Accurate and Efficient Crawling for Relevant Websites

Martin Ester
Simon Fraser University
School of Computing Science
Burnaby BC,
Canada V5A 1S6
ester@cs.sfu.ca

Hans-Peter Kriegel
Matthias Schubert
University of Munich
Institute for Computer Science
Oettingenstr. 67,
D-80538 Munich, Germany
{kriegel,schubert}@dbs.informatik.uni-muenchen.de

Abstract

Focused web crawlers have recently emerged as an alternative to the well-established web search engines. While the well-known focused crawlers retrieve relevant webpages, there are various applications which target whole websites instead of single webpages. For example, companies are represented by websites, not by individual webpages. To answer queries targeted at websites, web directories are an established solution. In this paper, we introduce a novel focused website crawler to employ the paradigm of focused crawling for the search of relevant websites. The proposed crawler is based on a two-level architecture and corresponding crawl strategies with an explicit concept of websites. The external crawler views the web as a graph of linked websites, selects the websites to be examined next and invokes internal crawlers. Each internal crawler views the webpages of a single given website and performs focused (page) crawling within that website. Our experimental evaluation demonstrates that the proposed focused website crawler clearly outperforms previous methods of focused crawling which were adapted to retrieve websites instead of single webpages.

1. Introduction

Focused web crawlers have recently emerged as an alternative to the established web search engines like Google [12]. A focused web crawler [3] takes a set of well-selected webpages exemplifying the user interest. Searching for further relevant webpages, the focused crawler starts from a set of given pages and recursively explores the linked webpages. While the crawlers used for refreshing the indices of web search engines perform a breadth-first search of the whole web, a focused crawler explores only a small portion of the web using a best-first search guided by the user interest. Compared to web search engines, focused crawlers obtain a much higher precision and return new pages which are not yet indexed. Recently, focused web crawlers have received a lot of attention in the research areas of database systems, information retrieval and data mining [2,3,5,6,8,17].

Web search engines index individual webpages. Web directories provide a more abstract view on the web, listing relevant websites for a variety of topics. A website is a linked set of HTML-documents published by the same person or institution serving a common purpose. For several applications, the information about the topics of websites allows more accurate retrieval than the information about topics of single webpages. For example, companies are represented by entire websites, not by individual webpages. As another example, when looking for the price of a new computer, it is very helpful to search only the websites of computer retailers instead of searching the whole World Wide Web (WWW).

However, using a web directory for addressing these problems has several drawbacks. Web directories offer in most cases only a very small portion of the websites that are relevant to a given topic. The given categorization might totally lack the topic a user is interested in. Last but not least, web directory services might not be up-to-date due to manual maintenance. In this paper, we therefore extend focused crawling to the search for relevant websites offering a method to significantly increase the recall of existing web directories. Additionally, such a crawler can act as an alternative approach for searching the web for topics not yet listed in any web directory.

To adopt focused crawling for website retrieval, the simplest way is to use one of the well-established methods

**Proceedings of the 30th VLDB Conference,
Toronto, Canada, 2004**

for focused webpage crawling and, in a step of post-processing, analyse the resulting webpages in order to find relevant sites. This analysis can be done by looking for relevant homepages or by applying a website classifier [11] to all pages retrieved from a given website. However, this approach is severely limited by the fact that there is no guarantee that the crawled webpages are representative of their corresponding websites.

In this paper, we argue that in order to achieve efficient and accurate website crawling the concept of websites has to be integrated into the focused crawler itself. Therefore, we introduce a novel focused crawler directly searching for relevant websites instead of single pages. The proposed focused website crawler is based on a two-level graph abstraction of the World Wide Web (WWW) representing both webpages and websites together with their links. The crawling task is divided into two major subtasks corresponding to the two different levels of abstraction:

- An *internal crawler* views the webpages of a single given website and performs focused (page) crawling within that website.
- The *external crawler* has a more abstract view of the web as a graph of linked websites. Its task is to select the websites to be examined next and to invoke internal crawlers on the selected sites.

The proposed two-level architecture allows the crawler to control the number of pages to be downloaded from each website and enables it to find a good trade-off between accurate classification and efficient crawling. Our experimental evaluation demonstrates that website classification based on the homepages is considerably less accurate than classification methods employing more than one webpage. Furthermore, we compare our prototype of a focused website crawler to a focused webpage crawler with website post-processing and show that the introduced methods of focused website crawling clearly increase the efficiency as well as the accuracy of retrieving relevant websites from the WWW. The outline of the paper is as follows. After this introduction, we briefly survey related work. In section 3, we define the task of focused website crawling and a basic solution. Section 4 presents our novel approach to focused website crawling. Section 5 reports the results of our experimental evaluation. The last section summarizes the paper and discusses several directions for future research.

2. Related Work

In this section, we discuss related work on focused crawling as well as on text and web classification. One of the first focused web crawlers was presented by [8] which introduced a best-first search strategy based on simple criteria such as keyword occurrences and anchor texts. Later, several papers such as [2] and [3] suggested to exploit measures for the importance of a webpage (such as authority and hub ranks) based on the link structure of the world-wide-web to order the crawl frontier. These measures, which are very successfully used to rank result lists of web search engines, also proved to be very effective in focusing a crawler on the topic of interest of a user.

