

# Focused Web Crawling: A Generic Framework for Specifying the User Interest and for Adaptive Crawling Strategies

Martin Ester, Matthias Groß, Hans-Peter Kriegel

Institute for Computer Science, University of Munich

Oettingenstr. 67, D-80538 München, Germany

email: {ester | gross | kriegel}@dbs.informatik.uni-muenchen.de

## Abstract

Compared to the standard web search engines, focused crawlers yield good recall as well as good precision by restricting themselves to a limited domain. In this paper, we do not introduce another focused crawler, but we introduce a generic framework for focused crawling consisting of two major components: (1) Specification of the user interest and measuring the resulting relevance of a given web page. The proposed method of specifying the user interest by a formula combining atomic topics significantly improves the expressive power of the user. (2) Crawling strategy. Ordering the links at the crawl frontier is a challenging task since pages of a low relevance may be on a path to highly relevant pages. Thus, tunneling may be necessary. The explicit specification of the user interest allows us to define topic-specific strategies for tunneling. Our system Ariadne is a prototype implementation of the proposed framework. An experimental evaluation of different crawling strategies demonstrates the performance gain obtained by focusing a crawl and by dynamically adapting the focus.

## 1 Introduction

The world wide web continues to grow at an amazing speed. In the beginning of 2000, the number of available web pages exceeded one billion [Ink00], and will exceed three billion in 2001, assuming a daily increase of 3 million pages. Although the large web search engines maintain huge indexes, they cover only a relatively small portion of the whole web, for example approximately 330 million

pages in the case of Northern Light [Nor01]. Therefore, the *recall* of a web search engine, i.e. the ratio of the number of relevant pages returned to the number of all relevant pages on the web, is typically rather low. Furthermore, many of the returned web pages are out of date or their URLs may even be invalid. Another problem is low *precision*, which is the ratio of the number of relevant answers to the number of all answers. This is due to the fact that the keywords specified by the user may occur in several contexts and most of them may be irrelevant to the user. Consequently, often a large number of irrelevant web pages is returned together with a relatively small number of relevant pages.

These drawbacks of web search engines result from their claim to cover the whole web and serve queries concerning all possible topics. Focused crawlers overcome the above drawbacks of web search engines, i.e. they yield good recall as well as good precision by restricting themselves to a limited domain. They do not rely on large static indices of the web but crawl the current content of the web. A *focused crawler* [CBD99a] takes a set of well-selected web pages exemplifying the user interest. Searching for further relevant web pages, the focused crawler starts from the given pages and recursively explores the linked web pages. While the crawlers used for refreshing the indices of the web search engines perform a breadth-first search of the whole web, a focused crawler explores the web using a best-first search according to the user interest. A focused search in the web graph is possible because hyperlinks often connect two web pages dealing with the same topics. More precisely, two web pages connected by a hyperlink have a much higher probability of being relevant for the same topics than two randomly chosen web pages.

A major problem of a focused crawler is to effectively order the links at the *crawl frontier* (the out-links of the already visited web pages) so that a maximum number of relevant pages is loaded while loading only a minimum number of irrelevant pages. This is a challenging task since pages of a low relevance may be on a path to highly relevant pages. Thus, it may be temporarily necessary to visit moderately relevant pages before again finding very relevant ones, i.e. *tunneling* may be necessary. In the long run, the

precision even of the best focused crawler degenerates because for most of the topics there is only a small percentage of relevant pages on the whole web. Therefore, a focused crawler should be able to decide when to stop which is, however, beyond the capabilities of current systems.

Another open problem is the adequate specification of the user interest. The well-known focused crawlers allow the user to choose a set of topics which is implicitly connected by the OR operator. We argue that more general formulae using also the AND as well as the NOT operator, which are standard for web search engines, are more appropriate also in the context of focused crawling. Clearly, methods for measuring the relevance of a web page with respect to such complex formulae have to be developed.

## 1.1 Related Work

[CGP98] discusses the question how to order the links at the crawl frontier in order to obtain more important pages first. Several metrics exploiting the text of the source page, the anchor text of the respective link and the URL of the destination page are introduced. A basic idea is that the anchor text may be a reasonable predictor for the text of the destination page. In this approach, the user interest is specified by a so-called driving query and the relevance of a page is measured by calculating the similarity between the driving query and this page. A major limitation of this approach is that there is no explicit strategy of tunneling.

[RC99] uses reinforcement learning to train a crawler how to tunnel effectively. Relevant pages are considered as immediate rewards while relevant pages on a path via the current web page are future rewards. Based on example web sites containing relevant pages, the crawler learns to choose the next link such that the reward over time is maximized. Note that the learning method requires large collections of already visited web pages and hyperlinks. Thus, training is performed off-line which severely limits the applicability of this approach to focused crawling.

[CBD99a] and [CBD99b] present a generic architecture of a focused crawler with a classifier and a distiller as the major components. The user interest is specified by a set of example pages from which the system proposes a set of adequate categories in a given taxonomy. The relevance of visited web pages is measured by the text classifiers of the categories. The distiller supports different strategies of ordering the links at the crawl frontier, for example the relevance of the text and the “centrality” of the source page are used for this purpose. To measure the centrality of a page, the concepts of authority rank and hub rank proposed by [Kle98] are modified. To consider the topics of the focused crawl, the hyperlinks are initially weighted by the relevance of their destination page. The distiller has to re-evaluate the ordering of the crawl frontier on substantial changes in the crawl graph, i.e. it has to re-calculate the authority ranks and hub ranks. While the ordering method using the central-

ity measures implements an implicit strategy of tunneling, this approach does not include an explicit strategy of tunneling.

[DCL+00] introduces context graphs as a means to model the paths leading to relevant web pages. Context graphs represent typical link hierarchies within which relevant web pages occur together with the text content of such pages. A drawback of this approach is the fact that it relies on the indices of a web search engine in order to obtain the direct and indirect in-links of the start pages for learning the context graphs. The index of a search engine, however, might be unavailable, not cover the relevant topics or be out of date.

