOR WEB CRAWLER

Özet

İnternet ve internet temelli bilgilerin süratli büyümesi, internet dünyada en fazla kullanılan kaynak haline getirmiştir. Her gün milyonlarca bilginin girmesi ile büyüyen internette bilgiye ulaşmakta zorlaşmıştır. Doğru bilgiye ulaşmak için güvenilen siteler veya arama motorları kullanılmakatdır. İnternette güvenilen sayfalar birbiri arasında bağ oluşturarak, kullanıcıların doğru ve ölçeklenebilir bilgiye ulaşmasını sağlamaktadır. Arama motorlarında alınan başlangıç sayfalarında bulunan linkleri çoklu ajanlar kullanarak takip edilmiş ve ölçeklendirilmeye çalışılmıştır.İlk arama yapıldıktan sonra bunlar veritabanına kaydedilmiş başka aramalar için başlangıç sayfası olarak kullanılmıştır. Böylece daha once ulaşılan bilgiye daha kısa sürede ulaşmak, daha fazla sayfa üzerinde arama yapmak ve daha güvenilir bilgiye ulaşmak amaçlanmıştır.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

API	Application Programming Interface
CPU	Central Processing Unit
DNS	Domain Name System
DOM	Document Object Model
FIFO	First In First Out
HITS	Hyperlink-Induced Topic Search
HTML	Hypertext mark-up Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol Address
IR	Information Retrieval
ISP	Internet Service Provider
kNN	K-Nearest Neighbor
MD5	Message-Digest Algorithm 5
OLAP	Online Analytical Processing
SVM	Support Vector Machine
TCP/IP	Transmission Control Protocol / Internet Protocol
TF/IDF	Term Frequency/ Inverse Document Frequency
URL	Uniform Resource Locator
WWW	World Wide Web

1 INTRODUCTION

World Wide Web is the largest publicly accessible data source in the world. It changes the ways of doing business, providing and receiving education, managing the organization etc. The most direct effect is the completed change of information collection, conveying, and exchange. The Web has many unique characteristics, which make mining useful information and knowledge a fascinating and challenging task.

All these characteristics of Web present challenges and opportunities for mining and discovery of information and knowledge from the Web. Web Mining is a technique to discover and analyze the useful information and knowledge from the Web data e.g. Web hyperlink structure, pages content and usage data. The Web Mining research relates to several research communities such as data mining, database, information retrieval and artificial intelligence. The most recognized approach is to categorize Web Mining into three areas: Web structure mining, Web content mining and Web usage mining [1].

Web Structure Mining: The goal of web structure mining is to generate structural summary about the web site and Web page. It discovers useful information and knowledge from hyperlinks which represent the structure of the web.

Web Content Mining: Web content mining extracts or mines useful information and knowledge from web page contents. Web content mining essentially is an analog of data mining techniques from relational databases, since it is possible to find similar types of knowledge from the unstructured data.

Web Usage Mining: The aim of the Web usage mining is discover the useful information from the secondary data derived from the interactions of the users while surfing the Web. It focuses on the techniques that could predict user behavior while

the user interacts with Web. It uses web usage logs, which record every click made by the each user.

The Web Mining process is similar to the data mining process. But, there are some differences in data collection. In data mining the data is already collected in the data ware house, on the other hand, in the web mining data can be collected at the server-side, client-side, and proxy servers or obtained from an organization's database.

Web Crawlers are used for downloading web pages. They fetch the web page, parse the web content and pick up new links. They follow these links to get access more reliable information. There are some crawler algorithms in literature. All of them are multi-threaded. The relevance of a page is changed by user. However, crawlers use relevance score for similarity to the given query.

2 WEB MINING

World Wide Web is growing rapidly in the last decade, so the World Wide Web is the largest publicly accessible data source in the world. It changes the ways of doing business, providing and receiving education, managing the organization etc. The most direct effect is the completed change of information collection, conveying, and exchange. The Web has many unique characteristics, which make mining useful information and knowledge a fascinating and challenging task.

On the web, the amount of data / information is huge and continues growing. The information coverage is wide and diverse. Someone can find information on the Web. Data of all types exists on the Web e.g. structured tables, semi-structured Web pages, unstructured texts, and multimedia files (images, audios, and videos). Multiple pages may present the same or similar information using completely different words and/or formats, so information on the Web is heterogeneous. A significant of information on the Web is linked. Hyperlinks exist among Web pages within a site and across different sites within a site. Hyperlinks serve as information organization mechanism. Across different sites, hyperlinks represent implicit conveyance of authority to target pages. The information on the Web is noisy. The noise comes from some sources e.g. navigations links, advertisements, copyright notices, privacy policies and spamming. The Web is dynamic. The information on the Web changes constantly. Keeping up with the change and monitoring the change are important issues for many applications.

All these characteristics of Web present challenges and opportunities for mining and discovery of information and knowledge from the Web. Web Mining is a technique to discover and analyze the useful information and knowledge from the Web data e.g. Web hyperlink structure, pages content and usage data. The Web Mining research relates to several research communities such as data mining, database, information retrieval and artificial intelligence. The most recognized approach is to

categorize Web Mining into three areas: Web structure mining, Web content mining and Web usage mining[2].

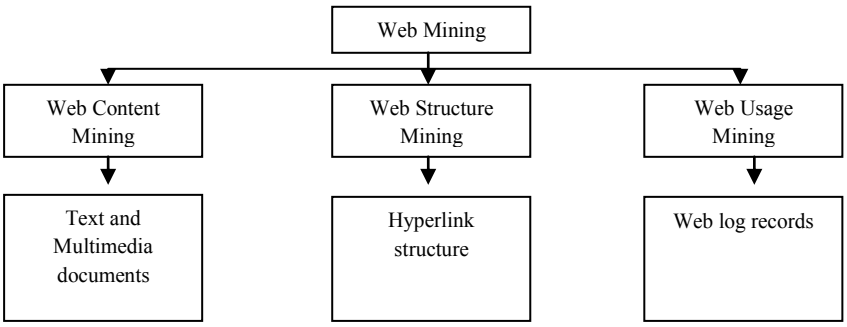

Figure 2.1 Web mining categories and objects

Web Structure Mining: The goal of web structure mining is to generate structural summary about the web site and Web page. It discovers useful information and knowledge from hyperlinks which represent the structure of the web.

Web Content Mining: Web content mining extracts or mines useful information and knowledge from web page contents. Web content mining essentially is an analog of data mining techniques from relational databases, since it is possible to find similar types of knowledge from the unstructured data.

Web Usage Mining: The aim of the Web usage mining is discover the useful information from the secondary data derived from the interactions of the users while surfing the Web. It focuses on the techniques that could predict user behavior while the user interacts with Web. It uses web usage logs, which record every click made by the each user.

The Web Mining process is similar to the data mining process. But, there are some differences in data collection. In data mining, the data is already collected in the data ware house, on the other hand, in the web mining data can be collected at the server-side, client-side, and proxy servers or obtained from an organization's database.

Once the data is collected, there are three-step processes; data pre-processing, Web data mining and post-processing.

2.1 Web mining Pre-processing

Some pre-processing tasks are usually performed before mining the web. For traditional text documents (no HTML tags), the tasks are stopword removal, stemming, and handling of digits, hyphens, punctuations, and cases of letters. For Web content mining, additional tasks such as HTML tag removal and identification of main content blocks also require careful considerations. For web usage mining, fusion & synchronization, data cleaning, pageview identification, user identification, session identification and integration of clickstreams techniques are used for pre-processing step.

2.1.1 Stopword Removal

Stopwords are frequently occurring and insignificant words in a language that help construct sentences but do not represent any content of the documents. Articles, prepositions and conjunctions and some pronouns are natural candidates. Common stopwords in English include:

a, about, an, are, as, at, be, by, for, from, how, in, is, of, on, or,
that, the, these, this, to, was, what, when, where, who, will, with

Such words should be removed before documents are indexed and stored. Stopwords in the query are also removed before retrieval is performed.

2.1.2 Stemming

In many languages, a word has various syntactical forms depending on the contexts that it is used. For example, in English, nouns have plural forms, verbs have gerund forms (by adding "*ing*"), and verbs used in the past tense are different from the present tense. These are considered as syntactic variations of the same root form. Such variations cause low recall for a retrieval system because a relevant document may contain a variation of a query word but not the exact word itself. This problem can be partially dealt with by stemming.

Stemming refers to the process of reducing words to their stems or roots. A stem is the portion of a word that is left after removing its prefixes and suffixes [2].

2.1.3 Pre-Processing Tasks for Text

Digits: Numbers and terms that contain digits are removed except some specific types, e.g., dates, times, and other pre-specified types expressed with regular expressions. However, in search engines, they are usually indexed.

Hyphens: Breaking hyphens are usually applied to deal with inconsistency of usage. For example, some people use "state-of-the-art", but others use "state of the art". If the hyphens in the first case are removed, inconsistency problem is eliminated. Thus, in general, the system can follow a general rule (e.g., removing all hyphens) and also have some exceptions. Note that there are two types of removal. First, each hyphen is replaced with a space and second, each hyphen is simply removed without leaving a space so that "state-of-the-art" may be replaced with "state of the art" or "stateoftheart".

Punctuation Marks: Punctuation can be dealt with similarly as hyphens.

Case of Letters: All the letters are usually converted to either the upper or lower case.

2.1.4 Pre-Processing Task for Web Page

Identifying different text fields: In HTML, there are different text fields, e.g., title, metadata, and body. Identifying them allows the retrieval system to treat terms in different fields differently. For example, in search engines terms that appear in the title field of a page are regarded as more important than terms that appear in other fields and are assigned higher weights because the title is usually a concise description of the page. In the body text, those emphasized terms (e.g., under header tags <h1>, <h2>, . . . bold tag <b>, etc.) are also given higher weights.

Identifying anchor text: Anchor text associated with a hyperlink is treated specially in search engines because the anchor text often represents a more accurate description of the information contained in the page pointed to by its link. In the case that the hyperlink points to an external page (not in the same site), it is especially valuable because it is a summary description of the page given by other people rather than the author/owner of the page, and is thus more trustworthy.

Removing HTML tags: The removal of HTML tags can be dealt with similarly to punctuation. One issue needs careful consideration, which affects proximity queries and phrase queries. HTML is inherently a visual presentation language.

Identifying main content blocks: A typical Web page, especially a commercial page, contains a large amount of information that is not part of the main content of the page. For example, it may contain banner ads, navigation bars, copyright notices, etc., which can lead to poor results for search and mining. Several researchers have studied the problem of identifying main content blocks. They showed that search and data mining results can be improved significantly if only the main content blocks are used.

- **Partitioning based on visual cues**: This method uses visual information to help find main content blocks in a page. Visual or rendering information of each HTML element in a page can be obtained from the Web browser. For example, Internet Explorer provides an API that can output the X and Y coordinates of each element. A machine learning model can then be built based on the location and appearance features for identifying main content blocks of pages. Of course, a large number of training examples need to be manually labeled [3].

- **Tree matching**: This method is based on the observation that in most commercial Web sites pages are generated by using some fixed templates. The method thus aims to find such hidden templates. Since HTML has a nested structure, it is thus easy to build a tag tree for each page. Tree matching of multiple pages from the same site can be performed to find such templates. Once a template is found, identifying which blocks are likely to be the main content blocks based on the following observation: the text in main content blocks are usually quite different across different pages of the same template, but the nonmain content blocks are often quite similar in different pages. To determine the text similarity of corresponding blocks (which are sub-trees), the shingle method described in the next section can be used.