Recently, more sophisticated focused crawlers such as [5], [9] and [17] incorporate more knowledge gained during the process of focused crawling. [9] introduced the concept of context graphs to represent typical paths leading to relevant webpages. These context graphs are used to predict the link distance to a relevant page and, consequently, are applied to order the crawl frontier. [17] explored a reinforcement learning approach, considering the successful paths observed, to weight the links at the crawl frontier based on the expected number of relevant pages reachable. [5] extends the architecture of a focused crawler by a so-called apprentice which learns from the crawler's successes and failures and is later consulted by the crawler to improve the ratio of relevant pages visited. Like a human user, the apprentice analyses the HTML structure of a webpage to judge the relevance of the outlinks of this page. To the best of our knowledge, all focused crawlers presented in the literature search for individual webpages and not for whole websites. The only site-oriented features of established page crawlers are the measures to prevent so-called spider traps and the prevention of host-to-host reinforcement proposed by Bharat and Henzinger [1]. A spider trap is an infinite loop within the WWW that dynamically produces new pages trapping a web crawler within this loop. A common approach to avoid most spider traps limits the maximum number of pages to be downloaded from a given website in order to escape the trapping situation [6]. However, these crawlers do not have any means to control the search within a website.

Most text classification algorithms rely on the so-called vector-space model. In this model, each text document is represented by a vector of frequencies of the most relevant terms. Due to the typically high dimensionality of the vector space, most frequencies are zero for any single document and many of the standard classification methods perform poorly. However, methods that do not suffer so much from high dimensionalities have been very successful in text classification, such as naive Bayes [19], support vector machines [14,19] or centroid based text classification [13]. An increasing number of publications especially deal with the classification of webpages. In particular, several methods have been proposed to exploit the hyperlinks for improving the classification accuracy. [7] introduces several methods of relational learning considering the existence of links to webpages of specific classes. [4] presents techniques for using the class labels and the text of neighboring webpages.

Most existing methods aim at classifying single webpages, not complete websites. [11] introduced the problem of website classification and presented several

methods based on the representation of a website as a labeled website tree where the labels are drawn from a set of page classes. However, all methods require that the user defines an appropriate (for the task of website classification) set of page classes and provides sufficient numbers of training webpages for each of these classes. We argue that this large overhead is a major obstacle for practical applications. Furthermore, [11] only deals with the classification of given websites, but does not present a focused website crawler. While it suggests a simple search strategy for exploring a given website, this strategy is not suitable as an "internal crawl strategy". Furthermore, there is no "external crawl strategy" and no discussion of the interaction between these components. Though the approach described in [15] overcomes the first problem of defining page classes it does not treat any aspects of focused web crawling.

3. The Task of Focused Website Crawling

3.1. A Graph-Oriented View of the WWW

We identify a webpage p by its URL. Then, $content(p) \rightarrow \sigma \in \Sigma^*$ denotes the string we receive when trying to download p. Furthermore, we assume a function $f: \Sigma^* \rightarrow T \cong N^d$ which transforms a string (for example, the contents of a webpage) into a d-dimensional feature vector. Let $\Lambda(p)$ be the set of all links $(p,q_1), (p,q_2), \ldots, (p,q_n)$, from p to $q_i \neq p$. The link (p,q) points from the *source page* p to the *destination page* q. Links within the same webpage are ignored. We define the *webpage graph* as a directed graph $G= (V,E)$ with V being the set of all existing webpages (extended by a special element which is needed to represent broken links) and E being the union of $\Lambda(p)$ for all $p \in V$.

The goal of focused *website* crawling is to retrieve new relevant websites from the WWW. A website is a linked set of webpages that is published by the same person, group or institution and usually serves a common purpose, e.g. to present a whole organization or company. Unfortunately, this intuitive definition is not well suited for automatic retrieval. Since there is no reliable way to find out who really published a webpage and for what purpose, there is no exact method to determine the webpages belonging to a certain site. Nonetheless, no-one would deny the existence of websites and thus, in order to retrieve relevant sites it is necessary to find a pragmatic definition that is suitable for the majority of cases. In this paper, we take advantage of the characteristic that a very large percentage of all websites is published under one dedicated domain or subdomain. For the cases that a website is spread over several domains/subdomains, we do not loose any results, but may have some duplicates in the result set. However, if large websites are classified more than once, their chance of being part of the result increases as well. The other problem of our definition is

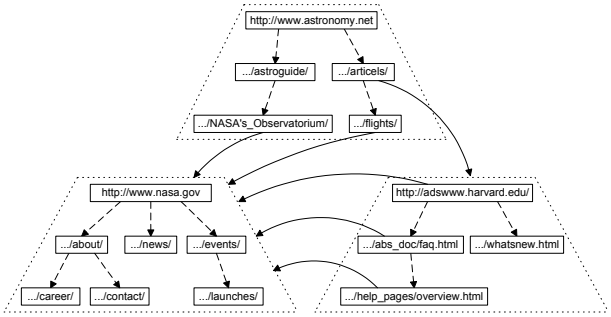

Figure 1: Sample portion of the website graph.

the case that one domain hosts several websites. Thus, websites without a domain of there own are not discovered. However, websites without a domain of there own are important in rare cases only and there is no search system on the WWW that can claim to achieve 100% recall.

