Intelligent Crawling on the World Wide Web with Arbitrary Predicates

Charu C. Aggarwal
IBM T. J. Watson Resch. Ctr.
Yorktown Heights, NY 10598
charu@watson.ibm.com

Fatima Al-Garawi
IBM T. J. Watson Resch. Ctr.
Yorktown Heights, NY 10598
algarawi@cs.columbia.edu

Philip S. Yu
IBM T. J. Watson Resch. Ctr.
Yorktown Heights, NY 10598
psyu@watson.ibm.com

ABSTRACT

The enormous growth of the world wide web in recent years has made it important to perform resource discovery efficiently. Consequently, several new ideas have been proposed in recent years; among them a key technique is focused crawling which is able to crawl particular topical portions of the world wide web quickly without having to explore all web pages. In this paper, we propose the novel concept of intelligent crawling which actually *learns* characteristics of the linkage structure of the world wide web while performing the crawling. Specifically, the intelligent crawler uses the inlinking web page content, candidate URL structure, or other behaviors of the inlinking web pages or siblings in order to estimate the probability that a candidate is useful for a given crawl. This is a much more general framework than the focused crawling technique which is based on a pre-defined understanding of the topical structure of the web. The techniques discussed in this paper are applicable for crawling web pages which satisfy arbitrary user-defined predicates such as topical queries, keyword queries or any combinations of the above. Unlike focused crawling, it is not necessary to provide representative topical examples, since the crawler can learn its way into the appropriate topic. We refer to this technique as *intelligent crawling* because of its adaptive nature in adjusting to the web page linkage structure. The learning crawler is capable of reusing the knowledge gained in a given crawl in order to provide more efficient crawling for closely related predicates.

Keywords

World Wide Web, Crawling, Querying

1. INTRODUCTION

With the rapid growth of the world wide web, the problem of resource collection on the world wide web has become very relevant in the past few years. Users may often wish to search or index collections of documents based on topical or keyword queries. Consequently, a number of search engine technologies such as *Yahoo!*, *Lycos* and *AltaVista* [15, 16, 17] have flourished in recent years. The standard method for searching and querying on such engines has been to 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.

Some interesting methods proposed in recent years are those of *Fish Search* [3] and *focused crawling* [6]. The essential idea in focused crawling is that there is a short range topical locality on the web. This locality may be used in order to design effective techniques for resource discovery by starting at a few well chosen points and maintaining the crawler within the ranges of these known topics. In addition, the hubs and authorities [13] for the different web pages may be identified and used for the purpose of crawling. The idea in this framework is that resources on a given topic may occur in the form of hub pages (web pages containing links to a large number of pages on the same topic) or authorities (documents whose content corresponds to a given topic). Typically, the hubs on a given topic link to the authorities and vice-versa. The method in [6] uses the hub-authority model in addition to focusing techniques in order to perform the crawl effectively. Starting with these methods, there has been some recent work on various crawlers which use similar concepts [8, 14] in order to improve the efficiency of the crawl. Other related work may be found in [2, 4, 5, 11, 9, 10].

The basic idea of crawling selectively is quite attractive from a performance perspective; however, the focused crawling technique [6] relies on a restrictive model based on certain pre-ordained notions of how the linkage structure of the world wide web behaves. Specifically, the work on focused crawling [6] assumes two key properties: (1) **Linkage Locality:** Web pages on a given topic are more likely to link to those on the same topic. (2) **Sibling Locality:** If a web page points to certain web pages on a given topic, then it is likely to point to other pages on the same topic. Such a page may also be referred to as a hub [13], and the highly topic-specific pages that it points to are referred to as authorities. Typically the hubs on a subject are likely to point to authorities and vice-versa.

In order to achieve the goal of efficiently finding resources of a given topic, the focused crawling technique [6] starts at a set of representative pages on a given topic and forces

WWW10 May 1-5, 2001, Hong Kong.
ACM 1-58113-348-0/01/0005.

the crawler to stay focused on this topic while gathering web pages. A topic is defined with the help of a hypertext classifier which is pre-trained with a representative data set and corresponding topical classes. The crawling system is dependent on a *hierarchical* scheme in order to effectively keep the crawler focused on web pages which are closely related to the current topic. Relevance is forced on the crawler with the use of hierarchy-dependent hard and soft focusing rules. Both of these rules use classes which are hierarchically related to the crawl-topic in order to identify which candidate documents are most suitable for exploration. In particular, the observation that using a greater level of specificity (hard focus rule) [7] for forcing relevance may cause the crawler to *stagnate* shows the sensitivity of the system to using arbitrary predicates such as a non-hierarchical classification system or a combination of different kinds of constraints on the web page. Provision of such trained hierarchical classifiers which are well representative of the web resource structure is not always possible from a practical perspective. The stagnation results of [7] would indicate that the crawling technique is also highly sensitive to the quality of the hierarchy used. Users are often likely to want to provide arbitrary *predicates* in order to perform resource discovery. These arbitrary predicates could be simple keywords (as in search engines where pre-crawled data is stored), topical searches using hypertext classifiers (which could be non-hierarchical), document-to-document similarity queries, topical linkage queries, or any combination of the above. A predicate is implemented as a subroutine which uses the content and URL string of a web page in order to determine whether or not it is relevant to the crawl. It is assumed that a user has the flexibility of providing a module which computes this predicate.

Another of the assumptions in the focused crawling model is the availability of starting points which are well representative of the topical area of discovery. Users may not be aware of the best possible starting points which are representatives of the predicate. It is also clear that a crawler which is intelligent enough to start at a few general pages and is still able to selectively mine web pages which are most suitable to a given predicate is more valuable from the perspective of resource discovery.

Focussed crawling is a *fixed* model in which it is assumed that the web has specific linkage structure in which pages on a specific topic are likely to link to the same topic. Even though there is evidence [7, 13] that the documents on the world wide web show topical locality for many broad topical areas, there is no clear understanding of how this translates to arbitrary predicates. In addition, the method does not use a large amount of information which is readily available such as the exact content of the inlinking web pages, or the tokens in a given candidate URL. Such data may provide valuable information in order to direct a crawl effectively; however, it has not been accounted for in the focused crawling model. In general, one would expect that for an arbitrary predicate, one of these factors may turn out to be more important, and that the ordering of the different factors may vary with the crawl.