2.1.5 Duplicate Detection

There are different types of duplication of pages and contents on the Web. Copying a page is usually called duplication or replication, and copying an entire site is called mirroring. Duplicate pages and mirror sites are often used to improve efficiency of browsing and file downloading worldwide due to limited bandwidth across different geographic regions and poor or unpredictable network performances. Of course, some duplicate pages are the results of plagiarism. Detecting such pages and sites can reduce the index size and improve search results. Several methods can be used to find duplicate information. The simplest method is to hash the whole document, e.g., using the MD5 algorithm, or computing an aggregated number (e.g., checksum). However, these methods are only useful for detecting exact duplicates. On the Web, one seldom finds exact duplicates. For example, even different mirror sites may have different URLs, different Web masters, different contact information, different advertisements to suit local needs, etc.

One efficient duplicate detection technique is based on n-grams (also called shingles). An *n*-gram is simply a consecutive sequence of words of a fixed window size *n*. For example, the sentence, "John went to school with his brother," can be represented with five 3-gram phrases "John went to", "went to school", "to school with", "school with his", and "with his brother". Note that 1-gram is simply the individual words.

Let $S_n(d)$ be the set of distinctive *n*-grams (or shingles) contained in document *d*. Each *n*-gram may be coded with a number or a MD5 hash (which is usually a 32-digit hexadecimal number). Given the *n*-gram representations of the two documents d_1 and d_2, $S_n(d1)$ and $S_n(d2)$, the Jaccard coefficient can be used to compute the similarity of the two documents

$$sim(d1, d2) = \frac{|Sn(d1) \cap Sn(d2)|}{|Sn(d1) \cup Sn(d2)|} \qquad (2\text{-}1)$$

A threshold is used to determine whether d_1 and d_2 are likely to be duplicates of each other. For a particular application, the window size *n* and the similarity threshold are chosen through experiments.

2.1.6 Data Fusion and Cleaning

In large-scale Web sites, it is typical that the content served to users comes from multiple Web or application servers. In some cases, multiple servers with redundant content are used to reduce the load on any particular server. Data fusion refers to the merging of log files from several Web and application servers. This may require global synchronization across these servers [4].

Data cleaning is usually site-specific, and involves tasks such as, removing extraneous references to embedded objects that may not be important for the purpose of analysis, including references to style files, graphics, or sound files. The cleaning process also may involve the removal of at least some of the data fields (e.g. number of bytes transferred or version of HTTP protocol used, etc.) that may not provide useful information in analysis or data mining tasks.

2.1.7 Pageview Identification

Identification of pageviews is heavily dependent on the intra-page structure of the site, as well as on the page contents and the underlying site domain knowledge. Recall that, conceptually, each pageview can be viewed as a collection of Web objects or resources representing a specific "user event," e.g., clicking on a link, viewing a product page, adding a product to the shopping cart. For a static single frame site, each HTML file may have a one-to-one correspondence with a pageview. However, for multi-framed sites, several files make up a given pageview. For dynamic sites, a pageview may represent a combination of static templates and content generated by application servers based on a set of parameters. In addition, it may be desirable to consider pageviews at a higher level of aggregation, where each pageview represents a collection of pages or objects, for examples, pages related to the same concept category.

2.1.8 User Identification

The analysis of Web usage does not require knowledge about a user's identity. However, it is necessary to distinguish among different users. Since a user may visit a site more than once, the server logs record multiple sessions for each user. Using the phrase of user activity record is referred to the sequence of logged activities

belonging to the same user. In the absence of authentication mechanisms, the most widespread approach to distinguishing among unique visitors is the use of client-side cookies. Not all sites, however, employ cookies, and due to privacy concerns, client-side cookies are sometimes disabled by users. IP addresses, alone, are not generally sufficient for mapping log entries onto the set of unique visitors. This is mainly due to the proliferation of ISP proxy servers which assign rotating IP addresses to clients as they browse the Web. It is not uncommon to find many log entries corresponding to a limited number of proxy server IP addresses from large ISP. Therefore, two occurrences of the same IP address (separated by a sufficient amount of time), in fact, might correspond to two different users. Without user authentication or client-side cookies, it is still possible to accurately identify unique users through a combination of IP addresses and other information such as user agents and referrers [5].

2.1.9 Sessionization

Sessionization is the process of segmenting the user activity record of each user into sessions, each representing a single visit to the site. Web sites without the benefit of additional authentication information from users and without mechanisms such as embedded session ids must rely on heuristics methods for sessionization. The goal of a sessionization heuristic is to reconstruct, from the clickstream data, the actual sequence of actions performed by one user during one visit to the site.

2.1.10 Path Completion

Another potentially important pre-processing task which is usually performed after sessionization is path completion. Client- or proxy-side caching can often result in missing access references to those pages or objects that have been cached. For instance, if a user returns to a page A during the same session, the second access to A will likely result in viewing the previously downloaded version of A that was cached on the client-side, and therefore, no request is made to the server. This results in the second reference to A not being recorded on the server logs. Missing references due to caching can be heuristically inferred through path completion which relies on the knowledge of site structure and referrer information from server logs [5]. In the case of dynamically generated pages, form-based applications using the HTTP POST

method result in all or part of the user input parameter not being appended to the URL accessed by the user (though, in the latter case, it is possible to recapture the user input through packet sniffers which listen to all incoming and outgoing TCP/IP network traffic on the server side).

2.1.11 Data Integration

The above pre-processing tasks ultimately result in a set of user sessions, each corresponding to a delimited sequence of pageviews. However, in order to provide the most effective framework for pattern discovery, data from a variety of other sources must be integrated with the preprocessed clickstream data. This is particularly the case in e-commerce applications where the integration of both user data (e.g., demographics, ratings, and purchase histories) and product attributes and categories from operational databases is critical. Such data, used in conjunction with usage data, in the mining process can allow for the discovery of important business intelligence metrics such as customer conversion ratios and lifetime values [5].

In addition to user and product data, e-commerce data includes various product-oriented events such as shopping cart changes, order and shipping information, impressions (when the user visits a page containing an item of interest), click-throughs (when the user actually clicks on an item of interest in the current page), and other basic metrics primarily used for data analysis. The successful integration of these types of data requires the creation of a site-specific "event model" based on which subsets of a user's clickstream are aggregated and mapped to specific events such as the addition of a product to the shopping cart. Generally, the integrated ecommerce data is stored in the final transaction database. To enable full featured Web analytics applications, this data is usually stored in a data warehouse called an e-commerce data mart. The e-commerce data mart is a multi-dimensional database integrating data from various sources, and at different levels of aggregation. It can provide pre-computed e-metrics along multiple dimensions, and is used as the primary data source for OLAP (Online Analytical Processing), for data visualization, and in data selection for a variety of data mining tasks[6].

2.2 Web Structure Mining

Traditional data mining tasks as association rule mining, market basket analysis and cluster analysis commonly attempt to find patterns in a dataset characterized by a collection of independent instances of a single relation. This consistent with the classical statistical inference problem of trying to identify a model given a random sample a common underlying distribution[7].

The challenging for web structure mining is to deal with the structure of the hyperlinks within the web itself. The growing interest in web mining, the research of structure analysis had increased and had resulted as a new research area called Link Mining, which is located at the intersection of the work analysis, hypertext and web mining, relational learning and inductive logic programming and graph mining.

The web contains a variety of objects with almost no unifying structure, with differences in the authoring style and content much greater than in traditional collections of text documents. The objects of World Wide Web are web pages, and links. They are in-, out-, and co-citation which two pages that are both linked to by the same page. Attributes of web include HTML tags, word appearances and anchor texts. This diversity of objects creates new problems and challenges, since is not possible to directly made use of existing techniques such as database management or information retrieval. Thus, link mining produces some agitation on some of the traditional data mining.

- Link-based classification. Link-based classification is the most recent upgrade of the classic data mining task to linked domains[8]. The task is focusing on the prediction of the category of a webpage, inks between pages, anchor text, html tags and other possible attributes found on the web page.

- Link-based Cluster Analysis: The goal in cluster analysis is to find naturally occurring sub-classes.the data is segmented into groups, where similar objects are grouped into different groups. Different than the previous task, link based cluster analysis is unsupervised and can be used to discover hidden patterns from data.

- Link Type: There are a wide range of tasks concerning the prediction of the existence of links, such as predicting the type of link between two entities, or predicting the purpose of a link.

- Link Strength: links could bee associated with weights.

- Link Cardinality: the main task here is to predict the number of links

There are many ways to use the link structure of the web to create notions of authority. The main goal is developing applications for link mining is to made good use of the understanding of these intrinsic social organization of the web.

2.2.1 HITS concept and PageRank Method

HITS and PageRank methods approach focus on the link structure of the web to find the importance of the Web pages.

2.2.1.1 HITS: Computing Hubs and Authorities

HITS concept identifies two kinds of pages from Web hyperlink information structure; authorities which are pages with good sources of content) and hubs which are pages with good sources of links.

Hubs and authorities exhibit what could be called a mutually reinforcing relationship: a good hub is a page that points to many good authorities; a good authority is a page that is pointed to by many good hubs[9].

HITS associated a non-negative authority weight $x^{<p>}$ and a non-negative hub weight $y^{<p>}$

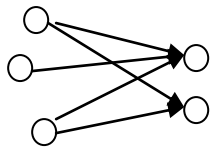

Figure 2.2 A densely linked set of Hubs and Authorities

Numerically the mutually reinforcing relationship can be expressed as follows: if *p* points to many pages with large *x*-values, then it should receive a large *y*-value; if *p*

is pointed to by many pages with large *y*-values, then it should receive a large *x*-value. Given weights $x^{<p>}$, $y^{<p>}$, then the *x*-weights and *y*-value are as follows.

$$X^{<p>} \leftarrow \sum y^{<p>} \qquad Y^{<p>} \leftarrow \sum X^{<p>} \qquad (2\text{-}2)$$

Although HITS provides good search results for a wide range of queries, HITS did not work well in some cases;

- Mutually reinforced relationships between hosts: Sometimes a set of documents on one host point to a single document on a second host, or sometimes a single document on one host point to a set of document on a second host. These situations could provide wrong definitions about a good hub or a good authority.

- Automatically generated links: Web document generated by tools often have links that are inserted by the tool.

- Non-relevant nodes: sometimes pages point to other pages with no relevance to the query topic.

```
G := set of pages
for each page p in G do
        p.auth = 1              // p.auth is the authority score of the page p
        p.hub = 1               // p.hub is the hub score of the page p
function HubsAndAuthorities(G)
        for step from 1 to k do         // run the algorithm for k steps
                for each page p in G do         // update all authority values first
                        for each page q in p.incomingNeighbors do /*
                p.incomingNeighbors is the set of pages that link to p*/
                                p.auth += q.hub
                for each page p in G do  // then update all hub values
                        for each page r in p.outgoingNeighbors do //
                p.outgoingNeighbors is the set of pages that p links to
                                p.hub += r.auth
```

Figure 2.3 HITS Algorithm [10].