Formally, for each page p, $host(p)$ returns the domain or subdomain of p, i.e. the substring of the URL of p between the protocol and the file section. We define a *website* W as a subgraph $W=(V',E')$ of the webpage graph with the following properties:

$$\forall\, u,v \in V' : host(u) = host(v)$$

$$\forall\, u \in V', v \notin V' : host(u) \neq host(v)$$

$$\forall\, e \in E' : source(e) \in V' \wedge destination(e) \in V' .$$

We define the homepage as the webpage that is referenced by the URL consisting of the domain name only (e.g. http://www.cs.sfu.ca). Thus, each website has a unique homepage that can be accessed knowing the website name only. Compared to the webpage graph, the website graph (which is the conceptual view of our website crawler onto the WWW), has several important differences: We distinguish two different types of nodes at different levels of abstraction, page nodes and site nodes. We distinguish two different types of edges, representing inter-site links and intra-site links. Edges for intra-site links point to page nodes, but edges representing inter-site links point to site nodes. These differences are due to the fact that a focused website crawler searches for whole websites and visits one website after another starting the exploration of a new website from its home page. For a more formal definition, let $G= (V,E)$ be the webpage graph. We distinguish between *intrinsic links* (p,q) with $host(p) = host(q)$ and *transversal links* with $host(p) \neq host(q)$. Let U denote the union of V and let W be the set of all existing websites. We define the *website graph* as a directed graph $WG=(U,D)$ with the set of edges D given as follows:

$$\forall\, (p,q) \in E : host(p) = host(q) \Rightarrow (p,q) \in D$$

$$\forall\, (p,q) \in E : host(p) \neq host(q) \Rightarrow (p, host(q)) \in D$$

Figure 1 depicts a small sample portion of the website graph consisting of three websites. Intrinsic links are represented by dashed arrows, transversal links by solid arrows.

3.2. Retrieving Websites with Focused Crawling

A focused webpage crawler [3] takes a set of well-selected webpages exemplifying the user interest. Searching for further relevant webpages, the focused crawler starts from the given pages and recursively explores the linked webpages. The conceptual view of the WWW of a focused page crawler is the *webpage graph*. The crawl frontier consists of all hyperlinks (or the referenced webpages) from downloaded pages pointing to not yet visited pages. The performance of the crawler strongly depends on the crawling strategy, i.e. the way the frontier is ordered.

There are several ways of post-processing the results of focused *page* crawlers to adapt them for the task of retrieving relevant *websites*. The simplest way is to select all homepages of websites found within the relevant pages of a crawl and to conclude that all corresponding websites are relevant. However, the classification of websites based on the homepage alone is not as accurate as more sophisticated methods of website classification. A homepage might consist of structural information only, e.g. frame tags or provide not much text. Thus, homepages often do not contain a meaningful description of the purpose of a website. As a consequence this approach to extract relevant websites from the results of a focused webpage crawl suffers from inaccurate results. Furthermore, since the webpage crawler does not prefer homepages over other webpages, the rate of newly discovered websites tends to be rather low.

Another approach of post-processing is to group the resulting webpages by their website (i.e. domain) and apply a website classifier like [15] to each set of webpages. Though this approach promises better classification accuracy, it still has drawbacks. Since the set of webpages downloaded for each site is controlled by the page crawler which is not conscious of websites at all, this selection of pages might not be well suited for representing the website. Thus, the crawler does not guarantee that enough webpages per site are downloaded. In our experiments, it turned out that usually more than 50% of the websites that were classified as relevant by this method, were represented by one webpage only. On the other hand, the efficiency suffers from the effect that very relevant websites might be scanned completely due to the high relevance scores of most of their pages. In addition to the number, also the selection of webpages of a conventional focused crawler causes a problem. Since a focused crawler prefers relevant pages, a website might be represented by the pages closest to the relevant topic. But this selection is not a good representation for websites that are irrelevant. Thus, websites containing some pages with relevant information, belonging, however, to the other class are misclassified. For example, a university might be classified as relevant for skiing because there are some student pages referring to this topic. We argue that in order to achieve high classification accuracy and to control the number of pages to be downloaded, a focused website crawler requires an explicit concept of websites and corresponding crawl strategies.