A more interesting and significantly more general alternative to focused crawling is *intelligent crawling*; here no specific model for web linkage structure is assumed; rather the crawler gradually *learns* the linkage structure statistically as it progresses. Since each (candidate) web page can be characterized by a large number of features such as the content of the inlinking pages, tokens in a given candidate URL, the predicate satisfaction of the inlinking web pages, and sibling predicate satisfaction, it may be useful to learn how these features for a given candidate are connected to the probability that it would satisfy the predicate. In general, we expect the exact nature of this dependence to be *predicate-specific*; thus, even though for a given predicate the documents may show topical locality, this may not be true for other predicates. For some predicates, the tokens in the candidate URLs may be more indicative of the exact behavior, whereas for others the content of the inlinking pages may provide more valuable evidence. There is no way of knowing the importance of the different features for a given predicate a-priori. It is assumed that an intelligent crawler would learn this during the crawl and find the most relevant pages.

We propose an intelligent crawling system which starts at a few general starting points and collects all web pages which are relevant to the user-specified predicate. Initially, the crawler behavior is as random as a general crawler, but it gradually starts *auto-focusing* as it encounters documents which satisfy the predicate. In general, a crawler which is insensitive to the starting point is more *stable*, since it can always learn its way into the appropriate topic. Even when good starting points are available, the stability of a learning crawler ensures that it remains on topic throughout the entire process.

One of the aims of this paper is to illustrate that depending upon the nature of the predicate, the importance of different factors (eg. short range locality, content information, etc.) can vary quite a bit. Therefore, a learning crawler has certain advantages over a crawler which has been designed with fixed notions about the structure of the world wide web. This also helps in the creation of arbitrary predicates which so not depend critically on a hierarchical classification scheme in order to keep the crawler on topic. Since the world wide web is rapidly evolving over time [12], it may often be desirable to repeat the same queries at different time periods. Alternatively, it is often desirable to perform crawls for very closely related predicates. In such cases, the learning crawler provides a key advantage, since it is able to reuse the learning information that it gained from a given crawl in order to improve the efficiency of subsequent crawls. In the empirical section, we will show specific examples of such cases.

This paper is organized as follows. In the next section, we discuss the general framework for the crawler, and the factors which may be used for the purpose of self-learning. In section 3, we will provide an overview of the statistical model which is used for the purpose of the crawling. The actual implementation and algorithmic details are discussed in section 4. Section 5 discusses the empirical results, and the conclusions and summary are discussed in section 6.

2. GENERAL FRAMEWORK

The crawler is implemented as a graph search algorithm which works by treating web pages as nodes and links as edges. Each time a web page is crawled, it is parsed in order to find both its text content and the Universal Resource Locators (URLs) that it links to. The crawler keeps track of the nodes which it has already visited, as well as a potential list of *candidates*. A web page is said to be a can-

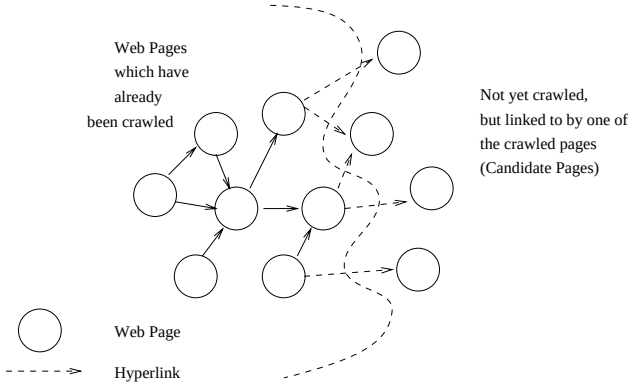

Figure 1: Illustration of candidates during a crawl

didate when it has not yet been crawled, but some web page which links to it has already been crawled. An example of a set of candidates is illustrated in Figure 1. For each such candidate, considerable amount of information is available based on the web pages which link to it, their content, and the exact tokens in the URL address of the candidate itself. This information may be used by the crawler to decide the order in which it visits the web pages. In addition, an aggregate level of self-learning information is collected during the crawl which models the relationship between the features in the candidates to the actual predicate satisfaction probability. To summarize, the priority of the nodes in the order in which they are likely to satisfy the predicate is based on two factors:

• Self-Learning Information which has already been collected during the crawl. We denote this information by $\mathcal{K}$.

• The features of the current sets of candidates and their relationship to $\mathcal{K}$.

Thus, at each point the crawler maintains candidate nodes which it is likely to crawl and keeps calculating the priorities of the nodes using the information about which nodes are most likely to satisfy the predicate. The pages are crawled in this order, and the appropriate statistical information is collected based on the relationship between the features in the candidate URLs and the predicate satisfaction of the crawled web page. This statistical information may be used to create the model which learns the structure of the web as more and more pages are crawled.

2.1 Statistical Model Creation

In order to create an effective statistical model which models linkage structure, what are we really searching for? We are looking for specific *features* in the web page which makes it more likely that the page links to a given topic. These features may include but are not restricted to:

• The content of the web pages which are known to link to the candidate URL (the set of words).

• URL tokens from the candidate URL (eg. if we are looking for skiing web pages, the word "ski" in the URL is highly suggestive.) For this purpose, we also discuss feature extraction mechanisms from the tokens inside the candidate URL. In general, URL names of web pages contain highly relevant information, since web page and server names are usually not chosen randomly.

• The nature of the inlinking web pages of a given candidate URL. Thus, if the inlinking web pages satisfy the predicate, then the candidate is also more likely to satisfy the predicate itself.

• The number of siblings of a candidate which have already been crawled that satisfy the predicate. A web page is said to be a sibling of a candidate URL, when it is linked to by the same page as the candidate. When a candidate URL has a large number of siblings which satisfy the predicate, then this is good evidence that the candidate itself satisfies the predicate.

In general, the importance of any of the above kinds of features may be predicate dependent; it is the crawler's job to learn this. It is expected that as the crawl progresses, and more and more web pages satisfying the predicate are encountered the system will be able to show high specificity in finding web pages which satisfy the user-defined predicate.

3. OVERVIEW OF STATISTICAL MODEL

The model input consists of the features and linkage structure of that portion of the world wide web which has already been crawled so far, and the output is a priority order which determines how the candidate URLs (linked to by the already crawled web pages) are to be visited. Our statistical model maintains a dynamically updated set of statistical information $\mathcal{K}$ which it has learned during the crawl, and a set of features in the given web page and computes a priority order for that web page using this information.

The set of features may be any of the types which have been enumerated above including the content information, URL tokens, linkage or sibling information. This priority order dictates which candidate is next picked to be crawled. As we shall see later, the particular priority order which we determine calculates the interest factor on the likelihood that the features for a candidate web page make it more likely that this page satisfies the predicate.