2.2.1.2 PageRank Model

L Page and S. Brin proposed the Page Rank algorithm to calculate the importance of Web pages using the link structure of the web. In their approach, Brin and Page extends the idea of simply counting in-links equally, by normalizing by the number of links on a page. The Page Rank algorithm is defined as "we assume page A has

pages T_1, . . T_n which point to it (i.e. Webs are citations). The parameter *d* is a damping factor, which can be set between 0 and 1. *d* is usually set to 0,85. Also *C(A)* is defined as the number of links going out of page *A*. The PageRank of a page *A* is given as follows:

$$P(A) = (1 - d) + d(P(T1)/C(T1) + \ldots 4\, P(Tn)/C(Tn)) \quad (2\text{-}3)$$

Note that the PageRank forms a probability distribution over web pages, so the sum of all Web pages' PageRanks will be one and the d damping factor is the probability at each page the random surfer will be bored and request another random page".

Note that the rank of a page is divided evenly among its out-links to contribute to the ranks of the pages they point to. The equation is recursive, but starting with any set of ranks and iterating the computation until it converges may compute it. PageRank can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web. PageRank algorithm needs a few hours to calculate the rank of millions of pages[11].

2.2.1.3 HITS Concept and PageRank Method Applications

HITS is used for the first time in the Clever search engine from IBM, and PageRank is used by Google [12] combined with other several features such as anchor text, IR measures and proximity.

The notion of authoritativeness comes from the idea that locating a set of relevant pages, but rather the relevant pages of the highest quality. However, the Web consists not only of pages but also of links that connect one page to another. This structure contains a large amount of information that should be exploited.

PageRank and HITS belong to a class of ranking algorithms, where the scores can be computed as a fixed point of a linear equation. HITS and PageRank are used as starting points for new solutions, and there are some extensions of these two approaches. There are other link-based approaches to be applied on the Web [13].

Besides being used for weighting Web pages, link resource can also be used for clustering or classifying Web pages. The principle is based on the assumption that (1) if page *p1* has a link to page *p2*, *p1* should be similar to *p2* in content, and (2) if

p1 and *p2* are co-cited by some common pages, *p1* and *p2* should also similar. Web pages can be clustered into a lot of connected page communities with respect to their citation and co-citation strengths among the pages. uses links can be categorized as [14];

- Relevant Linkage Principle: Links points to relevant resources.
- Topical Unity Principle: Documents often co-cited are related, as are those with extensive bibliographic overlap.
- Lexical Affinity Principle: Proximity of text and links within a page is a measure of the relevance of one to another. Even though those link algorithms can always provide a good support for Web information retrievals, clustering and knowledge discoveries on the Web, authors also find problems associated with those technologies.

The original PangeRank algorithm, introduced by Page, pre-compute ranking vector based on all the Web pages. This ranking vector is computed once and used for any queries later. The ranking is actually independent of the specific queries when using it. The authors tried to solve this limitation by computing a set of PageRank vectors, each biased with a different topic. In other words, for each topic, they assigned a weight for each page. Therefore, searches in different topics could select corresponding vectors for ranking.

2.3 Web Content Mining

The heterogeneity and the lack of structure that permeates much of the ever expanding information sources on the World Wide Web, such as hypertext documents, makes automated discovery, organization, and management of web-based information difficult. Traditional search and indexing tools of the Internet and the World Wide Web such as Lycos, Alta Vista, WebCrawler, MetaCrawler, and others provide some comfort to users, but they do not generally provide structural information nor categorize, filter, or interpret documents. In recent years these factors have prompted researchers to develop more intelligent tools for information retrieval, such as intelligent web agents, as well as to extend database and data mining techniques to provide a higher level of organization for semi-structured data available on the web. The agent-based approach to web mining involves the

development of sophisticated artificial intelligent systems that can act autonomously or semi-autonomously on behalf of a particular user, to discover and organize web-based information.

The web content mining is differentiated from two different points of view[15]. Information Retrieval View and Database View is summarized the research works done for unstructured data and semi-structured data from information retrieval view. It shows that most of the researches use bag of words, which is based on the statistics about single words in isolation, to represent unstructured text and take single word found in the training corpus as features. For the semi-structured data, all the works utilize the HTML structures inside the documents and some utilized the hyperlink structure between the documents for document representation. As for the database view, in order to have the better information management and querying on the web, the mining always tries to infer the structure of the web site to transform a web site to become a database [16].

There are several ways to represent documents; vector space model is typically used. The documents constitute the whole vector space. If a term *t* occurs *n(D, t)* in document *D* , the *t*-th coordinate of *D* is *n*(*D*, *t*) . When the length of the words in a document goes to ∞, ||*D*||= $\max_t$ *n*(*D*, *t)*. This representation does not realize the importance of the words in a document. To resolve this, TFIDF (Term Frequency Inverse Document Frequency is introduced; the *t*-th coordinate of D can be represented as:

$$\frac{n(D,t)}{\sum_t n(D,t)} \times IDF(t) \qquad (2\text{-}4)$$

Where, *IDF*(*t*) = 1+ log(*DF*(*t*) / *N*) and *DF*(*t*) means that *t* occurs in *DF*(*t*) out of *N* documents.

By multi-scanning the document, feature selection can be implemented. Under the condition that the category result is rarely affected, the extraction of feature subset is needed. The general algorithm is to construct an evaluating function to evaluate the features. As feature set, Information Gain, Cross Entropy, Mutual Information, and Odds Ratio are usually used.

The classifier and pattern analysis methods of text data mining are very similar to traditional data mining techniques. The usual evaluative merits are Classification Accuracy, Precision, Recall and Information Score.

Before text mining, it is basically needed to identify the code standard of the HTML documents and transform it into inner code, then use other data mining techniques to find useful knowledge and patterns.

2.4 Web Usage Mining

Web usage mining tries to discover the useful information from the secondary data that derived from the interactions of the users while surfing on the Web. It focuses on the techniques that could predict user behavior while the user interacts with Web. The potential strategic aims are in each domain into mining goal as: prediction of the user's behavior within the site, comparison between expected and actual Web site usage, adjustment of the Web site to the interests of its users. There are no definite distinctions between the Web usage mining and other two categories. In the process of data preparation of Web usage mining, the Web content and Web site topology will be used as the information sources, which interacts Web usage mining with the Web content mining and Web structure mining. Moreover, the clustering in the process of pattern discovery is a bridge to Web content and structure mining from usage mining[17].

There are lots of works have been done in the IR, Database, Intelligent Agents and Topology, which provide a sound foundation for the Web content mining, Web structure mining. Web usage mining is a relative new research area, and gains more and more attentions in recent years.

2.4.1 The Usage Mining on the Web

Web usage mining is the application of data mining techniques to discover usage patterns from Web data, in order to understand and better serve the needs of Web-based applications [18].

The Web usage mining is parsed into three distinctive phases; preprocessing, pattern discovery, and pattern analysis.

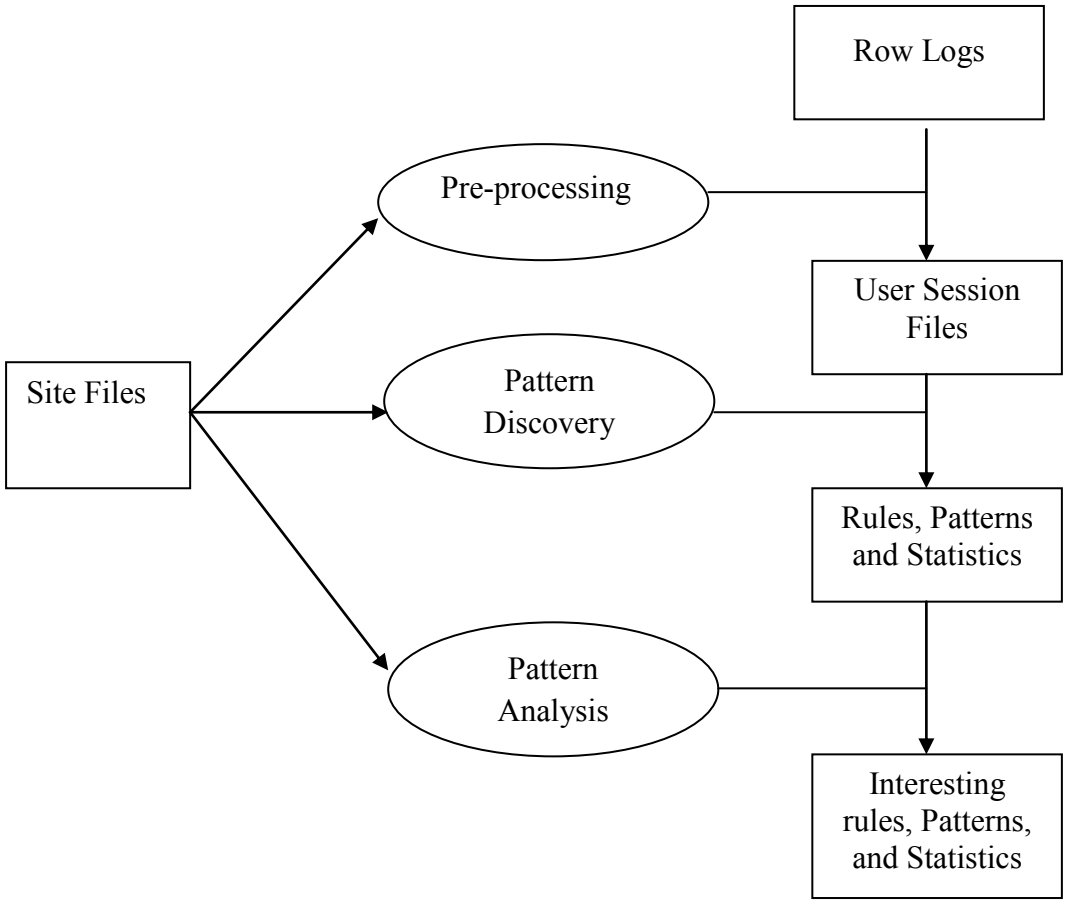

Figure 2.4 The subtask of Web Usage Mining

The purpose of Web usage mining is to reveal the knowledge hidden in the log files of a web server. By applying statistical and data mining methods to the Web log data, interesting patterns concerning the users' navigational behavior can be identified, such as user and page clusters, as well as possible correlations between web pages and user groups.

Each access to a Web page is recorded in the access log of the Web server that hosts it. The entries of a log file consist of fields that follow a predefined format. The fields of common log format are;

remotehost rfc931 authuser date "request" status bytes

where

- remotehost is the remote hostname or IP number if the DNS hostname is not available
- rfc931, the remote log name of the user

- authuser, the username with which user has authenticated himself/herself, available when using password-protected World Wide Web pages
- date, the date and time of the request
- "request", the request line exactly as it came from client (the file, the name, and the method used to retrieve it)
- status, the HTTP status code returned to the client, indicating whether the file was successfully retrieved and if not, what error message was returned
- bytes, the content-length of the documents transferred.

2.4.2 Preprocessing

Web log data is prepared and preprocessed in order to use them in the consequent phases of the process.

Web log data may need to be cleaned from entries involving pages that returned an error or graphics file accesses. In some cases such information might be useful, but in others such data should be eliminated from a log file. Furthermore, crawler activity can be filtered out, because such entries do not provide useful information about the site's usability.

Another problem to be met has to do with caching. Accesses to cached pages are not recorded in the Web log, therefore such information is missed. Caching is heavily dependent on the client-side technologies used and therefore cannot be dealt with easily. In such cases, cached pages can usually be inferred using the referring information from the logs.

Moreover, a useful aspect is to perform pageview identification, determining which page file accesses contribute to a single pageview.