3.3. Focused Website Crawling

Website crawling can be considered as the process of successively transforming a subgraph G_0 of the website graph WG with $V_0=\{W_1,\ldots,W_n\}$, $n \geq 1$, where W_i is a website, $1 \leq i \leq n$, into a sequence of subgraphs $G_1,\ldots,G_m$ such that in each step exactly one website node from WG is added to G_i to obtain G_{i+1}. V_0 is called the set of *start websites*. In the context of *focused* website crawling, we assume two classes of websites, a class of relevant sites (the target class) and a class of irrelevant sites (the "other"-class) with respect to some user interest. The set of start sites V_0 should (mainly) consist of relevant sites. To distinguish between relevant and irrelevant websites, some (automatic) classifier is required which predicts the class of a website (V',E') based on the feature vectors $f(p)$ of the pages $p \in V'$. The *website classifier* is a function that takes a website from W and a website class from the set of classes C and returns a numerical confidence value for this website w.r.t. the given class.

$$confidence: W \times C \rightarrow [0..1]$$

A website is called *relevant,* if its confidence for the target class C_{target} exceeds its confidence for the "other" class.

$$relevance: W \rightarrow \{true, false\}$$

The website classifier is trained using the start websites that can be provided either explicitly by the user (if available) or implicitly by selecting some subtrees (and the corresponding websites listed in these subtrees) of a directory service like [10,12,18].

Based on the website classifier and the notion of relevant websites, we introduce the following performance measure for focused website crawlers. The *pages per relevant site rate (pprs-rate)* of the website crawler after step s is defined by the ratio of the number of downloaded webpages to the number of relevant websites found, i.e

$$pprs(G_s) = \frac{\sum_{W \in (G_s \setminus G_{s-w})} |pages(W)|}{|\{ W \in (G_s \setminus G_{s-w}) \mid relevance(W) = true\}|}$$

where $pages(W) = \{p \in W \cap (G_s \setminus G_{s-w})\}$ denotes the set of pages in W that were visited so far and w is the beginning of the time interval that is observed. The pprs-rate thus measures the average effort to retrieve one additional *relevant* website. It depends on two factors: (1) the number of pages that have to be downloaded within a relevant website (to be controlled by an internal crawler) and (2) the number of pages downloaded from irrelevant websites that were examined before finding the relevant website (to be controlled by the external crawler).

The task of a *focused website crawler* is to find as many relevant sites as possible, while downloading as few webpages as possible. A website crawl terminates if the wanted number of relevant sites is found or the pprs-rate

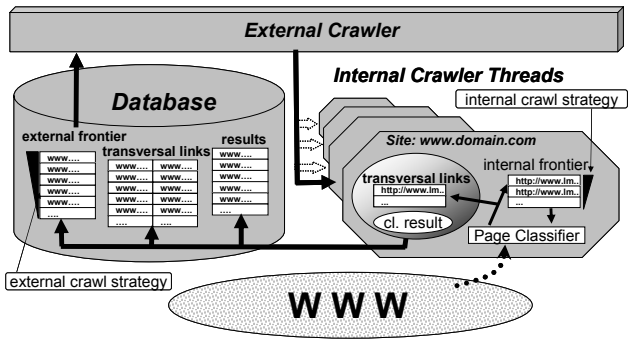

Figure 2: Architecture of the focused website crawler.

decreases significantly. In the next section, we will introduce our architecture of a focused website crawler.

4. A Focused Website Crawler

4.1. The Architecture

Focused website crawling is performed on two levels. The external or website level traverses the first level of the website graph. The external crawl orders the (hyperlinks to) yet unknown websites and invokes internal crawls on the top-ranked ones. Since there are much less domains than webpages, the external crawl frontier is rather small compared to the crawl frontier of an ordinary focused crawler. Thus, even for large crawls ranking can be done on-the-fly and sophisticated ranking algorithms can be applied. The second level is the internal or webpage level. It examines the current website to identify its purpose and extracts links to other websites while downloading as few webpages as possible. Since the webpages within the internal crawl frontier are needed for a limited time only and their number is usually small, it can be stored in main memory. Thus, expensive I/O operations are avoided and the crawl frontier can be accessed and updated very fast. Let us note that several internal crawlers examine different websites simultaneously. Thus, it is guaranteed that the data is drawn from several remote hosts at the same time which ensures a high overall download rate. Furthermore, controlling the number of pages visited from each website helps to keep the additional load at each website as low as possible, helping to increase the acceptance of the focused crawler within the webmaster community.

Figure 2 shows our architecture for a focused website crawler. The *external crawler* stores the external frontier consisting of websites only. To decide which website has to be examined next, it ranks the external frontier. To expand the frontier and to decide, if a chosen site is relevant, the external crawler invokes an internal crawler. The *internal crawler* traverses the website building an internal crawl frontier that is restricted to the pages of this site. During this traversal it examines the webpages to determine the site class. Furthermore, it collects all transversal links to other unexplored websites together with the confidence w.r.t. the target class of their source pages. As a result, the internal crawler returns information about the website class and the set of transversal links from the domain to new unexplored domains. Note that these transversal links are not real hyperlinks, but an aggregation of all hyperlinks that are found within the website directing to pages located within an other website. Thus, the number of transversal links from one site to another website is limited to one.