There is also some related work on developing learning strategies to improve the precision of standard web search engines. Systems such as Syskill & Webert [PMB96] or WebWatcher [JDM97] learn from the way a user processes the obtained web documents in order to re-rank these answers or to find additional relevant documents. Recently, [DLCT00] presented an approach to learning based web query processing. To bridge the gap between the issued keywords and the real user interest, the system learns from the user while he browses through the first answers. Furthermore, the system has the capability of finding relevant pages in the neighborhood of web pages containing the specified keywords.

## 1.2 Contributions of this Paper

In this paper, we do not introduce another focused crawler, but we introduce a generic framework for focused crawling and present a prototype implementation together with an experimental evaluation. The proposed framework consists of two major components:

- Specification of the user interest and measuring the resulting relevance of a given web page.  
Basically, the user interest is expressed by a logical formula using topics from a given taxonomy as literals. To measure the relevance of a page, text classifiers measure the relevances for the basic topics which are then combined according to the formula of user interest.
- Crawling strategy.

We propose a general algorithmic scheme for crawling strategies and discuss its components. In order to rank the links at the crawl frontier, we exploit the HTML meta data (anchor texts and URLs of the outlinks). The explicit specification of the user interest allows us to define topic-specific strategies for tunneling which form another important aspect of the crawling strategy. Furthermore, we show how the formula of the user interest can also be used to derive criteria for stopping the crawler.

By its design, our generic framework is more powerful and flexible than previously known focused crawlers which can be considered as instances of this framework. Our system *Ariadne* (Algorithm Regarding Interesting Anchortext for Directed Neighborhood Expansion) is a prototype implementation of this framework. In our experimental evaluation, we compare different instances to demonstrate the per-

formance gain obtained by different strategies for focusing a crawl and dynamically adapting the focus.

The paper is organized as follows. In section 2, we introduce our framework for specifying and measuring the user interest. The crawling strategy is presented in section 3. In section 4, some important aspects of our implementation in the Ariadne system are discussed. The results of an experimental evaluation are reported in section 5. Section 6 summarizes our contributions and outlines interesting directions for future research.

## 2 Specifying and Measuring the User Interest

### 2.1 Basic Assumptions and Notations for Web Crawling

We identify a web page  $p$  by its URL. Let  $t$  denote the time. Then, by  $content: (p,t) \rightarrow \sigma \in \Sigma^*$  we denote the string we receive when we try to download  $p$  at time  $t$ . This may be  $\emptyset$  when there was no response, an error message (the number of those pages is very large!), or the “real” content of the page. It may contain HTML tags or Java script or even be a binary file. Thus we treat frames as separate pages and do not immediately follow redirects. For reasons of simplicity, we assume the web to be constant over the time a crawl takes, and therefore we drop the dependency on the time. This is no limitation when the crawl runs on a snapshot of the web, or stores the accessed pages in a local database.

Furthermore, we assume a function  $text: \Sigma^* \rightarrow T \cong N^d$  which transforms a string (for example, the contents of a web page) into a  $d$ -dimensional feature vector. Typically, all terms occurring in a collection of texts are counted, stop-words are excluded and the remaining terms are used as dimensions of the resulting feature vectors [CBD99a]. Note that feature vectors are not restricted to texts but are also the standard approach for representing multimedia databases.

Let  $\Lambda(p)$  be the set of all links  $(p, q_1, a_1), (p, q_2, a_2), \dots, (p, q_n, a_n)$  from  $p$  to pages  $q_i \neq p$  with anchor texts  $a_i$ . The link  $(p, q, a)$  has the *source page*  $p$  and the *destination page*  $q$ . As we want to identify  $q_1$  with an absolute URL, we replace relative URLs by absolute URLs. Links within the same page are ignored, and the “#...” postfixes are removed.  $(p, q_1, a_1)$  and  $(p, q_2, a_2)$  are only regarded as equal, if  $q_1 = q_2$  and  $a_1 = a_2$ .

We define the world wide web as a directed graph  $G = (V,E)$  with  $V$  being the set of all existing web pages (i.e. all  $p$  for which  $text(p) \neq \emptyset$ ). Because there do exist broken links  $(p, q, a)$  with  $text(q) = \emptyset$ , we cannot define  $E$  as the union of all  $\Lambda(p)$  for  $p \in V$  yet. Therefore, we add pages with empty content for all such broken links to the set

$V$ . Thus, we can speak of a link  $(p, q, a)$  if we just know the page  $p$ , but do not yet know whether  $q$  exists.

For each page  $p$ ,  $host(p)$  is the substring of the URL of  $p$  between the protocol and the file section. So this may be a domain or a subdomain. We do not consider the IP address for defining the host. A link  $(p, q, a)$  is called *intrinsic*, if  $host(p) = host(q)$ , otherwise *transversal* [Kle1].

*Crawling* is the act of transforming a subgraph  $G_0$  of  $G$ , given by  $E_0 = \{p_1, \dots, p_n\}$ ,  $n \geq 1$ , and  $V_0 = \emptyset$ , into subgraphs  $G_1, G_2, \dots, G_M$  successively, such that in each step exactly one edge and at most one vertex from  $G$  is added to  $G_i$  to obtain  $G_{i+1}$ .  $E_0$  is called the set of *start pages*. When we talk about *focused crawling*, we assume at least two classes of web pages, *good* ones and *bad* ones with respect to some user interest. The set of start pages  $E_0$  should (mainly) consist of good pages, and the goal of a focused crawler is to find as many good pages as possible, while adding as few bad pages as possible to the subgraph. Thus, the *precision* is the most important effectivity measure of a focused crawler.