Most important of all is the user identification issue. There are several ways to identify individual visitors. The most obvious solution is to assume that each IP address (or each IP address/client agent pair) identifies a single visitor. Nonetheless, this is not very accurate because, for example, a visitor may access the Web from different computers, or many users may use the same IP address (if a proxy is used). A further assumption can then be made, that consecutive accesses from the same host during a certain time interval come from the same user. More accurate approaches

for a priori identification of unique visitors are the use of cookies or similar mechanisms or the requirement for user registration. However, a potential problem in using such methods might be the reluctance of users to share personal information.

Assuming a user is identified, the next step is to perform session identification, by dividing the clickstream of each user into sessions. The usual solution in this case is to set a minimum timeout and assume that consecutive accesses within it belong to the same session, or set a maximum timeout, where two consecutive accesses that exceed it belong to different sessions [19].

2.4.3 Web Usage Mining Log Analysis

Log analysis tools which can be called as traffic analysis tools take as input raw Web data and process them in order to extract statistical information. Such information includes statistics for the site activity (e.g. as total number of visits, average number of hits, successful/failed/redirected/cached hits, average view time, and average length of a path through a site), diagnostic statistics (e.g. server errors, and page not found errors), server statistics (e.g. top pages visited, entry/exit pages, and single access pages), referrers statistics (e.g. top referring sites, search engines, and keywords), user demographics (e.g. top geographical location, and most active countries/cities/organizations), client statistics (visitor's Web browser, operating system, and cookies), and so on. Some tools also perform clickstream analysis, which refers to identifying paths through the site followed by individual visitors by grouping together consecutive hits from the same IP, or include limited low-level error analysis, such as detecting unauthorized entry points or finding the most common invalid URL. These statistics are usually output to reports and can also be displayed as diagrams.

This information is used by administrators for improving the system performance, facilitating the site modification task, and providing support for marketing decisions[20] However, most advanced Web mining systems further process this information to extract more complex observations that convey knowledge, utilizing data mining techniques such as association rules and sequential pattern discovery, clustering, and classification.

2.4.4 Tools for Web Usage Mining

Table 2.1 Web usage mining research projects and products

	Application	Data Source			Data Type				User		Site	
Project	Focus	Server	Proxy	Client	Structure	Content	Usage	Profile	Single	Multi	Single	Multi
WebSIFT()	General	X			X	X	X			X	X	
SpeedTracer	General	X					X			X	X	
WUM	General	X			X		X			X	X	
Shahabi	General			X	X		X			X	X	
Site Helper	Personalization	X				X	X		X		X	
Letizia	Personalization			X		X	X		X		X	
Web Watcher	Personalization		X			X	X	X		X		X
Krishnapuram	Personalization	X					X			X		X
Analog	Personalization	X					X			X	X	
Mobasher	Personalization	X			X		X			X	X	
Tuzhilin	Business	X					X			X	X	
SurfAid	Business	X				X	X			X	X	
Buchner	Business	X					X	X		X	X	
WebTrends	Business	X					X			X	X	
WebLogMiner	Business	X					X			X	X	
PageGather	Site Modification	X			X	X	X			X	X	
Manley	Characterization	X				X	X			X		X
Arlitt	Characterization	X				X	X			X		X
Pitkow	Characterization	X		X		X	X			X		X
Almeida	Characterization	X					X			X		X
Rexford	System Improve.	X	X				X			X	X	
Schester	System Improve.	X					X			X	X	
Aggrarwal	System Improve.		X				X			X	X	

3 INTELLIGENT AGENT

An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment, in order to meet its design objective [21].

However, there is no single all-encompassing meaning for agent. The widely accepted definition is “an agent acts on behalf of someone else, after having been authorized”. This can be applied to the software agents; they are instantiated and act instead of a user or a software program that controls them.

An agent needs some functional characteristics [22]. These characteristics are

- Autonomy: an agent may act without constant surveillance and direct human (or other) intervention and may have some kind of control over its actions and internal states, based on its beliefs, desires, and interactions. Autonomy is considered a prerequisite by many researches.
- Interactivity: An agent may interact with its environment and other entities. Interactivity entails all the behavioral aspects that an agent may exhibit within an environment. Two are the most dominant features that univocally determine an agent behavior's state:
    - Reactivity: The ability to perceive the environment and its changes, and to respond to them, when necessary.
    - Pro-activeness: The ability to take initiative and act in such a way to satisfy the goal the agent has been deployed for.
- Adaptability: An agent may perceive the existence of other agents within its "sight" or "neighborhood". Advanced adaptability schemes allow agents to alter their internal states based on their experience and their environment.
- Sociability: It is of great importance for agents to develop a behavior model that resembles aspects of human social life, such as companionship,

friendship and affability. Agent interaction is established via some kind of communication language.

- Cooperativity: The ability to collaborate in order to accomplish a common goal. This mode of operation is highly desirable in communities of agents seeking the same "holy grail".
- Competitiveness: An agent may compete with another agent, or deny service provision to an agent perceived as an opponent. This behavior is required in situations where only one agent can reach the target (i.e., electronic auctions).
- Temporal continuity: Since an agent functions continuously, proper design to ensure robustness is essential.
- Character: An agent may exhibit human-centered characteristics, such as personality and emotional state.
- Mobility: The ability to transit from one environment to another, without interrupting its functioning.
- Learning: An agent should be able to learn (get trained) through its reactions and its interaction with the environment. The more efficient the training process, the more intelligent the agents.

Intelligent agents are computational entities capable of autonomously achieving goals by executing needed actions. Intelligent agents have been widely used in Web applications. Web search engines known as crawler or spiders that run by web search engines, are used to fetch Web pages from Web servers. Another type of Web intelligent agent resides on user machines, helps a user search the web and perform personalized information and filtering system.

In web, some agents (e.g. WebSecker, and Copernic 2001[23]) connect various search engines, monitor Web pages for any changes, and schedule automatic searches. Some of them (e.g. Excalibur RetrievalWare and Internet Spider) collect, monitor, and index information from text documents on the Web. Some crawler (e.g. Focused Crawler) locates

Software agents can work autonomously and observe and learn from user actions. Agent technology has been applied information filtering and reduce information overload.

3.1 Search agents

The World Wide Web is experiencing an exponential growth both in number and in size. This enormous growth of the World Wide Web has made it important resource discovery efficiently. Users may often wish to search or index collections of documents based on topical or keyword queries. The standard method for searching and querying on such engines has been collect a large aggregate collection of documents and then provide methods for querying them. Such a strategy runs into problems of scale, since there are over a billion documents on the web and the web continues to grow at a pace of about a million documents a day. This results in problems of scalability both in terms of storage and performance.

Web search is a difficult task. There are a lot of machine learning work is being applied to one part of the web search, namely ranking indexed pages by their estimated relevance with respect to user queries. It is crucial task because it influences the perceived effectiveness of a search engine. Early search engines ranked pages principally based on their lexical similarity to the query. The key strategy was to devise the best weighting algorithm to represent Web pages, and queries in a vector space, such that closeness in such a space would be correlated semantic relevance.

Search engines are powered by state of the art indexing algorithms, which make use of automated programs, called robots or crawlers, to continuously visit large parts of the web in an exhaustive fashion. Web crawlers, also known as spiders, robots, spiders, wanderers, fish, and worms. They are programs that exploit the graph structure of the web to move from page to page. Since information on the web is scattered among billions of pages served by millions of servers around the globe, users who browse the web can follow hyperlinks to access information, virtually moving from one page to the next. A crawler can visit many sites to collect information that can be analyzed and mined in a central location, either online or offline.

There are many applications for web crawlers. One is the business intelligence, whereby organizations collect information about their competitors and potential collaborators. Another use of the web crawlers is to monitor web pages and sites of

interest, so that a user or a community can be notified when new information appears in certain places. Also, web crawlers can be used for spammers or collect personal information to be used in phishing and other identity theft attacks. However, the most widespread use of the web crawlers is in support of search engines. They collect pages for search engines to build their indexes, so they are the main consumers of internet bandwidth.

3.1.1 A Basic Crawler Algorithm

A crawler starts from a set of seed pages and then uses the external links within them to attend to other pages. The links in these pages are, in turn, extracted and corresponding pages are visited. The process repeats with the new pages offering more external links to follow, until a sufficient number of pages are identified or some higher level objectives are achieved.

A web crawler maintains a list of unvisited external links. They are called frontier. The list is initialized with the seed links which may be provided by the user or another program. In each crawling loop, the crawler picks the next link from the frontier, fetches the page corresponding to the link through HTTP, parses the retrieved page to extract its links, adds newly discovered links to the frontier, and stores the page (or other extracted information, possibly index terms) in a local disk repository. Before the links are added to the frontier they may be assigned a score that represents the estimated benefit of visiting the page corresponding to the link. The crawler process ends if the frontier becomes empty, and a certain number of pages have been crawled.

A crawler is a graph search. The web can be seen as a large graph with pages as its nodes and hyperlinks as its edges. A crawler starts at a few of the nodes (seed pages) and then follows the edges to reach other nodes. The process of fetching a page and extracting the links within it is analogous to expanding a node in graph search [24].

The frontier is the main data structure, within contains the links of unvisited pages. In graph search terminology the frontier is an open list of unexpanded (unvisited) nodes. Typical crawlers attempt to store the frontier in the main memory for efficiency. The crawler algorithm must specify the order in which new links are extracted from the frontier to be visited.

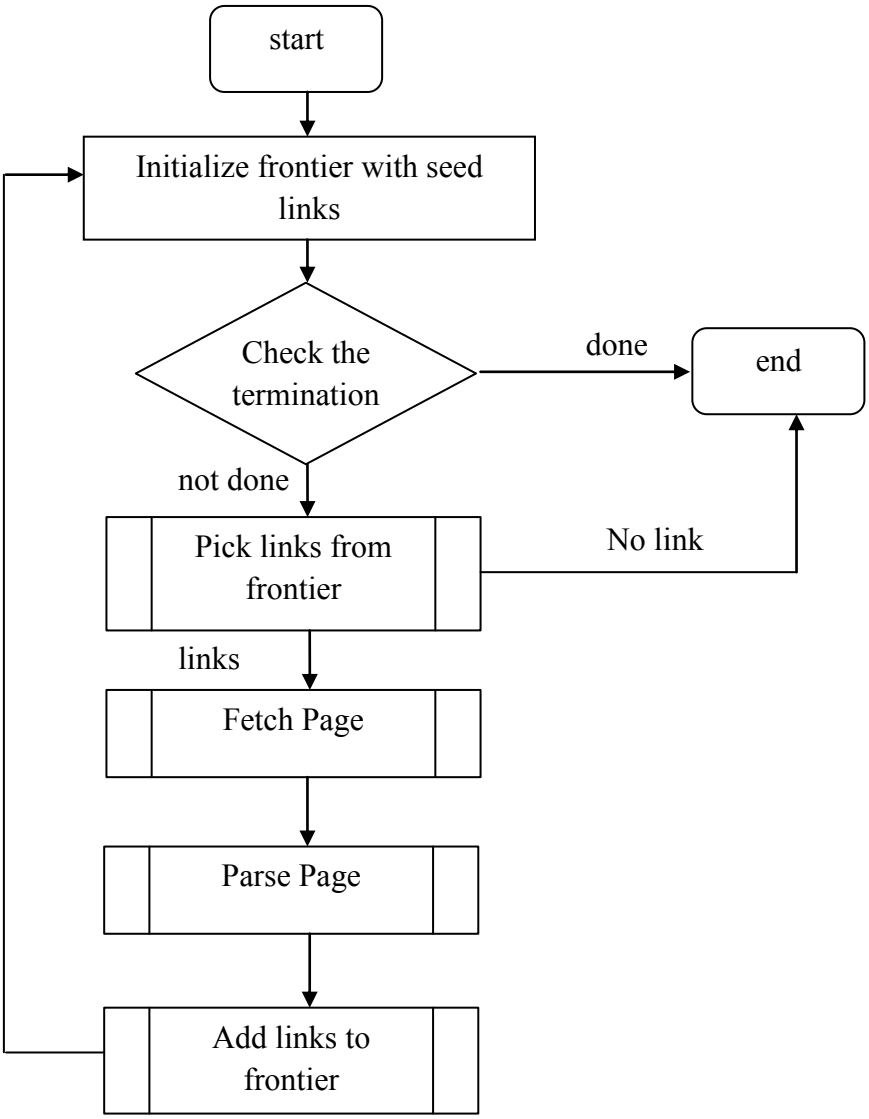

Figure 3.1 Flow of a basic sequential crawler [24]

The frontier may be implemented as a FIFO queue, so the links to crawl next comes from the head of the queue and the new links are added to the tail of the queues. Adding links to the tail is important. Due to the limited size of the frontier, there will be duplicated links. A linear search to find out if a newly extracted link is already in the frontier is costly. Store each of the frontier links as a separate hash-table. The links are as the key for hash table.