4.2. The External Crawler

The task of the external crawler is to order the external crawl frontier (consisting of links to not yet visited websites) and to decide which site has to be examined next by an internal crawler. The external crawler starts its traversal of the website level from the user-specified start websites and expands the graph by incorporating the newly found websites. Since the task of the external crawler is similar to the task of a focused crawler for webpages, most of the methods mentioned in section 2 are applicable to order the external frontier. The major difference is that distillation takes place at another more abstract level. Thus, the relevance scores attached to nodes and edges may be determined in a different way in order to achieve good results.

During a crawl, we distinguish two different sets of nodes of the website graph: Nodes corresponding to already examined websites are elements of V_{ex} and so-called border nodes that have not yet been examined, are elements of V_{bd}. The task of the crawler is to rank the elements of V_{bd} with respect to the information gained while examining the elements of V_{ex}. Each website $W \in V_{bd}$ is reachable by at least one link contained in some website $V_i \in V_{ex}$.

The (external) crawling strategy employed in this paper is simple but effective and is very similar to the basic crawler proposed in [5]. Note that most of the established crawling strategies [2,3,5,6,8,17] are applicable as well. For every node $W \in V_{bd}$, a *ranking score* is calculated as follows:

$$rank(W) = \frac{\sum_{V_i \in L_{ex}(W)} weight(V_i, W)}{|L_{ex}(W)|}$$

where $L_{ex}(W) = \{ V \mid V \in V_{ex} \wedge \exists \ edge(V,W) \}$ and $edge(V_i,W)$ denotes that there is at least one link from node V_i to node W. Furthermore, $weight(V,W)$ is a function that determines the confidence for each edge that its destination is relevant to the topic. In other words, an unknown website is judged by the average weight of the known edges referencing it. Thus, the website W with the highest $rank(W)$ should be crawled first. The edges do not directly correspond to the hyperlinks, but represent an aggregate of all hyperlinks leading from one website to another. Let us note that this method solves the same problem as the host-to-host cleaning improvement suggested in [1], i.e. it avoids that strongly connected

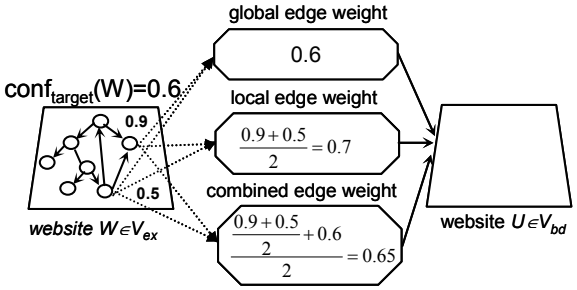

Figure 3: The three variants of edge weights for two sample websites W and U. The confidence of W w.r.t. the target class is 0.6. There are two pages in W referencing pages in U, one page with confidence (w.r.t. the target class) 0.9 and the other with confidence 0.5.

domains are overemphasized. The remaining task is how to determine the weights for the edges. To answer this question, we investigated the following three approaches :

- *Global edge weights:* Each edge is weighted by the confidence *w.r.t.* the target class of the *website* the edge is contained in:

$$weight_{global}(W,V) = confidence(W, C_{target})$$

where $confidence(W, C_{target})$ denotes the confidence value for website W *w.r.t.* the target class.

- *Local edge weights*: Each edge is weighted by the *average* confidence *w.r.t.* the target class of the *webpages* containing links pointing to the given website:

$$weight_{local}(W,V) = \frac{\sum \Pr[target \mid p]}{\left| \{p \mid p \in W \wedge \exists\ [q \in V \wedge (p,q)]\} \right|}$$

where $Pr[target|p]$ is the confidence of page p being contained in a target class website. These confidences for single webpages are also collected within the website classifier, but do not correspond to the complete set of webpages downloaded for classification.

- *Combined edge weights*: This is a combination of both methods integrating both scores to combine local and global aspects by taking the average weight of both methods.

$$weight_{combined}(W,V) = \frac{weight_{local}(W,V) + weight_{global}(W,V)}{2}$$

The advantage of local edge weights is that they distinguish the transversal links according to the relevance of the source pages of a link. Thus, transversal links found on irrelevant pages are weighted less than those found on highly relevant webpages. On the other hand, local edge weights might consider the links from source pages containing sparse text only as irrelevant since the page itself can be classified only poorly. This shows the strength of global edge weights. Since global edge weights consider the relevance of the complete site, they transfer relevance from other relevant pages to the link pages which do not provide enough content for proper classification. Combined edge weights incorporate both aspects. The links found in pages containing not enough text for reliable classification are at least judged by the relevance of the website and relevant pages transfer more importance to the links than irrelevant ones. Figure 3 displays an example for all three methods of edge weighting.

The performance of the external crawler influences one important aspect of the pprs-rate: the number of relevant sites that are examined compared to all websites that are crawled by an internal crawler. We will refer to this ratio as the *website harvest rate*. However, this aspect is not the only influence on the pprs-rate. Even an optimal external crawler will achieve very bad pprs-rates, if the internal crawler explores large numbers of webpages per site.