3.1 Probabilistic Model for Priorities

In this section, we discuss the probabilistic model for calculation of the priorities. In order to calculate the priorities, we compute the ratio which signifies whether a given set of events makes it more likely for a candidate to satisfy the user defined predicate. In order to understand this point a little better, let us consider the following case. Suppose that we are searching for web pages on online malls. Let us assume that only 0.1% of the pages on the web correspond to this particular predicate. However, it may happen that when the word "eshop" occurs inside one of the inlinking pages then the probability that the candidate is a page belonging to an online mall increases to 5%. In that case, the interest ratio for the *event* corresponding to the word "eshop" occurring in the inlinking page is $5/0.1 = 50$.

In order to explain the model more concisely, we will develop some notations and terminology. Let C be the event that a crawled web page satisfies the user defined predicate. For a *candidate page* which is about to be crawled, the value of $P(C)$ is equal to the probability that the web page will indeed satisfy the user-defined predicate if it is crawled. The value of $P(C)$ can be estimated by the fraction of web pages already crawled which satisfy the user defined predicate.

Let E be a fact that we know about a candidate URL. This fact could be of several types. For example, it could be a fact about the content of the inlinking web pages into this candidate URL, it could be a fact about the set of tokens in the string representing the URL, or it could be a fact about

the linkage structure of the URL. We will explore all of these options slightly later.

Our knowledge of the event E may increase the probability that the web page satisfies the predicate. For example, consider the case when the candidate URL is linked to by another web page which belongs to the same topic. In such a case, it is evident from earlier results on focused crawling [6], that the resulting web page is more likely to satisfy the predicate. Thus, in this case, we have $P(C|E) > P(C)$. In order to evaluate $P(C|E)$, we use the following relationship:

$$P(C|E) = P(C \cap E)/P(E) \quad (1)$$

Therefore, we have:

$$P(C|E)/P(C) = P(C \cap E)/(P(C) \cdot P(E)) \quad (2)$$

The idea is that the values of $P(C \cap E)$ and $P(E)$ can be calculated using the information that has been accumulated by the crawler. This is the self-learning data $\mathcal{K}$ which is accumulated over time during the crawling process. Correspondingly, we calculate the interest ratio for the event C, given event E as $I(C, E)$. Therefore, we have:

$$I(C, E) = P(C|E)/P(C) \quad (3)$$

Note that when the event E is favorable to the probability of the candidate satisfying the predicate, then the interest ratio $I(C, E)$ is larger than 1. Correspondingly, when the event E is unfavorable, then this interest ratio will be in the range $(0, 1)$. Such a situation occurs when the event E makes the candidate less desirable to crawl.

Let $E_1 \ldots E_k$ be a set of k events. Let the composite event $\mathcal{E}$ be defined by the occurrence of all of these events. In other words, we have $\mathcal{E} = E_1 \cap E_2 \ldots E_k$. Then the composite interest ratio is defined as follows:

$$I(C, \mathcal{E}) = \pi_{i=1}^{k} I(C, E_i) \quad (4)$$

Again, a composite event $\mathcal{E}$ is interesting when the corresponding interest ratio is larger than 1. We will now proceed to examine the different factors which are used for the purpose of intelligent crawling.

3.2 Content Based Learning

In order to identify the value of the content in determining the predicate satisfaction of a given candidate page, we find the set of words in the web pages which link to it (inlinking web pages). These words are then used in order to decide the importance of the candidate page in terms of determining whether or not it satisfies the predicate. Let $W_1, W_2 \ldots W_n$ be the set of the words in the lexicon. Only a small subset of these words are present in the web pages which point to the candidate. We define the event Q_i to be true when the word i is present in one of the web pages pointing to the candidate.

Let $M = \{i : \text{Event } Q_i \text{ is true}\}$
Now, let us consider a given word W_i such that $i \in M$. Therefore, the event Q_i is true. If C be the event that a candidate URL is likely to satisfy the predicate, then let us calculate the value of $I(C, Q_i)$:

$$I(C, Q_i) = P(C \cap Q_i)/(P(C) \cdot P(Q_i)) \quad (5)$$

It now remains to estimate the parameters on the right hand side of the above equation. In order to estimate these parameters, we can only rely on the experimental evidence of the web pages which we have crawled so far. The exact details of these estimations will be discussed in a later section.

It is noted that most words will convey noisy information, and may not be of much use in calculating the interest ratios. In fact, since most words are unlikely to have much bearing on the probability of predicate satisfaction, the filtering of such features is important in reduction of the noise effects. Therefore, we use only those words which have high statistical significance. In order to do so, we calculate the level of significance at which it is more likely for them to satisfy the predicate. Let $n(C)$ be the number of pages crawled so far which satisfy the user defined predicate. Then, if N is the total number of pages which have been crawled so far, we have $n(C) = N \cdot P(C)$. The significance factor for the event C and condition Q_i is denoted by $S(C, Q_i)$ and is calculated as follows:

$$S(C, Q_i) = (P(C|Q_i) - P(C))/(\sqrt{P(C) \cdot (1 - P(C))/n(C)}) \quad (6)$$

Note that we use the absolute value of the above expression since the value of $P(C|Q_i)$ may be either larger or smaller than $P(C)$; correspondingly the interest ratio will be either smaller or larger than 1. For some pre-defined significance threshold t, we now define the significant composite ratio to include only those terms which are in M, and for which $S(C, Q_i)$ is above this threshold. Note that the same process can be applied to all the words which are not in M; however, such words are rarely statistically significant. Consequently, it is of value to use only the subset of the words which results in a noise free evaluation of the level of significance of the importance of a web page. The interest ratio for content based learning is denoted by $I_c(C)$, and is calculated as the product of the interest ratios of the different words in any of the inlinking web pages which also happen to satisfy the statistical significance condition. Correspondingly, we have:

$$I_c(C) = \pi_{i:i \in M, S(C,Q_i) \geq t} I(C, Q_i) \quad (7)$$

Recall that the value of t denotes the number of standard deviations by which the presence is greater than the mean for the word to be useful. Under the assumption of normally distributed data, a value of $t = 2$ results in about 95% level of statistical significance. Therefore, we chose the value of $t = 2$ consistently in all results tested.