### 2.2 Specifying the User Interest

Taxonomies are widely used to organize large collections of documents, e.g. in web directories such as Yahoo!. A *taxonomy* is a tree that consists of nodes each representing a different topic which is usually defined by a set of sample documents. Note that the sample documents may be web pages but also any other text documents such as scientific articles or news. To measure the relevance of further web pages with respect to a given topic, typically a classifier is trained from the sample documents of the respective node. The special node *ANY* forms the root of the taxonomy. The edges represent the subtopic relationship and induce a partial order  $\leq$  on the nodes and on the corresponding topics. Figure 1 depicts a typical part of the taxonomy used for our experiments.

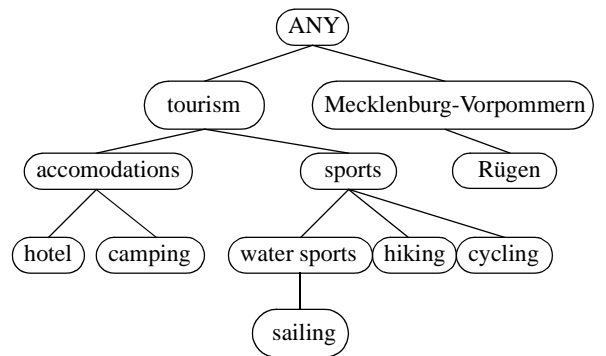
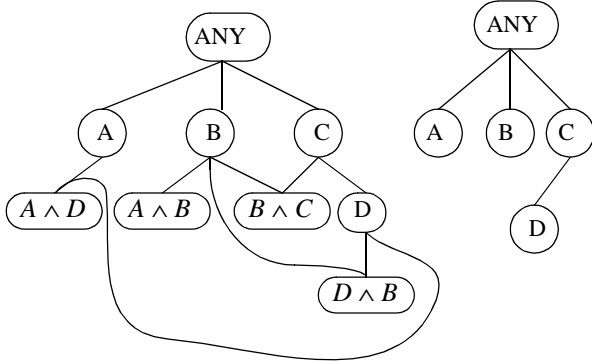


Figure 1: Part of our taxonomy used in the experiments.

In existing focused crawlers using a taxonomy, the user interest is defined by marking some nodes as “good” which is interpreted as a disjunction of the marked topics. The in-

tersection of topics has to be modelled explicitly in the taxonomy. Thus, the tree structure is replaced by a complex graph structure, see figure 2. Alternatively, redundant extra nodes may be added to the taxonomy. Obviously, the more nodes there are, the more classifiers have to be trained, maintained, and evaluated.



**Figure 2:** Example of a complex topic graph in comparison to our flat taxonomy.

In this paper, we propose a more general formalism to express the user interest. The interest can be specified by a formula using the logical operators  $\wedge$ ,  $\vee$  and  $\neg$  and the topics which are part of a given taxonomy. We want to use the  $\wedge$  operator to model the intersection of two topics instead of adding it as an additional node to the taxonomy. Thus, the taxonomy is kept flat while the number of possible user formulae is increased.

### 2.3 Measuring the User Interest

To measure the relevance of a web page for the user interest given by a formula  $\Phi$ , we need to quantify the elementary relevances for the single topics and then to transform the formula into an arithmetic expression which yields the total relevance of the page.

There are several possibilities to construct evaluation functions for the single topics, e.g. by giving example documents and training classifiers from them. It is also possible to build evaluation functions “by hand”, based on domain knowledge. Such functions can be regarded as scoring functions for weighted keyword queries. Thus, our model can also be considered as a generalization of this most frequent query type to search the web.

For learning evaluation functions (classifiers) from example documents, we denote by  $exd(A) \in 2^{\Sigma^*}$  the set of example documents for a taxonomy node  $A$ , which may be empty only for inner nodes, and by  $exi(A)$  the set of indirect example documents, defined as

$$exi(A) = \bigcup_{B \leq A} exd(B).$$

The classifier related to node  $A$  is based on the set of all terms which appear in any document in  $exi(A)$  and assigns to each document the relevance for belonging to one of the

son nodes of  $A$ . To avoid pathologies of the classifiers, we require that  $exi(A)$  and  $exi(B)$  are disjoint if  $A$  and  $B$  are not comparable with respect to  $\leq$ .

To conclude, for each topic  $A$  of the taxonomy we define an evaluation function

$$\mu_A: T \rightarrow [0, 1]$$

with properties

$$\begin{aligned} \mu_A(t) &= 1 \text{ for } A = ANY, \\ \mu_A(t) &= \text{large for } t \in exi(A), \\ \mu_A(t) &= \text{small for } t \in exi(\neg A). \end{aligned}$$

Note that the evaluation functions can be applied to any texts represented by  $d$ -dimensional feature vectors, for example to the text content of a web page, to the anchor text of a link or to the URL of its destination page.

We assume that the “large” and “small” values that  $\mu_A$  and  $\mu_B$  take on documents are of comparable order. Otherwise problems may occur when evaluating formulae contain  $\neg$ .

Now we define the evaluation function for an arbitrary user formula  $F$ ,  $\mu_F: T \rightarrow [0, 1]$ .

If the formula consists only of one topic,  $F = A$ , then  $\mu_F(t) = \mu_A(t)$ . Otherwise  $\mu_F(t)$  is recursively defined over the structure of the formula, using arithmetical interpretations of the logical operators. Because disjointness and independence of the topics are hard to prove (and even to define), we tend to a possibilistic instead of a probabilistic model.

In possibility theory [DLP94],  $\wedge$ ,  $\vee$  and  $\neg$  are usually interpreted by

$$\begin{aligned} \mu_{\neg F}(t) &= 1 - \mu_F(t), \\ \mu_{F \wedge G}(t) &= \min\{\mu_F(t), \mu_G(t)\}, \\ \mu_{F \vee G}(t) &= \max\{\mu_F(t), \mu_G(t)\}. \end{aligned}$$

Other implementations are possible as well, e.g. using weights (see [Fag99]).

The result of applying an evaluation function to some text is called the *score* of this text with respect to the corresponding formula.

Finally, a web page is called *relevant* with respect to some formula  $F$ , if its score is at least as high as the minimum of the strictly positive scores of the start pages of the focused crawler.