4.3. The Internal Crawler

The internal crawler is responsible for the main advantage of a dedicated website crawler namely that the results are more reliable due to better classification accuracy. On the other hand, the efficiency strongly depends on the ability of the internal crawler to restrict the number of downloaded webpages per site to as few pages as possible. Furthermore, additional goals have to be fulfilled like the avoidance of spider traps and the retrieval of new promising transversal links.

The main task of the internal crawler is to select a representative sample set of webpages from a website W and determine for each page p_i the likelihood (called *confidence* in this context) of p_i appearing in website class C_k. To determine this probability $Pr[w_i | w_i \in W \wedge W \in C_k]$, we employ a text classifier. To choose the sample set, we employ focused crawling using a so-called internal crawl strategy. To determine the class of an entire website W, for each class C_k we calculate the probability that W was generated by the process corresponding to class C_k.

Additionally, there are several other side goals of the internal crawler like collecting new transversal links and avoiding spider traps.

4.3.1 The Webpage Classifier

The task of the webpage classifier is to decide how likely it is that a certain webpage p_i appears in a website W of Class C_k. The task of this classifier is slightly different from the task of the classifier in an ordinary focused crawler. A webpage that is likely to appear in a typical website does not necessarily have to be relevant for the user interest. The page classifier should be capable to handle multi-modal classes, i.e. classes that are strongly fractioned into an unknown number of subclasses. This feature is important because the webpages found in websites of a common class provide several pageclasses, e.g. contact-pages, directory pages etc.**.** For our crawler, we employed a centroid based *k*-nearest neighbor (kNN)

classifier as described in [13]. This variant of kNN classification constructs the centroid of the training word vectors for each class. The class is now determined by choosing that class belonging to the closest centroid. In order to achieve multi-modality, we adopted an idea mentioned in the summary of [13]. We clustered each training set using the *k*-means algorithm and represented a class as the set of centroids of the resulting clusters. Let us note that we started our prototype by using naive Bayes classification, but changed to this classifier due to its better accuracy.

Formally, each class C_k of our classifier is represented by a set of centroids CS_k. Let $d_{min}(p, CS_k)$ denote the distance of the word vector p of a given webpage to the closest element of CS_k. Then we estimate the confidence value for p belonging to C_k as follows:

$$\Pr[p \mid C_k] = \frac{\ln[d_{\min}(p, c_k)]}{\sum_{c_j \in CS_k} \ln[d_{\min}(p, c_j)]}.$$

In other words, we use the logarithm of the distance to the closest centroid in CS_k and normalize over all classes. Let us note that we use the logarithm to weight close distances higher than far distances. If a page has a large distance to the centroids of all classes, the confidence values are very similar for all classes. The closer the distance to a centroid is, the more sensitively the distance is measured. The resulting confidences are used by the local and combined edge weights for determining the weights of the transversal links.

To train the classifier, we first select a set of relevant websites. The websites in our experiments, for example, were taken from common directory services [10,12,18]. To represent the "other"-class, we choose several websites belonging to a variety of other non-relevant topics. Since we need to learn which kinds of webpages might occur in a relevant site and which not, we have to draw a representative sample of webpages from each training website. The pages downloaded during the process of classification of a website are limited to a small set around the homepage, since these pages are most likely connected to the purpose of the site. Thus, we should use these pages for training as well. We restrict the training pages to the first *k* pages when traversing the website using breadth first search. This simple method worked out well in our experiments.

4.3.2 The Internal Crawl Strategy

The internal crawl strategy determines the sample of pages downloaded from the website to be classified. Each internal crawl is started at the homepage. As mentioned before, the information about the purpose of a website is usually located around the homepage since most publishers want to tell the user what a website is about, before providing more specific information.

Analogously to a focused page crawler, the internal crawler traverses the web using a best-first search strategy. However, the internal crawl is restricted to the webpages of the examined site. The goal is to find a set of webpages reflecting the site's purpose in a best possible way. This is a major difference to focused page crawlers which try to find as many relevant pages as possible. However, looking for relevant pages is only appropriate for site classification, if the examined website belongs to the target class. If the given website belongs to the other class, the crawler should prefer pages that typically occur in non-relevant websites in order to find a good representation. Thus, the internal crawler should rank the pages by their confidences for any class compared to the average confidences over all classes.

To solve this problem, our internal crawling strategy works as follows. Like in the external crawler, we again build the crawling strategy similar to the basic crawler in [5]. The ranking score of a webpage p is defined as the average weight of the links referencing p:

$$rank(p) = \frac{\sum_{q_i \in L_{in}(p)} weight((q_i, p))}{|L_{in}(p)|}$$

where L_{in} is the set of pages read so far that link to p. To represent the contribution of a page for the decision in favor of either class, we determine the weight of link $(q_i p)$ as:

$$weight(q_i, p) = \mathrm{var}(\Pr[q_i \mid target])$$

where $Pr[q_i|target]$ is the confidence of q_i w.r.t. the target class obtained by the page classifier. The internal frontier is sorted in decreasing order of these confidence values.