3.3 URL Token Based Learning

The tokens contained inside a Universal Resource Locator (URL) may carry valuable information about the predicate-satisfaction behavior of the web page. The process discussed above for the content of the URL can also be applied to the tokens in the URL. For example, a URL which contains the word "ski" in it is more likely to be a web page about skiing related information. Therefore we first apply the step of parsing the URL. In order to parse the URL into tokens, we use the "." and "/" characters in the URL as the separators. We define the event R_i to be true when token i is present in the URL pointing to the candidate. As before, we assume that the event that the candidate satisfies the predicate is denoted by C. The interest ratio for the event C given R_i is denoted by $I(C, R_i)$. Let P be the set of all tokens which are present in the candidate URL. As in the case of content based learning, we define this interest ratio as follows:

$$I(C, R_i) = P(C \cap R_i)/P(C) \cdot P(R_i) \quad (8)$$

Table 1: Topic Locality Learning Information

Type of link	Expected	Actual
Pred- Pred	$P(C) \cdot P(C)$	f_1
Pred- Non-Pred	$P(C) \cdot (1 - P(C))$	f_2
Non-Pred- Pred	$P(C) \cdot (1 - P(C))$	f_3
Non-Pred- Non-Pred	$(1 - P(C)) \cdot (1 - P(C))$	f_4

The estimation of the parameters on the right hand side of this equation will be discussed in a later section. Note that not all tokens would have a useful interest ratio; for example a token such as "www" does not provide any significant information. In order to use only those tokens which have significance to a crawl, we calculate the corresponding significance factor:

$$S(C, R_i) = (P(C|R_i) - P(C))/(\sqrt{P(C) \cdot (1 - P(C))/n(C)}) \quad (9)$$

Correspondingly, we define the composite interest ratio for the token based scheme as the product of the interest ratios of all tokens in P which are above the significance threshold of t. Therefore, we have:

$$I_u(C) = \pi_{i:i \in P, S(C,R_i) \geq t} I(C, R_i) \quad (10)$$

The value of the threshold t was again chosen to be 2 for the same reasons as discussed in the previous subsection.

3.4 Link Based Learning

The idea in link based learning is somewhat similar to the focused crawler discussed in [6]. The focused crawler works on the assumption that there is very high short range topic locality on the web. In general, this fact may or may not be true for a given predicate. Therefore, the intelligent crawler tries to learn the significance of link based information during the crawl itself. This significance is learned by maintaining and updating statistical information about short-range topic locality during the crawl itself. Thus, if the predicate shows considerable short-range locality, the crawler would learn this and use it effectively. Consider, for example, when the crawler has collected about 10000 URLS and a fraction of $P(C) = 0.1$ of them of them satisfy a given predicate. If the linkage structure were completely random, then the expected fraction of links for which both the source and destination web page satisfy the predicate is given by 1%. In reality, because of the short range topic locality discussed in [6], this number may be much higher and is equal to $f_1 = 7\%$. The corresponding interest ratio is is given by $0.07/0.01 = 7$. Since this is greater than 1, it implies a greater degree of short range topic locality than can be justified by random behavior. In Table 1, we illustrate the different cases for a link encountered by the crawler for which both the inlinking and linked-to web page have already been crawled. The four possible cases for the pages are illustrated in the first column of the Table. The second column illustrates the expected proportion of web pages belonging to each class, if the linkage structure of the web were completely random. At the same time, we continue to collect information about the actual number of each of the four kinds of links encountered. The corresponding fractions are illustrated in Table 1.

Now, consider a web page which is pointed to by k other web pages, m of which satisfy the predicate, and $k - m$ of which do not. (We assume that these k pages have already been crawled; therefore we can use the corresponding information about their predicate satisfaction; those inlinking pages to a candidate which have not yet been crawled are ignored in the calculation.) Then, for each of the m web pages which satisfy the predicate, the corresponding interest ratio is given by $p = f_1/(P(C) * P(C)$. Similarly, for each of the $k-m$ web pages which do not satisfy the predicate, the corresponding interest ratio is given by $q = f_3/(P(C) \cdot (1-P(C))$. Then, the final interest ration $I_l(C)$ is given by $p^m \cdot q^{k-m}$.

3.5 Sibling Based Learning

The sibling based interest ratio is based on the idea that a candidate is more likely to satisfy a predicate if many of its siblings also satisfy it. (As in [13], a parent that has many children which satisfy the predicate is likely a hub and therefore a good place to find relevant resources.) For instance, consider a candidate that has 15 (already) visited siblings of which 9 satisfy the predicate. If the web were random, and if $P(C) = 0.1$, the number of siblings we expect to satisfy the predicate is $15 \cdot P(C) = 1.5$. Since a higher number of siblings satisfy the predicate (i.e. 9>1.5) , this is indicative that one or more parents might be a hub, and this increases the probability of the candidate satisfying the predicate.

To compute an interest-ratio based on this observation, we used the following rule: If s is the number of siblings that satisfy the predicate, and e the expected under the random assumption, then when $s/e > 1$ we have positive evidence that the candidate will satisfy the predicate as well. (Siblings that have not yet been visited are ignored, since we don't know whether they satisfy the predicate.) In the example above, the interest ratio for the candidate is 9/1.5=6, which suggests that the candidate is likely to satisfy the predicate. We denote this interest ratio by $I_s(C)$.

3.6 Combining the Preferences

The aggregate interest ratio is a (weighted) product of the interest ratios for each of the individual factors. Equivalently, we can combine the preferences by summing the weighted logarithms of individual factors.

$$PriorityValue = w_c \cdot \log(I_c(C)) + w_u \cdot \log(I_u(C)) + w_l \cdot \log(I_l(C)) + w_s \cdot \log(I_s(C))$$

Here the values w_c, w_u, w_l and w_s are weights which are used in order to normalize the different factors. By increasing the weight of a given factor, we can increase the importance of the corresponding priority. In our particular implementation, we chose to use weights such that each priority value was almost equally balanced, when averaged over all the currently available candidates. We will see that one of the interesting outcomes of such a strategy is that even though the different factors performed differently depending upon the nature of the predicate and the starting seed, the overall performance of the combination of factors was always superior to each individual factor. We will provide further insights on this issue in the empirical section.