There are some problems with  $\neg$  in possibilistic approaches [Elk94]. Therefore, we concentrate on formulae without  $\neg$  in this paper. To run a focused crawl for  $A \wedge (\neg B)$ , we propose to crawl for  $A$  and to exclude pages with a considerable score for  $B$  afterwards.

The proposed method of specifying the user interest by a formula combining atomic topics significantly improves the expressive power of the user. Roughly speaking, our approach integrates keyword-based search supported by the web search engines such as Altavista with a topic-based search used by web directories such as Yahoo!. As we will see in the next section, this novel approach is also beneficial for the crawling strategy because it allows us to derive criteria for tunneling and for terminating a focused crawl which are specific to the given user interest.

### 3 Crawling Strategy

If we already had loaded all web pages and had their scores for the single topics of the formula  $F$  stored in a local database, then we could reduce the problem of focused crawling to the following problem well-known in the database community: retrieve the  $k$  database objects with the highest scores for  $F$ . The algorithm  $A_0$  [Fag99] has been proposed to solve this problem and it has been shown that this algorithm is optimal for uniform distributions of the scores of the single topics. For the realistic case of non-uniform score distributions [GBK00] present an algorithm with an improved termination condition which is much more efficient.

The above algorithms, however, are not applicable in the context of a focused crawler. This is due to the fact that we have no random access and even no sequential access to the web pages but we can access a web page only if we have loaded a page linking to it. Consequently, a focused crawler cannot efficiently obtain lists of web pages sorted by descending scores (for the atomic topics) and then merge the top pages from these lists in a clever way (to obtain the best pages for the complete formula). Instead, we have to estimate the score of the destination page of a link and of other pages reachable via that page without loading the page. This is a central problem of the crawling strategy which is presented in section 3.

#### 3.1 A General Algorithmic Scheme

In this section, we propose an algorithmic scheme (see figure 3) to describe a wide class of focused and non-focused crawling strategies. It can also be used to compare different approaches more easily.

We assume that the input of the crawler consists of a set of start pages and a formula  $\Phi$  by which the user interest is defined, together with a corresponding taxonomy, see section 2. For all topics in the taxonomy, the evaluation functions are given. We further assume that most of the start pages have a strictly positive score with respect to the formula  $\Phi$ .

In the case where we have web pages as example documents for the nodes of the taxonomy, we could allow the set of start pages to be empty and automatically choose start pages from the example documents. That works well if  $\Phi$

consists of only one topic or is a disjunction. Otherwise all example documents could have a zero score under  $\Phi$ .

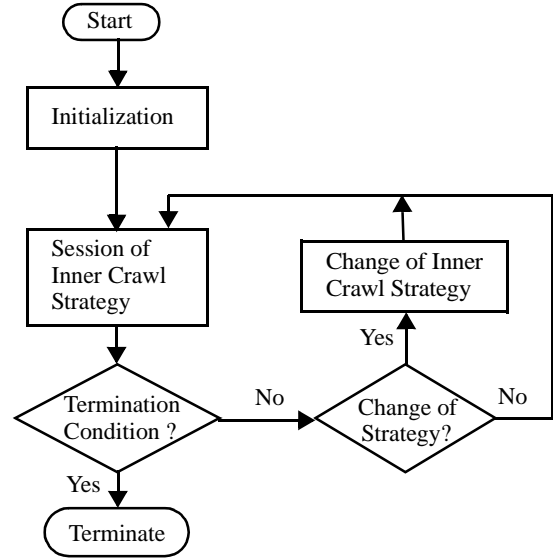


Figure 3: Algorithmic scheme of the crawling strategy

In the first step, the crawler is initialized, in particular an inner crawl strategy is chosen and the focus is set to the formula specifying the user interest. Then, the focused crawler repeats several sessions of an *inner crawl strategy* until a termination condition is met. In each session,  $N$  new web pages are processed unless the crawl frontier becomes empty - in this case the crawl stops. If the crawl is to be continued, the inner crawl strategy may be changed, in particular the focus of the crawl (see section 3.3). If the inner crawl strategy is static we call a crawler *simple*.

The following characteristics should not be changed during a crawl, or for a number of crawls to be compared:

- (a) how pages are preprocessed before applying the evaluation functions, i. e. the function  $\text{text}(p)$ ,
- (b) how the score of a page is determined,
- (c) the threshold for the score of a page to be relevant.

During a session, the score (see section 3.2) of the links at the crawl frontier may not be changed, nor the way in which it is composed.

To process a single page, the following steps have to be performed:

- obtain the link with the maximal score;
- delete all links from the crawl frontier which point to the same destination page (note that a lot of different links to this special destination page with different scores may be contained in the crawl frontier);
- download the page, if possible;
- calculate the relevance of the page (for this purpose, the page has to be converted to a feature vector) and present it to the user if the page is relevant;
- extract the outlinks, calculate their score and add them to the crawl frontier.

We can classify inner crawl strategies according to some important basic techniques:

- whether some kind of pruning is used: exclusion of certain hosts (for example because there were too many problems with that host, or pages from there used to be irrelevant), of URLs with certain syntactical or semantical properties or of links with low scores,
- how many pages are processed simultaneously - this has an implicit effect on the “tunneling” behavior,
- how the scores of the links are calculated - this is the subject of the following subsection.

### 3.2 Ranking the Links at the Crawl Frontier

The central task of the inner crawl strategy is to estimate as exactly as possible whether a link at the crawl frontier will be important for the crawl or not. In other words, we have to estimate the score of the destination page of the link and of other pages reachable via that page without loading the page. This estimate is used as the score of the respective link.

The relevance of a web page may be defined in different ways:

*direct relevance*: whether the page itself is relevant or not,  
*indirect relevance*: how many relevant pages will be reached directly or indirectly after processing that page.

The function  $\text{predict: links} \times 2^{\text{pages}} \rightarrow [0,1]$  takes a link as well as the set of the already crawled pages and returns a score such that a high (low) score indicates a high (low) relevance of the destination page.