The crawl history is a time-stamped list of links that were fetched by the crawler. A link is entered into the history only after the corresponding page is fetched. This history may be used for post-crawl analysis and evaluation. If the most relevant and important resources are found in early in the crawl process, history may be stored on disk, it is also maintained as an in-memory data structure for fast look-up, to check

whether a page has been crawled or not crawled. The check operation is required to avoid revisiting pages or wasting space in the limited-size frontier. Once a page is fetched, it may be stored or indexed for the master application. In the simplest for a page repository may store the crawled pages as a separate files, and each page must map to a unique file name. A hash-table is appropriate to obtain quick link insertion and look-up times.

In the fetching step, a crawler acts as a web client. A crawler sends an HTTP request to the server hosting the page and reads the response. The client needs to timeout connections to prevent spending unnecessary time waiting for responses from slow servers or reading huge pages. The client parses the response headers for status codes and redirections. Redirect loops are to be detected and broken by storing links from a redirection chain in a hash table and halting if the same link is encountered twice.

Once a page is downloaded, the crawler parses its content to extract information that will feed and possibly guide the future path of the crawler. A crawler parses simple hyperlink or link extraction, or more complex process of tidying up the HTML content in order to analyze the HTML tag tree.

HTML parses provide the functionality to identify tags and associated attribute-value pairs in a given web page. Extracting hyperlink URLs from a page, find anchor tags and grab the values of associated href attributes. However, these relative links should be converted to absolute links using the base URL of the page from where they were retrieved.

- The protocol and hostname should be converted to lowercase
- The ‘anchor’ or ‘reference’ part of the URL should be removed.
- Some commonly used characters should be performed URL encoding
- Trailing ‘/’s should be added.
- Recognize default web pages by using heuristics.
- ‘..’ should be removed to current or parent directory
- The port numbers should be removed

Table 3.1 Some transformations to convert URLs to a canonical form

Description and Transformation	**Examples**
Mixed/upper-case host names	http://IT.ISIKUN.EDU.TR
Lower-case	http://it.isikun.edu.tr/
Fragment	http://it.isikun.edu.tr/sss.html#3
Remove	http://it.isikun.edu.tr/sss.html
Needlesly encoded characters	http://it.isikun.edu.tr/%staj
Decode	http://it.isikun.edu.tr/~staj/
Guessed Directory	http://it.isikun.edu.tr
Add trailing slash	http://it.isikun.edu.tr/
Default file name	http://it.isikun.edu.tr/index.html
Remove	http://it.isikun.edu.tr/
Current or parent directory	http://it.isikun.edu.tr/staj/../../staj/
Resolve path	http://it.isikun.edu.tr/staj/staj/
Default port number	http://it.isikun.edu.tr:80/
Remove	http://it.isikun.edu.tr/

When parsing a web page to extract content information, removing commonly used words or stopwords is helpful. The removing stopwords fro the text is called stoplisting. Also one may stem the words found in the page. The stemming process normalizes words by conflating a number of morphologically similar words to a single root form or stem.

Crawlers may assess the value of a link or a content word by examining the HTML tag context in which it resides. For this reason, a crawler needs to utilize the tag tree or DOM structure of the HTML page [25].

```
<html>
<head>
<title>Projects</title>
</head>
<body>
<h2> My Projects </h2>
<img src=”design.jpg” align=”left” alt=”My Designs”>
<ul>
<li><a href=”project1.html”>Project 1 </a></li>
<li><a href=”project2.html”>Project 2 </a></li>
</ul>
</body>
</html>
```

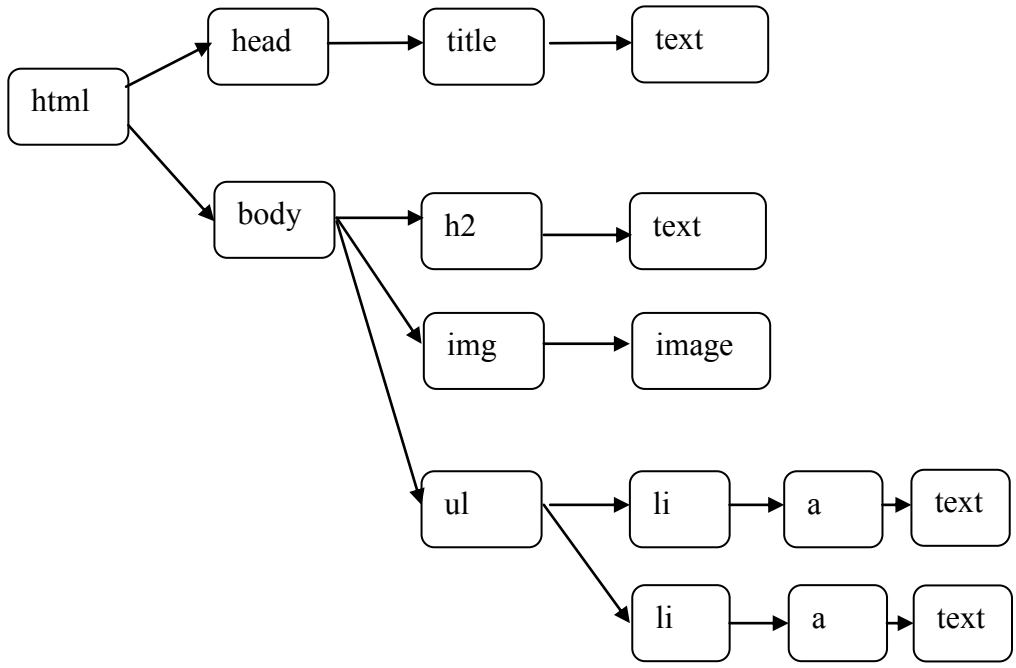

Figure 3.2 An HTML page and the corresponding tree

3.1.2 Multi-threaded Crawlers

A sequential crawling loop spends a large amount of time in which either the CPU or the network interface is idle. Multi-threading, where each threat follows a crawling loop, an provide reasonable speed up and efficient use of available bandwidth.

Each thread starts by locking the frontier to pick the next URL to crawl. After picking a URL it unlocks the frontier allowing other threads to access it. The frontier is again locked when new URLs are added to it. The locking steps are necessary in order to synchronize the use of the frontier that is shared among many crawling

loops. A typical crawler would also maintain a shared history data structure for a fast lookup of URLs that have been crawled.

The multi-threaded crawler model needs to deal with an empty frontier just like a sequential crawler. If a thread finds the frontier empty, it does not automatically mean that the crawler as a whole has reached a dead-end. It is possible that other threads are fetching pages and may add new URLs in the near future. One way to deal with the situation is by sending a thread to a sleep state when it sees an empty frontier. When the tread wakes up, it checks again for URLs. A global monitor keeps track of the number of threads currently sleeping. Only when all the threads are in the sleep state does the crawling process stop.

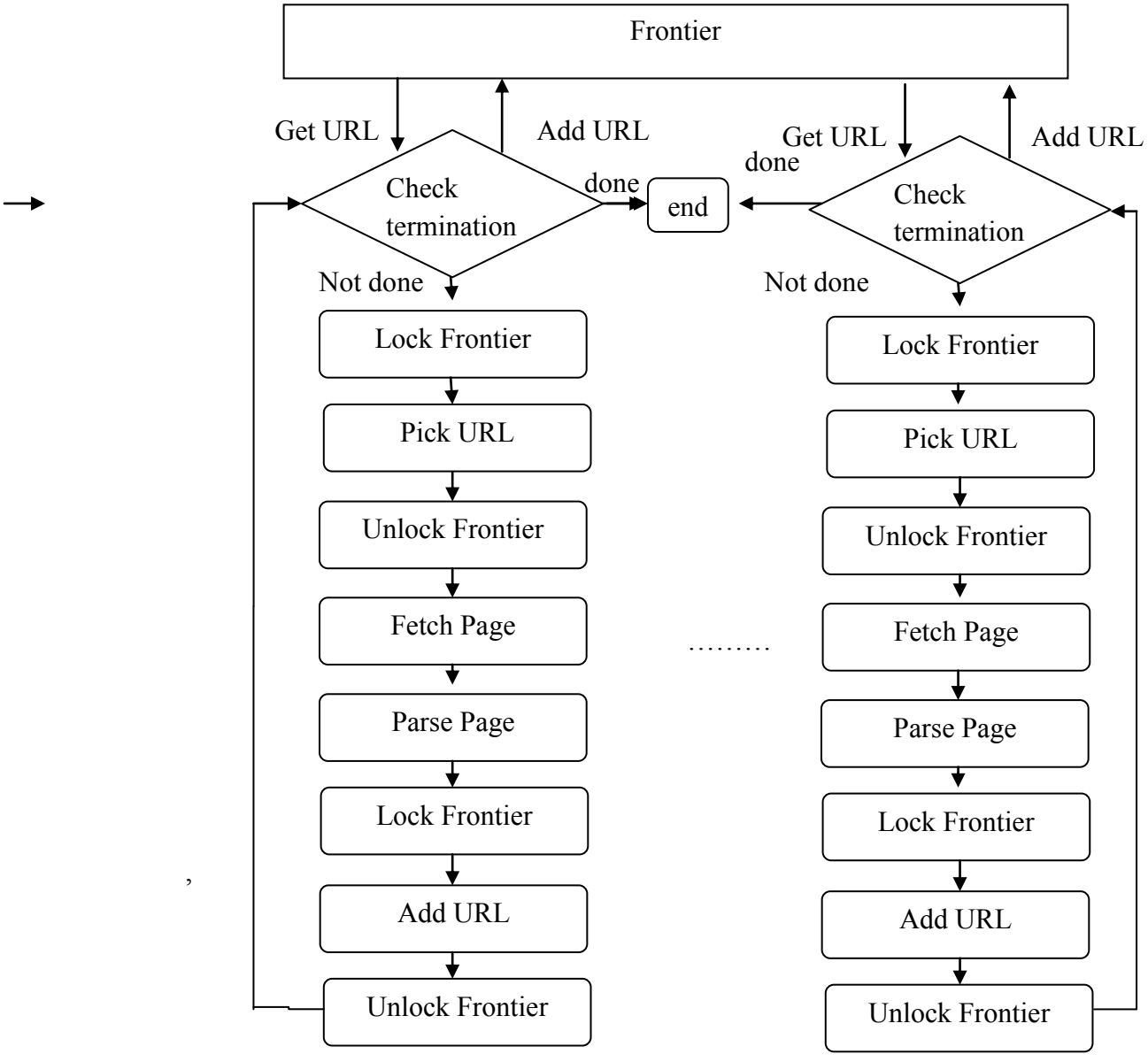

Figure 3.3 A Multi-threaded crawler model

3.1.3 Crawling Algorithms

There are a number of algorithms in the literature. The difference between these algorithms in the heuristics they use to score the unvisited links with some algorithms adapting and tuning their parameters before or during the crawl.