4. IMPLEMENTATION ISSUES

The basic crawler algorithm is illustrated in Figure 2. The algorithm works by crawling web pages in a specific prior-

```
Subroutine Truncate-List(Priority-List, max-size);
{ Discard the last few elements of the priority list,
  so that only max-size candidates remain; }

Subroutine Reassign-Priorities(Priority-List, K)
{ Calculate the priorities of each candidate using the elements
  in K; Re-order Priority-List using the recomputed priorities
  so that the highest priority item occurs first on the list; }

Algorithm Intelligent-Crawler();
begin
  Priority-List= { Starting Seeds };
  while not(termination) do
    begin
    Reassign-Priorities(Priority-List, K);
    Truncate-List(Priority-List, max-size);
    Let W be the first element on Priority-List;
    Fetch the web page W;
    Delete candidate W from Priority-List;
    Parse W and add all the outlinks in W to Priority-List;
    If W satisfies the user-defined predicate, then store W;
    Update K using content and link information for W;
    end
end
```

Figure 2: The Intelligent Crawler Algorithm

ity order based on the self-learning information $\mathcal{K}$ collected by the algorithm during its crawl. Every time a candidate URL is crawled, it is parsed and all the unvisited candidates linked to by the URL are added to the priority list. This list has a maximum length depending upon the main memory available to the system. This value is denoted by *max-size* in Figure 2. Therefore, in each iteration, we need to truncate the list in order to take this into account. This is achieved by the procedure *Truncate-List*. (In all our experiments, we did not experience any limitations because of the truncation of the candidate list.) In order to keep track of the candidates that have been visited so far, we maintain a hash table containing the corresponding information. In addition, the priorities are re-computed and adjusted in each iteration by the procedure *Reassign-Priorities*. The algorithm may be terminated when the Priority-List is empty, or when none of the elements on Priority-List has an interest ratio high enough to indicate that any of them are interesting enough from the perspective of predicate satisfaction. This corresponds to the termination criterion of Figure 2.

The data in $\mathcal{K}$ consists of the content based learning information, the URL token information, the link based learning information and the sibling based information. Specifically, we maintain the following in $\mathcal{K}$:
(1) Number N_t of URLs crawled.
(2) Number N_c of URLs crawled which satisfy predicate.
(3) Number n_i of web pages crawled in which word i occurs.
(4) Number n_i^C of web pages crawled in which the word i occurs, and which satisfy the predicate C.
(5) Number m_i of web pages crawled in which the token i occurs in the URL.
(6) Number m_i^C of web pages crawled in which the token i occurs in the URL and which satisfies the predicate C.
(7) Number N_l of links crawled. (A link is said to have been crawled, when both the source page A and pointed-to page B of that link have been crawled. Note that B may not necessarily have been crawled using pointer from page A.)
(8) Number of links of each of the four types in Table 1. The corresponding numbers are denoted by Npp, Npn, Nnp, and Nnn respectively.

Let Q_i denote the event that the word i occurs in at least one of the inlinking web pages. Similarly, let R_i denote the event that the token i occurs in the candidate URL. Let k be the number of crawled web pages inlinking into the candidate URL, and m be the number of them which satisfy the predicate. Once the above information has been collected, the values of the key parameters are estimated as follows:

$$P(C) = N_c/N_t$$
$$P(C|R_i) = P(C \cap R_i)/P(R_i) = m_i^C/m_i$$
$$I_l(C) = \left(\frac{Npp}{N_l * P(C) * P(C)}\right)^m \cdot \left(\frac{Nnp}{N_l * P(C) * (1-(P(C)))}\right)^{(k-m)}$$

The only other quantity we need to calculate is $P(C|Q_i)$. This is the probability of predicate satisfaction given that word i occurs in one of the inlinking web pages. This is slightly cumbersome to maintain from the perspective of implementation since each time a web page is crawled, we need to analyze the content of all of its inlinking web pages. Therefore, we computed the value of $P(C|Q_i)$ using the content of the candidates themselves instead of the inlinking pages. Note that this is a heuristic fix, but we found it to be a reasonable one for the purpose of the algorithm. Therefore we have:

$$P(C|Q_i) = P(C \cap Q_i)/P(Q_i) = n_i^C/n_i \qquad (11)$$

The information corresponding to $\mathcal{K}$ is maintained either in arrays or in counters. Specifically, counters were maintained for each of the items (1), (2), (7) and (8) of $\mathcal{K}$ (as enumerated above). We also maintained arrays in order to keep track of each of the items (3), (4), (5), and (6). This is because these items require either word-specific or token-specific frequency information. Each time a web page is crawled, the corresponding counters and elements in the arrays were updated.

We used the algorithm of Figure 2 in order to traverse the candidates. We used a linked list and hash table to store the information about web pages. The linked list contains information on pages yet to be visited (i.e. candidates) such as the candidate's priority, interest ratios, and page content. Whenever we crawl a candidate page, we add the candidates linked to by that web page into a linked list. The links are added only if the web page has not already been crawled earlier. This check is made by using a lookup into the hash table. Nodes from the linked list are removed as soon as a candidate has been visited. The hash table contains information on every candidate URL which has been determined so far. This helps us in obtaining quick access to the linked list of candidates. The priorities are also maintained in the same hash table. Once the page is actually visited we mark it in the table so that we know not to fetch the page again.

5. EMPIRICAL RESULTS

The primary factor which was used to evaluate the performance of the crawling system was the *harvest rate* $P(C)$, which is the percentage of the web pages crawled satisfying the predicate. We ran experiments using the different learning factors over different kinds of predicates and starting points. This section will show that the behavior of the web is quite heterogeneous in terms of which factors affect the per-

formance of a crawler. For example, for some sets of predicates, the content based information is the most reliable, whereas for other sets the URL tokens provide the greatest amount of knowledge. The intelligent crawling model is then able to use the composite model effectively in order to discover relevant resources by learning the factors which influence the crawl most effectively. This results in a robust composite model which is relatively insensitive to the nature of the starting point and the predicate. This would tend to indicate that a model which is based on a fixed understanding of the behavior of the web (such as that discussed in [7]) has certain inherent shortcomings which a learning model can address more effectively. We note also that the method in [7] critically depends upon a *hierarchical* classifier and well chosen starting points in order to perform topical resource discovery. Without providing such information, the technique cannot find meaningful resources. The predicates used in this paper are arbitrary, and the crawler is started at very general points such as the *Yahoo!* homepage. Thus, our crawling system relies on a far more general and realistic model; the results are therefore not directly comparable with those discussed in [7]. However, for the purpose of providing a good frame of reference to the nature of the specificity of the predicates used, we have included the results of a random crawling technique. In the random crawler, the candidates are fetched in random order. Our experiments show that with the set of predicates that we used, the random crawler obtains an average hit ratio of about 1%. This effectively means that only one out of 100 randomly crawled pages are relevant to the predicate. With the intelligent crawler, we ran experiments varying not only the predicates and seeds, but also repeated each crawl once using all the factors, and once using only a single factor. This was done to study how each of the individual factors fare versus the composite.