Some methods for estimating the indirect relevance have been proposed in the literature ([CBD99a], [DCL+00], [RC99]). They can also be used in our framework to order the links at the crawl frontier. These methods, however, have their limitations as discussed in section 1.1. We follow an alternative approach of defining an explicit strategy of tunneling, see section 3.3. Therefore, we concentrate on the direct relevance and try to give a comprehensive list of criteria that can be used.

Exploiting only local information, i. e.  $\text{predict: links} \times 2^{\text{pages}} \rightarrow [0,1]$  does not depend on the content of pages other than the source page, there are the following possible criteria, some of which were already mentioned in [CGP98]:

- (a) the relevance of the source page, which is usually the relevance according to the user formula. Special sections of the page, e. g. meta tags, could be scored independently,
- (b) the relevance of the URL of the destination page,
- (c) the relevance of the anchor text of the link (and the text close to it),
- (d) the relevance of the host of the destination page. If, e.g., 15 pages from a specific host were unimportant the 16th will probably be, too; this is true even more if the path is the same, or the filename is somewhat similar,
- (e) the link distance between the start pages and the destination page. We count the link distance on the path that has lead us to a page. This could also be replaced by the

shortest possible path in the subgraph already known, which would, however, cause additional cost.

(f) the file level of the URL (intuition: `www.myhost.edu` is more likely to be important than `www.myhost.edu/somestuff/someotherstuff/etc/info/sometypicalpage.html`),

(g) whether the destination page is on a different host, i.e. whether this is a transversal link. Typically, the user is interested in getting pages from as many different hosts as possible.

(h) the number of outlinks of the source page, which indicates whether it is a hub page.

To measure the relevance of a URL or anchor text, we propose to use the same elementary evaluation functions as for page content. But as there are usually only a few terms in a URL or anchor text, one cannot expect all scores to be high in a complex formula. So we need a strategy to modify the formula  $\Phi$  for this task.

In the case of  $\Phi = A \wedge B$ , we propose to take only the topic which appears globally more infrequently, or even to take  $A \vee B$ . To estimate the frequency of a topic, either domain knowledge or an estimation based on the already crawled pages can be used.

In our experiments, we observed that the number of anchor texts containing relevant features is relatively low. Nevertheless, the benefits of considering anchor text and URLs are considerable and they are among the main reasons for the good precision of the Ariadne standard strategy (see section 5) in the beginning of a crawl.

A selection of the above criteria for direct relevance can be organized in a vector-like way, with the common  $\leq$  in each component and the lexicographic order on the whole vectors (e.g. [CBD99b]). We prefer to combine the single criteria (a) to (h) of the direct relevance by a weighted geometric mean to obtain an overall score. Setting a weight to zero implies that we ignore that criterion. Clearly, it is necessary to tune the weights for the different criteria appropriately. The links at the crawl frontier are then ordered by descending values of their overall scores.

### 3.3 How to Support Tunneling Explicitly

*Tunneling* is the phenomenon that a crawler reaches some relevant page on a path which does not not only consist of relevant pages. Thus the crawler had to “tunnel” through some irrelevant pages. This phenomenon can be prevented by a *pruning strategy* not allowing to follow the links from an irrelevant page. This strategy can have a positive effect on the precision, but it results in low recall. Therefore, we generally allow tunneling while trying to keep the precision as high as possible.

The reader may ask why an explicit strategy of tunneling is necessary. If the crawl frontier only contains links with a low score, why not just visit the best out of these relatively irrelevant pages? Breadth-first search typically yields very low precision. If a topic is relatively general, a breadth-first

strategy may be good in the beginning, but will typically degenerate in most cases from link distances around 3 or 4 onwards. Therefore, we are aiming at a good compromise between the extremes of pruning and breadth-first search.

In the following, we propose a method to efficiently control the process of tunneling. This method called *changing the focus* has to answer two questions:

- In which situations should the focus be changed?
- How is the focus of the crawl changed?

Our answer to the first question is as follows. In the long run, the precision of any crawl will decrease. If the precision, however, decreases faster than expected it is necessary to broaden the focus of the crawl. The rationale behind this method is that pages of generalized topics more frequently lead to pages of the specialized topics of the crawl than random web pages (see also [CBD99a]). For example, if a focused crawl on “sailing” temporarily finds no more relevant pages, the focus might be generalized to “water sports” and some of the pages on “water sports” will probably be linked to more pages on “sailing”.

Let us assume that the crawler has finished  $i$  sessions as described in section 3.1. Let  $s_i$  denote the percentage of relevant pages among the  $N$  pages investigated during the last session,  $0 \leq s_i \leq 1$ , and let  $r_i$  denote the percentage of relevant pages among the  $iN$  pages which have been visited during the whole crawl. Let  $a$  and  $b$  be parameters with  $0 < a \leq b \leq 1$ , e. g.  $a = 0.5$  and  $b = 0.8$ .

We use the following method to detect situations when the focus has to be changed:

- if  $s_i < a \cdot r_i$ : generalize the focus, if possible;
- if  $s_i > b \cdot r_i$ : specialize the focus, if there have been corresponding generalizations before;
- otherwise: no changes of the focus are necessary.

Thus, when the precision decreases dramatically, we are leading the crawler to more general (but still somehow related) pages. On the other hand, if the precision increases again, we may (step by step) focus on the original user interest again.

There are several options of generalizing the focus of the crawler which will be explained in the following using the formula  $\Phi = A \wedge B$ :

#### Ignore the more frequent topic

If we know that topic  $A$  is less frequent on web pages than  $B$ , we generalize from  $A \wedge B$  to  $A$  which is more frequent than  $A \wedge B$  but less frequent than  $B$ .

Experiments on a multimedia database showed that if there are two indexes on the  $A$  and  $B$  values and we run over one (sorted) attribute, starting with the maximum value, then the  $n$  best answers for  $A \wedge B$  are reached much faster if we run over the attribute with the lower average value, or, in other words, over the *infrequent* attribute.