3.1.3.1 Naïve Best-First Crawler

A naïve best-first crawler represents a fetched web page as a vector of the words weighted by occurrence frequency. The crawler then computes the cosine similarity of the page to the query or description provided by the user, and scores the unvisited links on the page by this similarity value. The URLs are then added to the frontier that is maintained as a priority queue based on these scores. In the next iteration each crawl thread picks the best URL in the frontier to crawl, and returns with the unvisited URLs that are again inserted in the priority queue after being scored based on the cosine similarity of the parent page. The cosine similarity between the page p and a query q is computed by:

$$sim(q,p) = (Vq.Vp)/(|Vq|.|Vp|) \qquad (3.1)$$

Where Vq and Vp are term frequency (TF) based vector representations of the query and tge page respectively. Vq.Vp is the dot (linear) product of the two vectors, and |V| is the Euclidean norm of the vector V.

In a multiple thread implementation the crawler acts like a best–N-first crawler where N is a function of the number of simultaneously running threads. Thus best-N-first is a generalized version of the best-first crawler that picks N best URLs to crawl at a time. The best-first crawler keeps the frontier size within its upper bound by retaining only the last best URLs based on the assigned similarity scores [26].

3.1.3.2 SharkSearch

SharkSearch is a version of FishSearch with some improvements. In FishSearch, the crawlers search more extensively in areas of the web in which relevant pages have been found. Also, the algorithm discontinues searches in regions that do not yield relevant pages at the same time. SharkSearch uses a similarity measure like the one used in the naive best-first crawler for estimating the relevance of an unvisited URL.

However, SharkSearch has a more refined notion of potential scores for the links in the crawl frontier. The anchor text, text surrounding the links or link-context, and inherited score from ancestors influence the potential scores of links. The ancestors of a URL are the pages that appeared on the crawl path to the URL. SharkSearch, like its predecessor FishSearch, maintains a depth bound. That is, if the crawler finds unimportant pages on a crawl path it stops crawling further along that path. To be able to track all the information, each URL in the frontier is associated with a depth and a potential score. The depth bound (d) is provided by the user while the potential score of an unvisited URL is computed as:

$$score(url) = \gamma.inherited(url) + (1-\gamma).neighnorhood(url) \qquad (3\text{-}2)$$

Where $\gamma<1$ is a parameter, the neighborhood score signifies the contextual evidence found on the page that contains the hyperlink URL, and the inherited score is obtained fro the scores of the ancestors of the URL: More precisely, the inherited score is computed as:

$$inherited(url) = \begin{Bmatrix} \delta.sim(q,p)\ if\ sim(q,p) > 0 \\ \delta.inherited(p)\ \ otherwise \end{Bmatrix} \qquad (3\text{-}3)$$

where $\delta< 1$ is a parameter, q is the query, and p is the page from which the URL was extracted.

The neighborhood score uses the anchor text and the text in the “vicinity" of the anchor in an attempt to refine the overall score of the URL by allowing for differentiation between links found within the same page. For that purpose, the SharkSearch crawler assigns an anchor score and a context score to each URL. The anchor score is simply the similarity of the anchor text of the hyperlink containing the URL to the query q, i.e. sim(q; anchor text). The context score on the other hand broadens the context of the link to include some nearby words. The resulting augmented context, aug_context, is used for computing the context score as follows:

$$context(url) = \begin{Bmatrix} 1 & if\ anchor(url) > 0 \\ sim(q, aug\_context) & otherwise \end{Bmatrix} \qquad (3\text{-}4)$$

The neighborhood score is derived from the anchor score and the context score as :

$$neighborhood(url) = \beta.anchor(url) + (1-\beta).context(url) \qquad (3\text{-}5)$$

Where $\beta<1$ is parameter[30].

3.1.3.3 Focused Crawler

The basic idea of the focused crawler was to classify crawled pages with categories in a topic taxonomy. To begin, the crawler requires topic taxonomy such as Yahoo or the ODP. In addition, the user provides example URLs of interest (such as those in a bookmark file). The example URLs get automatically classified onto various categories of the taxonomy. Through an interactive process, the user can correct the automatic classification, add new categories to the taxonomy and mark some of the categories as “good" (i.e., of interest to the user). The crawler uses the example URLs to build a Bayesian classifier that can find the probability (Pr(c\p) that a crawled page *p* belongs to a category *c* in the taxonomy. Note that by definition Pr(*r*|*p*) = 1 where *r* is the root category of the taxonomy. A relevance score is associated with each crawled page that is computed as:

$$R(p) = \sum_{c \in good} \Pr(c \backslash p) \qquad (3\text{-}6)$$

When the crawler is in a “soft" focused mode, it uses the relevance score of the crawled page to score the unvisited URLs extracted from it. The scored URLs are then added to the frontier. Then in a manner similar to the naïve best-first crawler, it picks the best URL to crawl next. In the “hard" focused mode, for a crawled page *p*, the classifier first finds the leaf node *c** (in the taxonomy) with maximum probability of including *p*. If any of the parents (in the taxonomy) of *c** are marked as “good" by the user, then the URLs from the crawled page *p* are extracted and added to the frontier. Another interesting element of the focused crawler is the use of a distiller. The distiller applies a modified version of Kleinberg's algorithm [27] to find topical *hubs*. The hubs provide links to many authoritative sources on the topic. The distiller is activated at various times during the crawl and some of the top hubs are added to the frontier [28].

3.1.3.4 Context Focused Crawler

Context focused crawler uses Bayesian classifiers This classifier is trained to estimate the link distance between a crawled page and the relevant pages. The context focused crawler is trained using a context graph of *L* layers corresponding to each seed page. The seed page forms the layer 0 of the graph. The pages corresponding to the in-links to the seed page are in layer 1. The in-links to the layer

1 pages make up the layer 2 pages and so on. Once the context graphs for all of the seeds are obtained, the pages from the same layer (number) from each graph are combined into a single layer. This gives a new set of layers of what is called a merged context graph. This is followed by a feature selection stage where the seed pages (or possibly even layer 1 pages) are concatenated into a single large document. Using the TF-IDF scoring scheme, the top few terms are identified from this document to represent the vocabulary (feature space) that will be used for classification.

A set of naive Bayes classifiers are built, one for each layer in the merged context graph. All the pages in a layer are used to compute $Pr(t\backslash c_l)$, the probability of occurrence of a term t given the class c_l corresponding to layer l. A prior probability, $Pr(c_l) = 1\backslash L$, is assigned to each class where L is the number of layers. The probability of a given page p belonging to a class c_l can then be computed ($Pr(c_l\backslash p)$). Such probabilities are computed for each class. The class with highest probability is treated as the winning class (layer). However, if the probability for the winning class is still less than a threshold, the crawled page is classified into the “other" class. This “other" class represents pages that do not have a good fit with any of the classes of the context graph. If the probability of the winning class does exceed the threshold, the page is classified into the winning class.

The set of classifiers corresponding to the context graph provides us with a mechanism to estimate the link distance of a crawled page from relevant pages. If the mechanism works, the math department home page will get classified into layer 2 while the law school home page will get classified to “others." The crawler maintains a queue for each class, containing the pages that are crawled and classified into that class. Each queue is sorted by the probability scores ($Pr(c_l\backslash p)$). When the crawler needs a URL to crawl, it picks the top page in the non-empty queue with smallest l. So it will tend to pick up pages that seem to be closer to the relevant pages first. The out-links from such pages will get explored before the out-links of pages that seem to be far away from the relevant portions of the Web[29].

3.1.3.5 InfoSpiders

In InfoSpiders, an adaptive population of agents search for pages relevant to the topic. Each agent is essentially following the crawling loop while using an adaptive query list and a neural net to decide which links to follow. The algorithm provides an exclusive frontier for each agent.

Each agent consists of a list of keywords (initialized with a query or description) and a neural net used to evaluate new links. Each input unit of the neural net receives a count of the frequency with which the keyword occurs in the vicinity of each link to be traversed, weighted to give more importance to keywords occurring near the link (and maximum in the anchor text). There is a single output unit. The output of the neural net is used as a numerical quality estimate for each link considered as input. These estimates are then combined with estimates based on the cosine similarity (Equation 4.1) between the agent's keyword vector and the page containing the links.

A parameter α, $0 \leq \alpha \leq 1$ regulates the relative importance given to the estimates based on the neural net versus the parent page. Based on the combined score, the agent uses a stochastic selector to pick one of the links in the frontier with probability

$$\Pr(\gamma) = \frac{e^{\beta\sigma(\gamma)}}{\sum_{\gamma' \in \emptyset} e^{\beta\sigma(\gamma')}} \qquad (3\text{-}7)$$

where γ is a URL in the local frontier ($\emptyset$) and $\sigma(\gamma)$ is its combined score. The β parameter regulates the greediness of the link selector.

After a new page has been fetched, the agent receives “energy" in proportion to the similarity between its keyword vector and the new page. The agent's neural net can be trained to improve the link estimates by predicting the similarity of the new page, given the inputs from the page that contained the link leading to it. A back-propagation algorithm is used for learning. Such a learning technique provides InfoSpiders with the unique capability to adapt the link-following behavior in the course of a crawl by associating relevance estimates with particular patterns of keyword frequencies around links. An agent's energy level is used to determine whether or not an agent should reproduce after visiting a page. An agent reproduces when the energy level passes a constant threshold. The reproduction is meant to bias the search toward areas (agents) that lead to good pages. At reproduction, the

offspring (new agent or thread) receives half of the parent's link frontier. The offspring's keyword vector is also mutated (expanded) by adding the term that is most frequent in the parent's current document.[26]

4 DEVELOPING NEW STRATEGY TO CRAWLER

4.1 Properties of Crawler

InfoSpiders is the name of a multiagent model for online, dynamic information mining on the Web, which uses several artificial intelligence techniques to adapt to the characteristics of its networked information environment.

Each agent navigates from page to page following hypertext links, trying to locate new documents relevant to the user's query, having limited interactions with other agents. Agents mimic the intelligent browsing behavior of human users, being able to evaluate the relevance of a document's content with respect to the user's query, and to reason autonomously about future links to be followed. In order to improve the performance of the system adaption occurs at both individual and population levels, by evolutionary and reinforcement learning. The goal is to maintain diversity at a global level, trying to achieve a good coverage of all different aspects related to the query, while capturing the relevant features of the local information environment.

In InfoSpiders algorithm rely on either traditional search engines or a set of personal bookmarks in order to obtain a set of seed URLs pointing to pages supposedly relevant to the query handed in by the user. The size of the seed set is determined by the user and in turn determines the initial population size. The starting documents are prefetched, and each agent in the population is "positioned" at one of those documents and given an initial amount of energy. Energy is the currency that allows agents to survive and crawl. The initial population size is another parameter set by the user, determining how many starting points to use.