In order to illustrate our results, we present the *lift curve* which illustrates the gradual improvement[1] of the harvest rate $P(C)$ with the number of URLs crawled. As expected, the lift curve illustrates that the crawling system is initially slow to find relevant web pages, but gradually improves in its harvest rate, as it learns the nature of the relationship between the predicate and the the different features of the inlinking/sibling web pages. An example of the lift curve is illustrated in Figure 3, in which we have averaged the performance of the crawler on the predicate sports for five different starting points. The particular predicate which we used for this case was a (flat) classifier[2] which provided a numerical value for the relevance of a given category. We assumed that the predicate was satisfied when this numerical relevance was higher than a certain threshold. In the same figure, we have also shown the performance of a random crawler. It is clear that the amount of lift is significant enough to result in a crawling system which finds a set of web pages such that the majority of them are relevant to the predicate.

Since the graph in Figure 3 is averaged over multiple runs, we have only shown the average performance for the composite case when a combination of all the factors was used. In later charts, we will also illustrate how well the different factors performed individually using individual predicates and starting points.

[1]We have named the curve in this way because of its natural improvement in terms of the harvest rate.

[2]A description of this classifier may be found in [1].

An example of an individual lift curve in which the predicate is the category "'SPORTS" and the seed is the arts subsection of yahoo.com is illustrated in Figure 4. We used a subdirectory of *Yahoo!* to show that the crawler is able to learn its way out of the arts related section and then find its way into the sports related directory within *Yahoo!*. Note that the link based interest ratio (which depends critically upon topical locality) has a slower learning curve than the content based and the token based interest ratios. An interesting aspect of this curve is that even though there is a very wide variation between the harvest rate of the different interest ratios, the composite curve outperforms any of these individual graphs. Note the behavior of the sibling based interest ratio. At the point where it improves, the crawler hits the site www.majorleaguebaseball.com. This is an example of a sports related site which has very high connectivity to other sports sites. Hence, it is only at this point that the curve tends to show a somewhat improved performance. The sibling based interest ratio performs lower than the other factors on average, but it still outperforms the random crawler. In many cases, the tokens in the URL tended to show a similar behavior when a token which has high relevance to the predicate was encountered. For instance, in one experiment with starting point www.salon.com and category "SPORTS" as the predicate, our crawler eventually hits sites with tokens such as "sports" in the URL. When that happens, the crawler is likely to quickly learn the close relationship between the token and the actual predicate. At this point the crawler is able to efficiently leverage this information in order to quickly find pages that are very likely to satisfy the predicate based on their URLs. Of course, a crawler may not always find tokens in the URL which are very representative of its predicate-satisfaction. In such cases, the noise-thresholding technique is likely to indicate that such is the case, and it is the other factors (in the composite ratio) which will be given greater importance.

5.1 Robustness across choice of predicates

We were able to obtain consistent results across a variety of predicates. Our predicates comprised a combination of keywords, categories, and specific tokens in the URL. An example of a predicate consisting of a combination of a categorical specification and a keyword is illustrated in Figure 5. These are the results of a crawl with the initial seed www.yahoo.com and predicate corresponding to travel related sites which contain the keyword "Paris". In this case, the token based interest-ratio performed quite poorly unlike the previous case. The link based ratio performed slightly better, but did not perform as well as the composite ratio for most of the crawl. Thus, the trends were quite different from those observed in the Figure 4 in terms of the ordering of the factors which were the most important. In Figure 6, we have illustrated a crawl with the seed www.ebay.com and the predicate for automotive related sites which contain the keywords "Toyota" and "SUV". One of the striking aspects of this particular predicate was that the linkage based predicate showed very low harvest rates because of poor topical locality. Thus, this is an example of a predicate in which there is no short-range topical locality. The other factors seemed to show good harvest rates, as did the composite crawling technique. Thus, the results discussed in Figures 4, 5, and 6 reinforce the fact that different factors contribute differently depending upon the nature of the un-

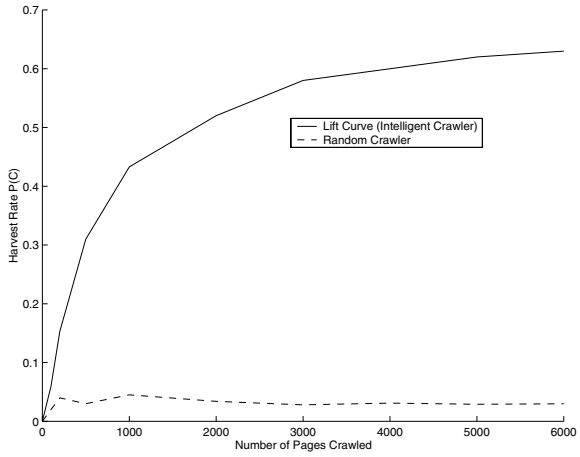

Figure 3: An Averaged Lift Curve

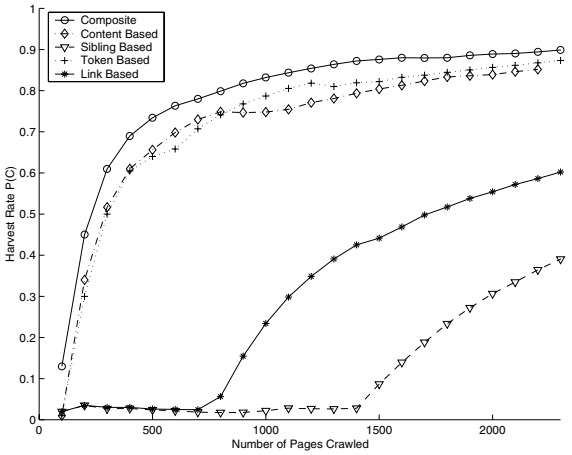

Figure 4: Seed is www.yahoo.com/ARTS and predicate is category "SPORTS"

derlying predicate. At the same time, the use of all factors in creating a composite learning ratio shows considerable robustness across the different crawls.

5.2 Robustness across choice of seeds

One of the interesting observations was that the relevance of the different factors was sometimes even sensitive to the choice of the seed from which a crawl was started. An example of such a case is illustrated in Figures 7 and 8, in which the predicate corresponds to the "SPORTS" related category, whereas the initial starting seeds are www.amazon.com amd www.ucla.edu respectively. In this case, it is clear from the two figures that the ordering of the harvest rates for the linkage based and content based ratios is reversed.