However, different notions of infrequent are reasonable, e.g.: for which topics did we visit the smaller number of pages, that are relevant with respect to this particular topic?

Or: which topic was in most of the cases responsible for lowering the score of a page?

#### Increase the weight of the more infrequent topic

When using weights to evaluate  $\wedge$  (see [FW00]), a larger weight is given to the more infrequent topic.

#### Generalize the more infrequent topic along the taxonomy

Looking for hotels on the island Rügen, for example, we might generalize to hotels in Mecklenburg-Vorpommern, hoping that we will get some links back to Rügen later.

For this method of generalization, it is important that the taxonomy is really a tree, so that it is clear for each topic which is the more general one.

A drawback of the above approaches of broadening the focus, however, is that we need a priori knowledge about which topic is more infrequent. But it is reasonable to estimate the “critical”, i. e. most infrequent topic of the formula, based on the subgraph we have already crawled. Since this subgraph is continuously growing, the estimation should converge to the real values. Typically, the estimate converges from above, as we have started at relevant pages which are untypical in the whole web.

In comparison to generalization, specializing is relatively easy. Either the more frequent topic of the formula is specialized or simply the last generalization step is redone.

When the focus has been changed, we have basically two choices: (1) adapt the scores of all links at the crawl frontier. This will yield better results, but of course we have to store all necessary data and we have the cost for recalculating the scores. (2) Second choice: only apply the new calculation of scores to links that are added to the crawl frontier. Thus, we save space for storing the additional attributes and time for re-calculating the scores, but do not profit fully from our technique.

### 3.4 Termination

In the previous section, we have proposed a tunneling strategy to react to decreasing precision. But from some point on, to continue the crawl only means to force the precision down to the percentage of relevant pages on the whole web, which is almost zero for most topics. How can we see that it is time to stop the crawl?

Reasonable criteria for terminating a focused crawl include:

- The crawl frontier is empty. However, this case is not likely to occur.
- A specified number of relevant pages has been found.
- A specified number of irrelevant pages has been visited.
- A specified number of irrelevant pages has been visited since the last relevant page was found.
- The precision falls below some threshold.
- The maximum score of the links at the crawl frontier falls below some threshold.
- The tunneling strategy (section 3.3) decides that the focus has to be generalized, but the formula has already been generalized to the most general level.

- The relevant pages we have found contain a *community*, that is a subset of web pages that are more closely linked to each other than to pages not belonging to this subset [FLG00].

Note that only the first criterion which is not realistic to be met and the last one which is very costly to test are non-parametric. All the other criteria require at least one parameter which, in general, must be specified by the user based on his domain knowledge.

## 4 Implementation

The proposed framework for focused crawling has been implemented in our system Ariadne. It is written in Java and has the following main characteristics:

- *Text preprocessing*: for English pages, we use the Porter stemming algorithm; we have stopword lists for English and German.

- *Classifiers*: due to their robustness in cases when only the “good” pages are characterized, so far we use simple text classifiers based on semi-automatically selected keywords. A more sophisticated naive Bayes classifier [Mit97] applicable in this framework is currently being developed.

- *Link extraction*: we do not follow links with certain syntactical properties (e.g. dynamic links) or file endings like .gif or .mp3 which are not likely to lead to pages containing text. We exclude hosts from which a specified number of pages could not be downloaded. Restrictions according to robots.txt files are respected. Besides that, we do not apply any kind of pruning.

- *Relevance estimation*: the score of a link is calculated as a weighted geometric mean of the local criteria mentioned in section 3.2. The weights have been assigned manually. Since there are not many interesting anchor texts or URLs in general, the influence of the relevance of the source page is dominant in most cases. Giving positive weight only to criterion (e), we implement breadth first search.

- *Inner crawl strategy*: Our implementation uses multi-threading to exploit the full bandwidth of the internet connection. Each thread follows the currently most promising link, downloads and processes the corresponding page, immediately scoring the newly extracted links and adding them to the crawl frontier. Alternatively, we could fetch several of the most promising pages simultaneously. After changing the focus, we adapt the scores of the links, which turns out to be rather efficient.

- *Persistency*: Ariadne uses MySQL, a GPL (GNU General Public License) relational database system, to store the information collected during a crawl. There are two types of relations: cache relations and crawl relations. The *cache relations* form a structured web cache of all parts of the web we have investigated in any crawl up to now. Each page is only downloaded once. Thus, our assumption of the time invariance of the web during a crawl holds. When running

multiple experiments on the same user interest, many of the pages are already in the database. This can be regarded as evidence of the stability of the used crawling techniques. Of course, the crawler can access information on cached pages only by their URL.

The *crawl relations* are specific for each crawl. A relation named CrawlPages basically stores information on the pages we have already visited. CrawlLinks stores information on the links we still want to visit, thus representing the crawl frontier, and CrawlHosts collects some data on specific hosts, e. g. the number of timeouts while trying to contact them.

## 5 Experimental Evaluation

In this section, we present results of an experimental evaluation of our framework using the system Ariadne. Crawls for different user formulae were performed. As typical examples, we use the following two formulae which relate to the taxonomy depicted in figure 1:

$$\Phi_1 = \textit{tourism} \wedge \textit{Mecklenburg - Vorpommern},$$

$$\Phi_2 = \textit{sailing} \wedge \textit{Ruegen}.$$

While  $\Phi_1$  defines a very broad user interest, the scope of  $\Phi_2$  is rather narrow because the island of Rügen is only a small part of the state of Mecklenburg-Vorpommern (a German state located at the Baltic Sea) and sailing covers only a tiny portion of the topics of tourism. A web search engine returned some 20,000 pages for  $\Phi_1$  and only 120 pages for  $\Phi_2$ . Note that this is only a rough approximation for the total number of web pages relevant for the two formulae: on the one hand, the focused crawler returns more pages because it does not require that the names of the topics occur in the web pages, on the other hand our crawler is more restrictive than a search engine since the score of a relevant page has to exceed the minimum score of the start pages. We chose 19 relevant start pages for  $\Phi_1$  and 5 such pages for  $\Phi_2$ .