At each step, each agent analyzes the text of the document where it is currently situated to estimate the relevance of its information neighborhood, given by the outgoing hyperlinks in the current page. The agent then uses the link relevance estimates to choose the next document to visit. For link l and for each keyword k, the neural net receives input:

$$in_{k,l} = \sum_{i:dist(k_i,l)\leq\rho} \frac{1}{dist(k_i,l)} \qquad (4\text{-}1)$$

where k_i is the ith occurrence of k in document D and dist(k_i,l) gives more weight to keyword occurrences in the vicinity of l, by counting intervening links up to a maximum window size of ±ρ links away. The neural network then sums activity across all of its inputs; each unit j computes a logistic activation function

$$o_j = \tanh\left(b_j + \sum_k w_{j_k} in_k^l\right) \qquad (4\text{-}2)$$

where bj is its bias term, w_{jk} are its incoming weights, and in_k^l its inputs from the lower layer. The output of the network is the activation of the output unit, λ_l. The process is repeated for each link in the current document. Then, the agent uses a stochastic selector to pick a link with probability distribution:

$$\Pr[l] = \frac{e^{\lambda_l \beta}}{\sum_{l' \epsilon D} e^{\lambda_l \beta}} \qquad (4\text{-}3)$$

After a document has been visited, the agent needs to update its energy. The energy is used to survive and move, so the agent will be rewarded with energy based on the estimated relevance of the visited documents. Agents are also charged with energy to account for the work performed to download the page from the network. The relevance estimation function and the cost function must be specified in an implementation of the InfoSpiders algorithm.

If an agent accumulates enough energy, it clones a new agent in the location where it is currently situated. At reproduction, the offspring's neural net is mutated by adding random noise to a fraction of the weights. The keyword vector is mutated by adding the most frequent term from the current document. The idea is that since the current page led to reproduction, it probably contains features that are correlated with the user's interests and therefore we want to expand the agent's representation to capture such features. The parent and offspring agents can continue the search independently of each other. If an agent runs out of energy, it is destroyed. Such reproduction and death mechanisms with their fixed thresholds make selection a local decision, independent of other agents. This way, agents are dispatched to the most promising areas of the information space.

Another important consequence of the local selection mechanism is that agents have minimal mutual interactions, making it possible for them to execute concurrently.

```
InfoSpiders(query, INIT_POP){
        starting_urls:= search_engine(query, INIT_POP);
        for agent(1. . .INIT_POP){
                initialize(agent,query);
                situate(agent, starting_urls);
                agent.energy:= THETA/2;
        }
        until extinction{
                foreach agent{
                        pick_out_link_from_current_document(agent);
                        agent.doc:=fetch_new_document(agent);
                        agent.energy+=estimate(agent.doc)-cost(agent.doc);
                        Q_learning(agent, estimate(agent.doc));

                        if(agent.energy>=THETA){
                                offspring :=mutate(clone(agent));
                                offspring.energy :=agent.energy/2;
                                agent.energy -= offspring.energy;
                }
                elseif(agent.energy<=0) kill(agent);
        }
}
```

Figure 4.1 InfoSpiders algorithms [26]

```
InfoSpiders(query, INIT_POP){
        starting_urls:= search_engine(query, INIT_POP);
        starting_urls.add(database(query, 50))
        for agent(1. . .INIT_POP){
                initialize(agent,query);
                situate(agent, starting_urls);
                agent.energy:= THETA/2;
        }
        until extinction{
                foreach agent{
                        pick_out_link_from_current_document(agent);
                        agent.doc:=fetch_new_document(agent);
                        frontier.add(new_link_from_document(agent))
                        agent.energy+=estimate(agent.doc)-cost(agent.doc);
                        Q_learning(agent, estimate(agent.doc));
                        if(agent.energy>=THETA){
                                offspring :=mutate(clone(agent));
                                offspring.energy :=agent.energy/2;
                                agent.energy -= offspring.energy;
                }
                elseif(agent.energy<=0) kill(agent);
        }
}
```

Figure 4.2 Database applied InfoSpiders algorithm

In addition to InfoSpiders algorithm, a database that is built from previous crawls that been starting from a seed set of URLs relating to the topic, and then crawling out to several levels away from the original seeds. The database includes downloaded pages URL and its score. Pages are classified using a Support vector Machine (SVM) trained on the features of a seed set of the topically related URLs .

The crawler also maintains several lists of URL data. First, it keeps a current list, which is the list of URLs that the crawler is currently downloading. It also maintains the crawl frontier list; URLs that have been seen during the course of the crawl but not yet added to the current list. The data associated with these URLs includes a rank score that determines when or if a given URL will be added to the current list. Once a URL has been selected to return from the current list, it is added to a downloaded list. This list is used to determine the success or fitness of the strategy used in crawler. Pages that are marked by the SVM as positive, or related to the topic, are also put into topic community list.

4.2 Crawler Evaluation

In real life users may judge the relevancies of page. In addition, a crawler may be evaluated on its ability to retrieve “good” pages. But, there is a difficult problem for recognizing these good pages. Meaningful experiments involving real users for assessing Web crawls are extremely problematic. For instance the very scale of the Web suggests that in order to obtain a reasonable notion of crawl effectiveness one must conduct a large number of crawls.

Another problem is, crawls against the live Web pose serious time constraints. Therefore crawls other than short-lived ones will seem overly burdensome to the user.

In the not so distant future, the majority of the direct consumers of information is more likely to be Web agents working on behalf of humans and other Web agents than humans themselves. Thus it is quite reasonable to explore crawlers in a context where the parameters of crawl time and crawl distance may be beyond the limits of human acceptance imposed by user based experimentation. In general, it is important to compare topical crawlers over a large number of topics and tasks. This will allow users to ascertain the statistical significance of particular benefits that we may

observe across crawlers. Crawler evaluation research requires an appropriate set of metrics. Recent research reveals several innovative performance measures. Two basic of them are crawled page's importance and a method to summarize performance across a set of crawled pages.

There are some methods for measuring crawled page's importance.

- Keywords in document: A page is considered relevant if it contains some or all of the keywords in the query. Also, the frequency with which the keywords appear on the page may be considered[33].
- Similarity to a query: Often a user specifies an information need as a short query. In some cases a longer description of the need may be available. Similarity between the short or long description and each crawled page may be used to judge the page's relevance [31].
- Similarity to seed pages: The pages corresponding to the seed URLs, are used to measure the relevance of each page that is crawled.[32]. The seed pages are combined together into a single document and the cosine similarity of this document and a crawled page is used as the page's relevance score.
- Classifier score: A classifier may be trained to identify the pages that are relevant to the information need or task. The training is done using the seed (or pre-specified relevant) pages as positive examples. The trained classifier will then provide boolean or continuous relevance scores to each of the crawled pages [31].
- Retrieval system rank: N different crawlers are started from the same seeds and allowed to run till each crawler gathers P pages. All of the N . P pages collected from the crawlers are ranked against the initiating query or description using a retrieval system such as SMART. The rank provided by the retrieval system for a page is used as its relevance score [27].
- Link-based popularity: One may use algorithms, such as PageRank or HITS , that provide popularity estimates of each of the crawled pages. A simpler method would be to use just the number of in-links to the crawled page to derive similar information. Many variations of link-based methods using topical weights are choices for measuring topical popularity of pages [32].

The performance of the crawler with metrics that is analogous to the information retrieval (IR) measures of precision and recall. Precision is the fraction of retrieved (crawled) pages that are relevant, while recall is the fraction of relevant pages that are retrieved (crawled). In a usual IR task the notion of a relevant set for recall is restricted to a given collection or database. Considering the Web to be one large collection, the relevant set is generally unknown for most Web IR tasks. Hence, explicit recall is hard to measure. Many authors provide precision-like measures that are easier to compute in order to evaluate the crawlers.

- Acquisition rate: relevance scores are measured the explicit rate at which "good" pages are found. Therefore, if 50 relevant pages are found in the first 500 pages crawled, the acquisition rate or harvest rate of 10% at 500 pages [34].
- Average relevance: If the relevance scores are continuous they can be averaged over the crawled pages. This is a more general form of harvest rate. The scores may be provided through simple cosine similarity or a trained classifier [29].

Since measures analogous to recall are hard to compute for the Web, authors resort to indirect indicators for estimating recall. Some such indicators are:

- Target recall: A set of known relevant URLs are split into two disjoint sets- targets and seeds. The crawler is started from the seeds pages and the recall of the targets is measured. The target recall is computed as:

$$target_{recall} = |P_t \cap P_c| / |P_t| \qquad (4\text{-}4)$$

  where P_t is the set of target pages, and P_c is the set of crawled pages. The recall of the target set is used as an estimate of the recall of relevant pages.

- Robustness: The seed URLs are split into two disjoint sets S_a and S_b. Each set is used to initialize an instance of the same crawler. The overlap in the pages crawled starting from the two disjoint sets is measured. A large overlap is interpreted as robustness of the crawler in covering relevant portions of the Web [35].

There are other metrics that measure the crawler performance in a manner that combines both precision and recall. For example, search length measures the number of pages crawled before a certain percentage of the relevant pages are retrieved.

The system was tested by evolving strategies to InfoSpiders Algorithm 50, 100, 250 and 500 pages. InfoSpiders and Improved Infospiders start from seed pages.

- "Java" string is used for crawling. In Google, java keyword has 378,000,000 results. InfoSpiders start from seed pages and agents are trying to found new pages. Improved InfoSpiders start from database and seed pages from search engine. The population size of agent is increased in Improved InfoSpiders. So, crawling takes less time.

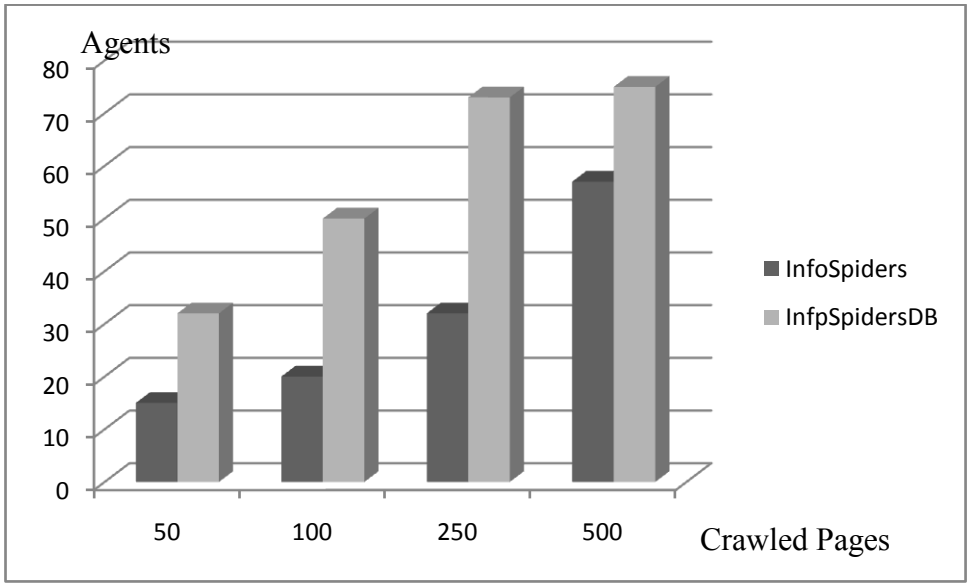

Figure 4.3 Population size

The score of crawling is increased in Improved InfoSpiders. Improved InfoSpiders fetches more web pages. It starts with more seed pages and make search very fast. The population size show that more agent affects the duration time of search. InfoSpiders starts with only 10 seed pages and fetching time and picking up new links takes more times. On the other hand, Improved InfoSpiders starts with more seed pages and it uses more agents. Agents are increased 70 in 500 pages crawled.