In spite of this, the composite crawling system is quite robust to the use of different seeds. In Figure 9 we show a comparison of several crawls with the predicate "SPORTS" but using different seeds. As is seen in Figure 9, four of the five starting points lead to similar behavior when all factors were used. Even the one seed (www.amazon.com) which does not perform quite as well as the others yields a harvest rate of around 40% which is quite effective in terms of the overall harvest rate.

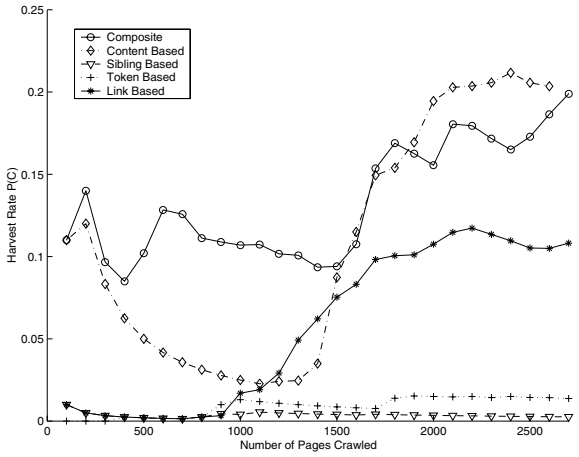

Figure 5: Seed is www.yahoo.com and predicate "TRAVEL related sites containing keyword Paris"

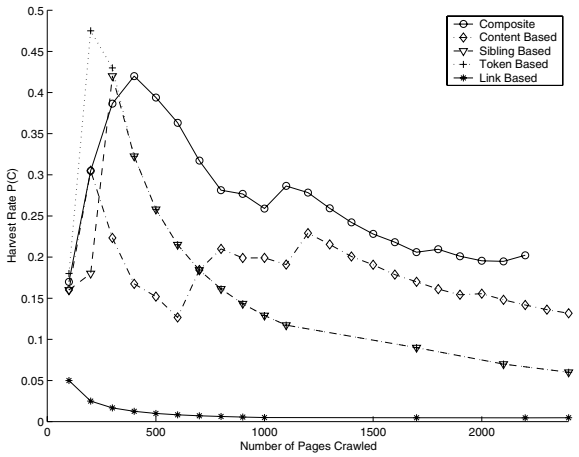

Figure 6: Seed is www.ebay.com and predicate "AUTOMOTIVE related sites that contain keywords Toyota SUV"

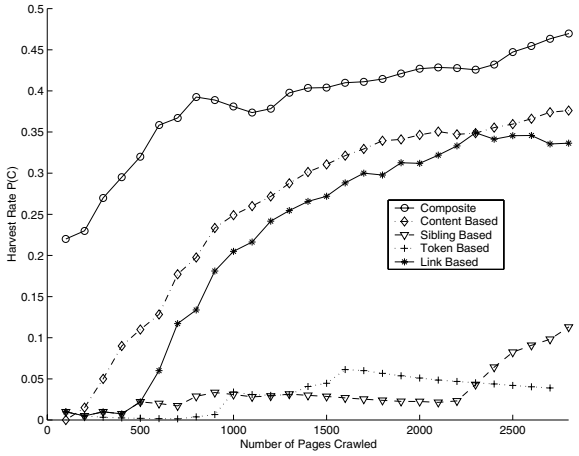

Figure 7: Seed is www.amazon.com and predicate is category "SPORTS"

Table 2: Performance of Different Kinds of crawls

Predicate	Harvest Rate	Random	Starting Seed
Keyword="shopping", URL Keyword=".uk"	36%	0.4%	uk.yahoo.com
Category=Automobiles, URL Keyword=".uk"	41%	0.2%	uk.yahoo.com
Category = Sports, Keyword="baseball", URL Keyword=".com"	33%	0.07%	www.yahoo.com
Category=Automobiles, Keyword="Dealer"	39%	0.09%	www.yahoo.com

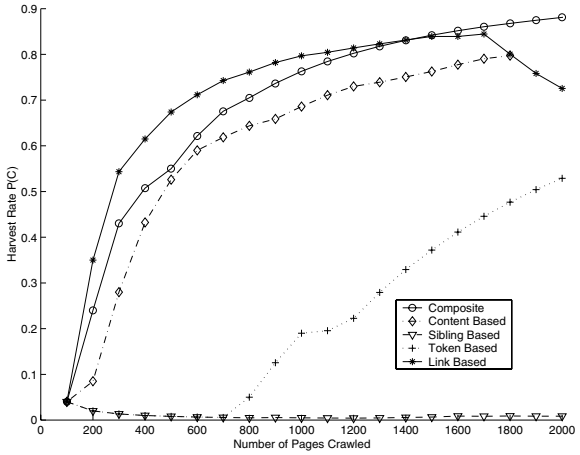

Figure 8: Seed is www.ucla.edu and predicate is category "SPORTS"

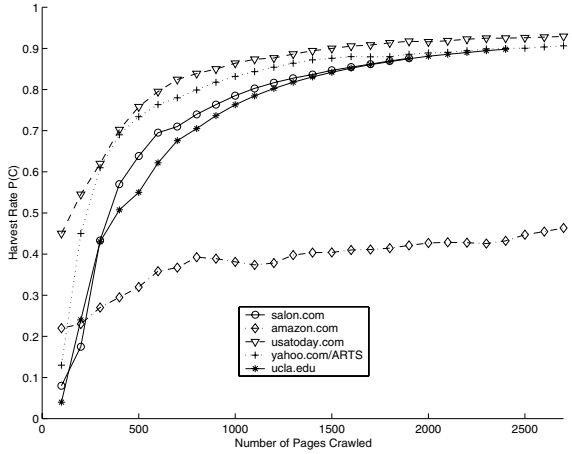

Figure 9: Comparison of several crawls with the predicate "SPORTS" but using different seeds

In addition we tested the crawling system with a wide variety of predicates and starting points and ran the system for about 10000 page crawls in each case. We present the summary results in Table 2. Note that the predicates (first column of table) consist of widely varying structures depending upon different kinds of constraints such as tokens in the URL (denoted by URL Keyword), keywords in the web page, the use of a classifier or any combination of the above. The resulting harvest rate for the crawl is illustrated in the second column. In each case, it is much higher than the performance of a random crawler and is usually between 0.25% and 0.5%. Such a harvest rate illustrated that the intelligent crawler is an efficient option for finding highly specific resources.