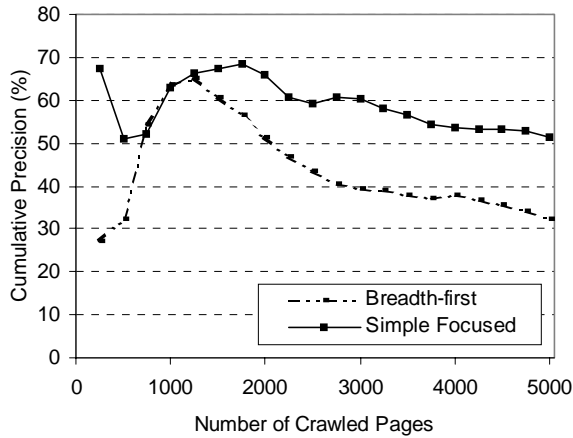
In our experiments, Ariadne was running on an HP C160 workstation with 384 MB under HP-UX 10.20, while the MySQL database server, version 9.11 for HP-UX 10.20, was running on an HP 9000/735 workstation with 256 MB. During our experiments, the cache relations together with their indexes have grown to more than 1 GB with some 402,000 page entries, 981,000 link entries and 45,000 host entries.

In this paper, we restrict our analysis to three major crawling strategies:

- Breadth-first,
- Simple Focused (without tunneling),
- Focused with Tunneling.

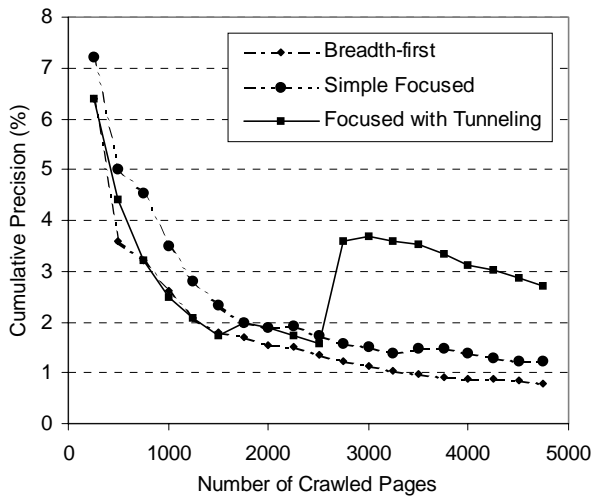
These strategies are compared with respect to their precision and with respect to the link distances of the relevant pages found.

Figures 4 and 5 depict the cumulative precision, i.e. for any number  $n$  of crawled pages the overall precision of the crawl until this point is reported.



**Figure 4:** Cumulative Precision for “Tourism AND Mecklenburg-Vorpommern”

The breadth-first crawler is surprisingly effective for the broad user interest  $\Phi_1$ , which is due to the large number of highly central start pages. Nevertheless, the simple focused crawler significantly outperforms the breadth-first strategy. Note that the simple focused crawler, for example, needs to load only 2700 pages compared to 5000 pages breadth-first requires in order to find 1627 relevant web pages. Since the precision of the simple focused crawler does not drop beyond 50 % for any session of the inner crawl strategy, the crawler will never decide to tunnel and, therefore, there is no extra graph for this strategy.

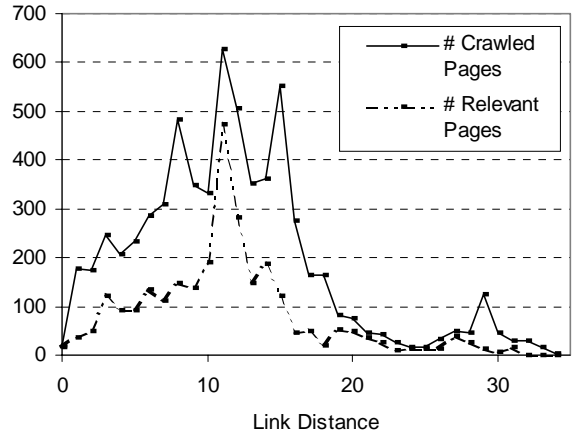


**Figure 5:** Cumulative Precision for “Sailing AND Rügen”

For the narrow user interest  $\Phi_2$ , the breadth-first crawler yields very poor precision. The simple focused crawler does not perform much better because there are only few

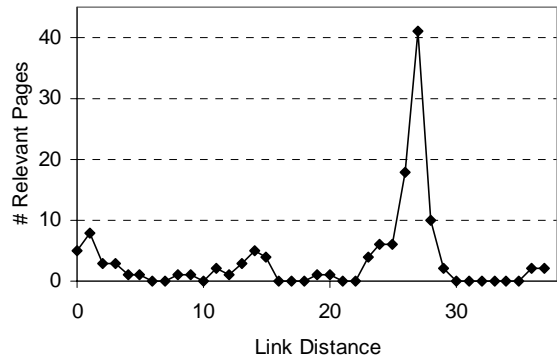
relevant pages and even fewer relevant anchor texts as well as URLs. The focused crawler with tunneling, however, obtains a significantly higher precision in the long run. The reasons for this phenomenon will be discussed below. We used the strategy of broadening the focus along the taxonomy. The tunneling strategy generalized from sailing on Rügen over water sports on Rügen, sports on Rügen, tourism on Rügen to the most general formula, tourism in Mecklenburg-Vorpommern.

At which link distances does the focused crawler find relevant pages? Figures 6 and 7 answer this question for the two different formulae.



**Figure 6:** Link Distances of Simple Focused for “Tourism AND Mecklenburg-Vorpommern”

Figure 6 demonstrates that the simple focused crawler finds most of its relevant pages beyond a link distance of 10. Note that it is not feasible to run a breadth-first crawl until this link distance if the user is only interested in a specific domain. Furthermore, we observe that the simple focused crawler keeps good precision even for very large link distances around 25.