4.3.3. The Website Classifier

The combination of the page classifier and the internal crawl strategy produces a sequence of webpages downloaded from the site. Furthermore, each webpage is classified and is associated with a confidence w.r.t. the target class.

The following statistical model incrementally (i.e. after each download of a new page) aggregates these page confidences to calculate an overall confidence (w.r.t. the target class) for the entire website. In our model, each website class defines a statistical process that can generate any webpage with a certain probability. A website W belonging to that class C_k is a set of webpages generated by drawing pages from the corresponding probability distribution. In the following, we present a maximum-likelihood classifier that assigns a website to the class with the highest probability of having generated the observed website W.

Let W_t denote the sample of site W that the internal crawler has retrieved by time t. The probability that the class C_k has generated W_t is given by

$$\Pr[W_t \mid C_k] = \prod_{p_i \in W} \Pr[p_i \mid C_k].$$

Applying Bayes theorem, the desired probability is:

$$\Pr[C_k \mid W_t] = \frac{\Pr[C_k] \cdot P[W_t \mid C_k]}{\sum_{i \in C} \Pr[C_i] \cdot P[W_t \mid C_i]}.$$

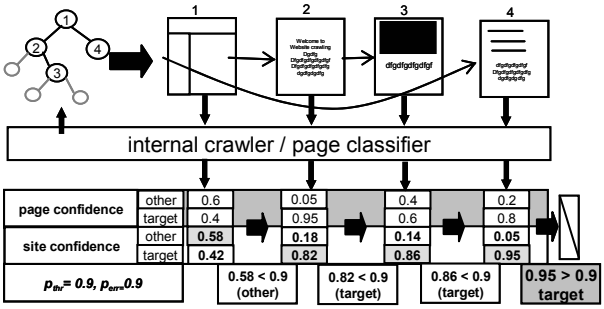

Figure 4: Illustration of website classification during an internal crawl

Unfortunately, this formalization suffers from two practical limitations:

- The apriori probabilities $Pr[C_k]$ are unknown for the WWW. However, the application of focused crawling enables us to make a suitable estimate. Since the focused website crawler focuses to relevant sites, the probability distribution within the whole web is expected to be very different from the probability distribution within relevant sites close to the frontier. Thus, we can use the rate of relevant sites found so far as an estimate for $Pr[C_k]$.
- Since there is no classifier guaranteeing 100 % accurate class predictions, the confidence values are not always realistic as well. The class prediction values generated by the classifier always suffer from a certain classification error. Thus, the combination of these results should consider this inaccuracy.

To incorporate the possibility of classification errors, we extend our model by integrating the classification error observed on the training data into the model. Thus, we obtain an error corrected probability for the occurrence of page p in a website of class C_k and the classification error p_{err}:

$$\Pr[p \mid C_k \wedge p_{err}] = \Pr[(p \mid C_k] \cdot (1 - p_{err}) + \Pr[p \mid C_{other}] \cdot p_{err}$$

The idea is that the probability that the prediction is made correctly is the confidence value multiplied with the probability that the classifier is correct. Additionally, we have to consider the case that the classifier made a wrong prediction. Thus, we have to add the confidence value of the "other"-class multiplied with the error probability perr. To estimate perr, we calculate the accuracy of the page classifier on the set of webpages in the training websites using 10-fold cross validation.

Using the error corrected probabilities avoids the effect that the influence of a single page is overestimated during classification. Even if the classifier outputs are 1.0 and 0.0, our process does not automatically overestimate the impact of a single page. Thus, the calculated value for $Pr[W_t \mid C_k]$ will usually produce meaningful values after some pages have been considered.

To stop classification, we define a certain confidence threshold $p_{threshold}$ and the internal crawl stops classification as soon as this confidence level is reached. By choosing $p_{threshold}$ the internal classifier can be adjusted to find an appropriate trade-off between accuracy and efficiency. However, if its value is chosen too high, the crawler will require too many pages with respect to a website's purpose. This is problematic, because the performance suffers significantly and the reservoir of characteristic pages within one website is limited. To conclude, after the confidence for W_t reaches $p_{threshold}$, we assume that the class of W is identical to the class of W_t and we denote:

$$confidence(W, C_{target}) = \Pr[C_{target} \mid W_t] \quad \text{and}$$

$$relevance(W) = \left(\Pr[C_{target} \mid W_t] > \Pr[C_{other} \mid W_t]\right).$$

Figure 4 illustrates the complete process of website classification. The displayed example describes the common case that a website starts with a frame page and, thus, the prediction of the class based only on the homepage would be wrong.

4.3.4. Retrieving Transversal Links and Terminating the Internal Crawler

Besides the primary goal to achieve accurate classification of the examined website, the internal crawler has a secondary objective of retrieving enough transversal links for extending the external crawl frontier. Therefore, the internal crawler collects all transversal links, i.e. the links leading to new unexplored websites. Additionally, the crawler stores the confidence values $Pr[p|C_{target}]$ of the source pages of the link. These values are used to calculate local and combined edge weights.