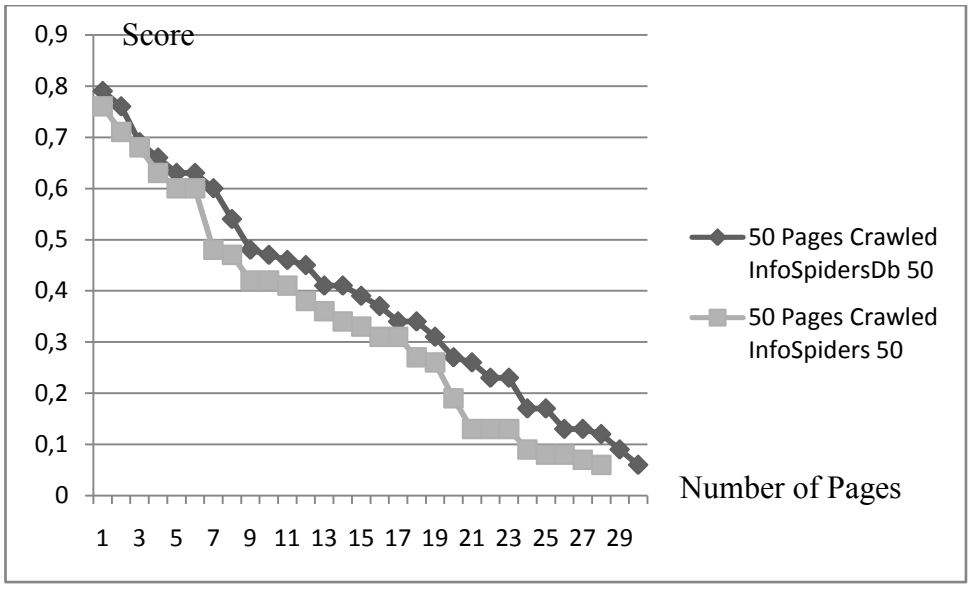

Figure 4.4 The efficiency score of crawled 50 pages

The efficiency score of InfoSpiders and Improved InfoSpiders show in Figure 4.4 . Improved InfoSpiders fetches more web sites so it can found more relevant sites.

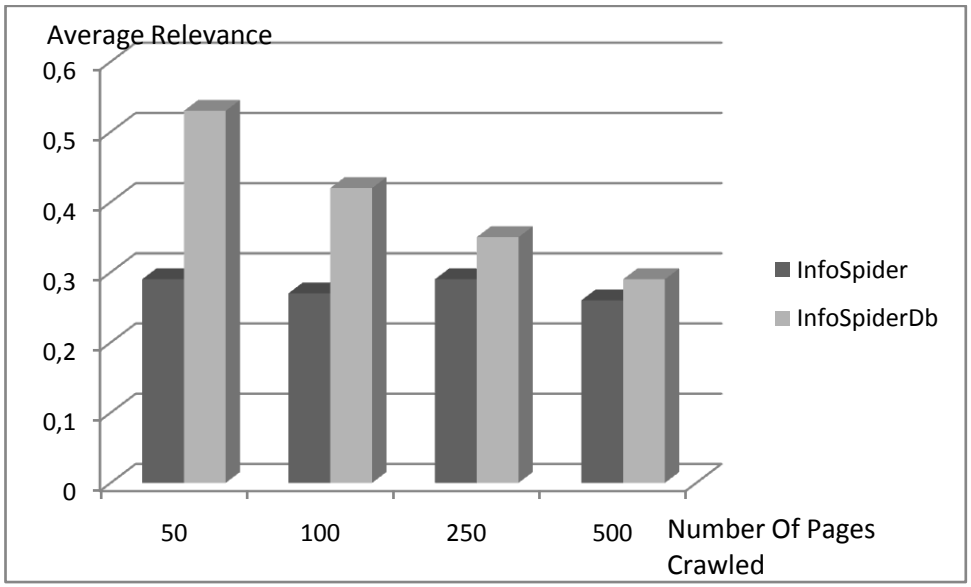

Figure 4.5 Average relevance score of crawled pages

The average relevance scores of InfoSpidersDB are higher. This is the more general form of the harvest rate. In 50 pages crawled the average relevance score of Improved InfoSpiders are 0,53. If the crawled pages are increased the average relevance score in decreased. On the other hand, InfoSpiders Algorithm the average relevance score is nearly the same..

5 CONCLUSION

Web mining is the new topic. Web mining research relates to several research areas such as Database, Information retrieval, artificial Intelligence and data mining. However, it uses a huge and unstructured data. With the growth of Web-based applications, especially electronic commerce, there is a significant interest analyzing Web mining. The intelligent agents help to improve the ability of our knowledge. They serve us all of the web mining process. In the web structure mining, multi-agents help us to improve the speed of our links. In the web usage mining they help to configure our web page and servers to serve better. In the web content mining multi-agents they work with search engine and find relevant pages that user's request. With the increase of sources on web, intelligent tools needed.

Improved InfoSpiders was able to be genetic algorithm to evolve strategies that combined analysis of text and link structure to outperform InfoSpiders strategy on crawls of several different durations. The evolved strategy is able to rank links on the same page differently based on how those URLS are linked to other pages in the crawl. The strategy evolved concentrated on exploiting the known good information more than on exploring new, possibly promising areas for search.

REFERENCES

[1] B., Lui, *Web Data Mining Exploring Hyperlinks, Contents and Usage Data*, Springer , 2007

[2] M. F. Porter. *An Algorithm for Suffix Stripping. Program*, 14(3), pp 130-137, 1980.

[3] D. Cai, S. Yu, J.-R. Wen and W.-Y Ma. *Extracting Content Structure for Web Pages based on Visual Representation.* In Proc. of the APWeb'03 Conf., Number 2642 in Lecture notes in Computer Science (LNCS), pp. 406–417, 2003

[4] D. Tanasa and B. Trousse. *Advanced Data Preprocessing for Intersite Web Usage Mining.* IEEE Intelligent Systems, 19(2), pp. 59–65, 2004.

[5] R. Kohavi, L. Mason, R. Parekh, and Z. Zheng. *Lessons and Challenges from Mining Retail E-Commerce Data. Machine Learning*, 57(1–2), pp. 83–113, 2004.

[6] A. Buchner, M. D. Mulvenna, *Discovering Internet Marketing Intelligence through Online Analytical Web Usage Mining*, In Proc. of the ACM SIGMOD Intl. Conf. on Management of Data (SIGMOD'99), pp. 54–61, 1999.

[7] Lise Getoor, *Link Mining: A New Data Mining challenge*, SIGKDD Eplorations, vol. 4, issue 2, 2003

[8] Q. Lu, and L.Getoor *Link-based classification*, In Proceeding of ICML-03, 2003

[9] J.M.Kleinberg, *Authoritative sources in a hyperlinked environment*. In proceedings of ACM-SIAM Synmposium on discrete Algorithms, pages 668-677 – 1998

[10] WikiPedia, the free encyliclopedia, *http://en.wikipedia.org/wiki/HITS_algorithm*, 9th January 2009

[11] Brin, S; Page, L. *The Anatomy of a Large-Scale Hypertextual web Search Engine*. Proceeding of the seventh International World Wide Web Conference, 1998

[12] Google, Search Engine, *http://www.google.com*, 10th December 2008

[13] M., Bianchini, M., Gori, F., Scarselli, *Inside PageRank*. ACM Transaction on Internet Technology (TOIT), Volume 5 Issue 1 – February, 2005.

[14] Ziv Bar-Yossef and Sridhar Rajagopalan. *Template Detection via Data Mining and its Applications*. In: Proceedings of WWW2002, Honolulu, Hawaii, USA. 580-591, May 7-11, 2002

[15] R. Cooley, B. Mobasher, and J. Srivastava. *Web Mining: Information and pattern discovery on the World Wide Web*. In Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97), 1997

[16] R. Kosala, H. Blockeel. *Web mining Research: A Survey*, 2000

[17] M. Spiliopoulou. *Data mining for the Web*. In Proceedings of Principles of Data Mining and Knowledge Discovery, Third European conference, PKDD'99, P588-589

[18] O. Zaiane, M. Xin, J. Han. *Discovering Web Access Patterns and Trends by applying OLAP and Data Mining Technology on Web Logs*. In Advances in Digital Libraries, pages 19-29, Santa Barbara, CA, 1998

[19] M, Erinaki, M., Vazirgiannis. *Web Mining for Web Personalization*, ACM, 2003

[20] J.Srivastava, R., Cooley, M., Deshpande, and P., Tan,. 2000. *Web usage mining: Discovery and applications of usage patterns from web data*. SIGKDD Explorations 1, 2 (Jan.), 12–23

[21] Wooldridge, Michael and Jennings, Nicholas R. (1995). *Intelligent agents: Theory and practice. Knowledge Engineering Review*, 10(2):115-152.

[22] Genesereth, M. R. and Ketchpel, S. (1994). *Software agents.* Communications of the ACM, 37 : 48-53.

[23] Copernic, Software the Search, Find and Manage Information, *http://www.copernic.com*, 10th December 2008

[24] Pant Gautam, Srinivasan Padmini, Menczer Fillippo, *Crawling the Web*,

[25] G. Pant. *Deriving Link-context from HTML Tag Tree*. In 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 2003.

[26] F. Menczer, G. Pant, M. Ruiz, and P. Srinivasan. *Evaluating topic-driven Web crawlers*. In Proc. 24th Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2001.

[27] J. Kleinberg. *Authoritative sources in a hyperlinked environment*. Journal of the ACM, 46(5):604-632, 1999

[28] S. Chakrabarti, M. van den Berg, and B. Dom. *Focused crawling: A new approach to topic-specific Web resource discovery*. Computer Networks, 31(11-16):1623-1640, 1999.

[29] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. *Focused crawling using context graphs*. In Proc. 26th International Conference on Very Large Databases (VLDB 2000), pages 527{534, Cairo, Egypt, 2000.

[30] M. Hersovici, M. Jacovi, Y. S. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur. *The shark-search algorithm An application: Tailored Web site mapping*. In WWW7, 1998.

[31] B. Amento, L. Terveen, and W. Hill. *Does "authority" mean quality? Predicting expert quality ratings of web documents*. In Proc. 23th Annual International. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2000.

[32] K. Bharat and M.R. Henzinger. *Improved algorithms for topic distillation in a hyperlinked environment.* In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1998.

[33] J. Cho, H. Garcia-Molina, and L. Page. *Efficient crawling through URL ordering*. Computer Networks, 30(1{7):161{172, 1998.

[34] S. Chakrabarti, K. Punera, and M. Subramanyam. *Accelerated focused crawling through online relevance feedback*. In WWW2002, Hawaii, May 2002.

[35] S. Chakrabarti. *Mining the Web*. Morgan Kaufmann, 2003.

[36] F. Menczer and R. K. Belew. *Adaptive retrieval agents: Internalizing local context and scaling up to the Web. Machine Learning*, 39(2-3):203-242, 2000.