**Figure 7:** Link Distances of Focused Crawler with Tunneling for “Sailing AND Rügen”

Figure 7 depicts the link distances of the relevant pages which the focused crawler with tunneling finds for the narrow area of sailing on Rügen. We notice that for several val-

ues of the link distance not a single relevant page was found, but for larger link distances there are again results. This implies that the tunneling phenomenon has in fact happened for several times during the reported crawl. Merely by looking at the graph, we can deduce that there exist paths to relevant pages with at least six irrelevant pages in between. Note that the peak at a link distance of 27 is due to a single host with a lot of relevant pages.

We analyzed the path of the crawler from the start pages to one of the first pages that must have been reached by tunneling (at a link distance of eleven) and we discovered that all the ten pages on this path were irrelevant for sailing on Rügen, but all of them were relevant for the fully generalized formula, tourism in Mecklenburg-Vorpommern. Thus, this relevant page would hardly have been found without generalizing the focus along the taxonomy.

Our experimental results show that an explicit strategy for tunneling is an effective means for finding lots of relevant pages for selective user formulae at a reasonable cost.

## 6 Conclusions

In this paper, we have introduced a generic framework for focused crawling consisting of two major components, the specification of the user interest and the crawling strategy. The user interest is expressed by a logical formula using topics from a given taxonomy and this explicit specification of the user interest allows us, for example, to define topic-specific strategies for effective tunneling. Our generic framework is more powerful and flexible than previously known focused crawlers and has been implemented in our system Ariadne. In an experimental evaluation, we have compared several instances to demonstrate the performance gain obtained by different strategies for focusing a crawl and for adapting the focus.

The proposed framework for focused crawling opens up a lot of promising directions for future research. User formulae containing a NOT operator should be investigated in more depth, in particular with respect to measuring the user interest and to changing the focus of a crawl. Another interesting issue is the integration and evaluation of methods for measuring the indirect relevance, for example using the hub rank and the authority rank of the web pages. Considering the different explicit strategies for tunneling, criteria should be developed which allow a focused crawler to automatically choose an optimum strategy depending on the user formula and other parameters obtained from the user. Different strategies for terminating a focused crawl have to be implemented and experimentally compared. Finally, the user interface of a focused crawler deserves more attention: how can the user adequately be supported in specifying his interest and how should the results of the crawl be presented to the user? A good presentation of intermediate results in combination with interactive facilities may enable the user to control the search in the huge web graph much more effectively.

## References

- [CBD99a] Chakrabarti S., van den Berg M., Dom B.: "Focused Crawling: a new Approach to Topic-Specific Web Resource Discovery", *Proc. World Wide Web Conference (WWW8)*, 1999.
- [CBD99b] Chakrabarti S., van den Berg M., Dom B.: "Distributed HypertextResourceDiscoveryThroughExamples", *Proc. Int. Conf. on Very Large Databases (VLDB '99)*, 1999, pp. 375-386.
- [CGP98] Cho J., Garcia-Molina H., Page L.: "Efficient Crawling Through URL Ordering", *Proc. World Wide Web Conference (WWW7)*, 1998.
- [DCL+00] Diligenti M., Coetzee F.M., Lawrence S., Giles C.L., Gori M.: "Focused Crawling Using Context Graphs", *Proc. Int. Conf. on Very Large Databases (VLDB '00)*, 2000, pp. 527-534.
- [DLP94] Dubois D., Lang J., Prade H.: "Possibilistic Logic", in Gabbay, Hogger, Robinson (Eds.): *Handbook of Logic in Artificial Intelligence and Logic Programming*, Vol. 3, Oxford University Press, 1994.
- [DLCT00] Diao Y., Lu H., Chen S., Tian Z.: "Toward Learning Based Web Query Processing", *Proc. Int. Conf. on Very Large Databases (VLDB '00)*, 2000, pp. 317-328.
- [Elk94] Elkan Ch.: "The Paradoxical Success of Fuzzy Logic", *IEEE Expert*, August 1994.
- [Fag99] Fagin R.: "Combining Fuzzy Information from Multiple Systems", *Journal of Computer and System Sciences*, Vol. 58, 1999, pp. 83-99.
- [FW00] Fagin R., Wimmers E.L.: "A Formula for Incorporating Weights into Scoring Rules", *Theoretical Computer Science*, Vol. 239, 2000, pp. 309-338.
- [FLG00] Flake G.W., Lawrence S., Giles C.L.: "Efficient Identification of Web Communities", *Proc. Int. Conf. on Knowledge Discovery and Data Mining (KDD '00)*, 2000, pp. 150-160.
- [GBK00] Güntzer U., Balke W.-T., Kießling W.: "Optimizing Multi-Feature Queries for Image Databases", *Proc. Int. Conf. on Very Large Databases (VLDB '00)*, 2000, pp. 419-428.
- [Ink00] "Web surpasses one billion documents: Inktomi/NEC press release." available at <http://www.inktomi.com>, Jan 18 2000.
- [JDM97] Joachims T., Freitag D., Mitchell T.: "WebWatcher: A Tour Guide to the World Wide Web", *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI '97)*, 1997, pp. 770-775.
- [Kle98] Kleinberg J.: "Authoritative Sources in a Hyperlinked Environment", *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 1998.
- [Mit97] Mitchell T.M.: "*Machine Learning*", McGraw-Hill, 1997.
- [Nor01] "Northern Light Technology Dramatically Expands its International Content", available at <http://www.northernlight.com>, Feb 13 2001.
- [PMB96] Pazzani M., Muramatsu J., Billsus D.: "Syskill & Webert: Identifying Interesting Web Sites", *Proc. Amer. Conf. on Artificial Intelligence (AAAI '96)*, 1996, pp. 54-61.
- [RC99] Rennie J., McCallum A.: "Using Reinforcement Learning to Spider the Web Efficiently", *Proc. Int. Conf. on Machine Learning (ICML 99)*, 1999.