5.3 Behavior of the composite factor

An observation from the previous charts is the behavior of the composite factor as related to the individual charts - in each case, the composite factor performs almost as well as or better than the best of the four factors. This is a very desirable property, though it is also somewhat intriguing- after all, if the composite factor is a weighted average of the four priorities, then why should the crawling behavior also not be averaged? This is because at different times in the crawl, different factors may show high evidence of predicate satisfaction, whereas the other factors may show little or no evidence. However, the composite factor will typically be most highly influenced by the factor which shows an interest factor significantly larger than 1, whereas the other factors will not influence the priority value as much. This is especially the case for the content and URL token based learning methods which have noise-thresholding techniques built in. Thus, at different points in the crawl, the composite is being created by a domination of some individual factor which is the best *for that moment in time*. This results in an overall cumulative advantage over any single factor when evaluated throughout a period of crawling.

5.4 Reuse of Learning Information

Since the world wide web is a dynamic entity, it may often be the case that the same queries are executed repeatedly over a period of time. In focused crawling [7], every new crawl must start from scratch with the same amount of information. Since a learning crawler collects information which relates the features of the inlinking/sibling pages of a candidate to its predicate satisfaction probability, we expected that if we were to reuse data collected from previous crawls, the crawler's initial learning effort could be reduced. An example of such a case is illustrated in Figure 10. Here the predicate is the SPORTS category and the starting seed corresponds to www.usatoday.com. It is clear that the curve which reuses the information from the previous crawl starts off with a very high predicate satisfaction

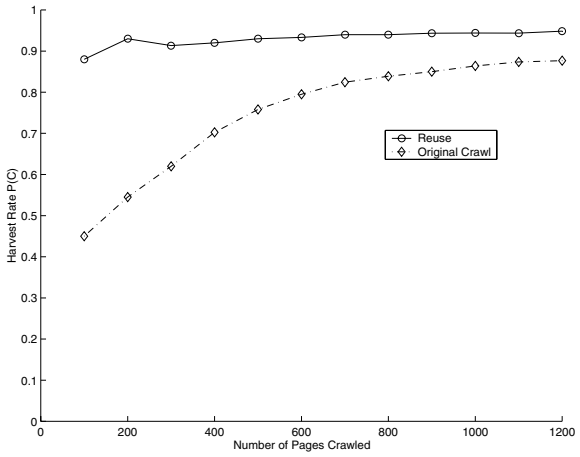

Figure 10: Example of information reuse where the seed is www.usatoday.com

probability. This high predicate satisfaction in the beginning saves a considerable amount of initial effort in a slow learning phase which has a low harvest rate. The ability to reuse such information has many other potential applications, such as the use of the knowledge learned from one starting seed to a crawl with a different seed and closely related predicate. For example, we did one query on shopping related sites which contained the keyword "books". We performed a crawl starting from the seed www.yahoo.com and stored the learning information. Then we performed a second crawl, except that in this case we modified the predicate a little. In this case, the predicate was constrained to have the extension ".uk" in its URL and the crawl was started from the UK (uk.yahoo.com) homepage of *Yahoo!*. Thus, our second predicate is interested only in shopping books in the United Kingdom. The crawl from the UK section of *Yahoo!* was started in two ways: **(1)** Using the information learned from the crawl starting at www.yahoo.com and **(2)** Without using this information. Upon collecting the data from the first 5000 web pages which were crawled, we found that the crawler which used the learning information showed a harvest rate of 42%, whereas, the crawler which did not use the information showed a harvest rate of only 28%. Thus, there is a clear advantage in the use of a learning crawler, in which the memory from each crawl can be effectively unleashed for other crawls. Clearly, such leverage cannot be used in crawling techniques such as those discussed in [6, 7] because of their inherently non-learning and static nature.

6. CONCLUSIONS AND SUMMARY

In this paper, we proposed an intelligent crawling technique which uses a self-learning mechanism that can dynamically adapt to the particular structure of the relevant predicate. A crawler which is intelligent enough to learn the predicate-dependent features is inherently more flexible in retaining effectiveness. Several factors are used during the crawl in order to evaluate its effectiveness, including the content of the web page, the URL tokens, the linkage structure, and the URL behavior. We show that different factors are more relevant for different predicates. Based on these different factors, we were able to create a composite crawler which is able to perform robustly across different predicates. We also illustrated some advantages of the learning crawler in terms of its ability to reuse the learning information from a given crawl in order to improve subsequent crawls.

7. REFERENCES

[1] C. C. Aggarwal, S. C. Gates, P. S. Yu. On the merits of using supervised clustering for building categorization systems. *SIGKDD Conference*, 1999.

[2] K. Bharat, M. Henzinger. Improved Algorithms for Topic Distillation in a Hyperlinked Environment. *ACM SIGIR Conference*, 1998.

[3] P. De Bra, R. Post. Searching for Arbitrary Information in the WWW: the Fish-Search for Mosaic. *WWW Conference*, 1994.

[4] S. Chakrabarti, et al. Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text. *WWW Conference*, 1998.

[5] S. Chakrabarti et al. Mining the Web's Link Structure. *IEEE Computer*, 32(8):60-67, August 1999.

[6] S. Chakrabarti, M. van den Berg, B. Dom. Focussed Crawling: A New Approach to Topic Specific Resource Discovery. *WWW Conference*, 1999.

[7] S. Chakrabarti, M. van den Berg, B. Dom. Distributed Hypertext Resource Discovery through Examples. *VLDB Conference*, 1999.

[8] M. Diligenti et al. Focused Crawling Using Context Graphs. *VLDB Conference*, 2000.

[9] J. Ding, L. Gravano, N. Shivakumar. Computing Geographical Scopes of Web Resources. *VLDB Conference*, 2000.

[10] Z. Bar-Yossef et al. Approximating Aggregate Queries about Web Pages via Random Walks. *VLDB Conference*, 2000.

[11] J. Cho, H. Garcia-Molina, L. Page. Efficient Crawling Through URL Ordering. *WWW Conference*, 1998.

[12] J. Cho, H. Garcia-Molina. The Evolution of the Web and Implications for an Incremental Crawler. *VLDB Conference*, 2000.

[13] J. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *SODA*, 1998.

[14] S. Mukherjea. WTMS: A System for Collecting and Analyzing Topic-Specific Web Information. *WWW Conference*, 2000.

[15] http://www.yahoo.com

[16] http://www.altavista.com

[17] http://www.lycos.com