Since, according to the above stop condition, classification might be finished after a few pages only, it is possible that the internal crawler has not yet found enough interesting transversal links. In such cases we want to continue the crawl until a reasonable number of transversal links has been extracted. To decide if enough links have been found within a website, we define the *linkWeight* as a measure for the contribution of page p to the set of relevant transversal links found within the site:

$$linkWeight(p) = \left(\Pr[p \mid C_{target}] \cdot |LT_p| + c\right)$$

where LT_p is the set of transversal links found in p and $c \geq 1$ is an constant. Furthermore, we define the *LinkRank* for the set of webpages W_t as :

$$LinkRank(W_t) = \sum_{p \in W_t} linkWeight(p) \cdot \frac{1}{\Pr[C_{traget} \mid W_t]}.$$

To employ the *LinkRank* for ensuring that enough relevant links are found, we continue the internal crawl even after classification has finished until it reaches a certain level $l_{threshold}$. The idea of this heuristic is that each webpage contributes its *linkWeight* to the *LinkRank* of the website. The more links are contained in p and the more relevant p is, the more will p contribute to the *LinkRank*. The constant c is added to ensure that the *linkWeight* has

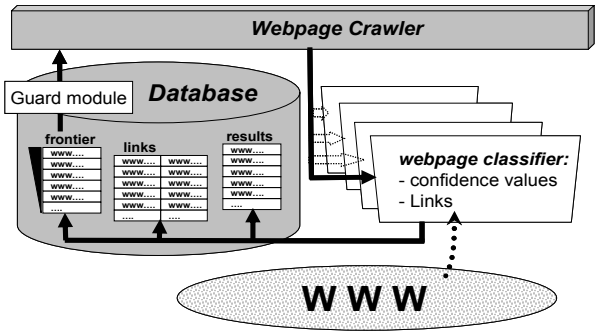

Figure 5: Architecture of our focused webpage crawler.

at least some value and thus the *LinkRank* grows constantly until $l_{threshold}$ is reached. The *LinkRank* increases slower for relevant websites and faster for irrelevant ones. Thus, an internal crawl of a relevant website will encompass more webpages than an internal crawl of an irrelevant site, which usually terminates after classification. This way relevant websites add more new links to the external frontier than irrelevant ones.

Let us note that we continue the crawl to reach $l_{threshold}$ by employing the mentioned internal crawling strategy. We argue that if a website is relevant, the crawling strategy is targeted to find new relevant pages which are most likely to contain relevant links. For websites classified to the other class, $l_{threshold}$ is reached rather fast anyway and switching the crawl strategy is not necessary.

An additional benefit of the internal crawler is that it makes the website crawler robust against spider traps. Since the number of webpages retrieved from one website is explicitly controlled, the crawler might run into a spider trap only in those rare cases where a site consists mostly of pages without any meaning to the classifier. To ensure termination in such cases, it is sufficient to restrict the number of pages downloaded from one domain. Unlike in page crawlers, no additional database table is needed to store websites containing a spider trap. This is not necessary within the focused website crawler since the crawler will not visit a website more than once.

5. EXPERIMENTAL EVALUATION

5.1. The Test Environment

We performed our experiments for the topics listed in table 1. For each topic, we first acquired a sample set of relevant websites taken from a category in [10,12,18]. Additionally, we selected a random mixture of websites to represent all other topics on the web. For each category, table 1 provides the number of training websites, and the directory service the websites were taken from. We stored the websites in a training database, to have a stable test environment consisting of 20,793 HTML-documents from 335 websites.

We implemented 2 focused crawlers. The first is our prototype of a focused website crawler. The second is a

Table 1: Overview of the training database.

topic	number of websites in db	websites provided by
horses	32	YAHOO
astronomy	39	YAHOO
sailing	39	Google
mountain biking	34	DMOZ
skate boarding	35	DMOZ
boxing	33	DMOZ
other	132	All

focused webpage crawler that crawls the internet by using only one frontier of webpages. To provide a fair comparison, both crawlers are based on the same algorithm for page classification and ranking. The design of our focused webpage crawler is illustrated in Figure 5. The system starts its crawl on a defined set of webpages (in our case the homepages of the websites found in a directory service). Each new unexplored webpage is stored in the crawling frontier. The page classifier generates confidence values for each webpage that is explored. Within the frontier, each unexplored webpage is measured by the average confidence value for the target class of the webpages linking it. The webpage providing the highest average confidence value within the frontier is examined next. In order to prevent spider traps and to keep the load for each website at an acceptable level, we implemented a guard module as described in [6]. This guard module prevents the page crawler from accessing webpages in websites that already contributed an extraordinarily high number of webpages to the already explored part of the web graph. To test the crawlers, we performed various crawls on the WWW. This test bed seemed to be suited best, although it is not guaranteed that the web stays the same between two crawls. However, due to the more stable character of the website graph, we argue that the influence to the results is negligible. Let us note that we performed some of the experiments again after several weeks and achieved almost identical results. On the other hand, downloading a representative section of the website graph to provide a static test environment is difficult. Since the part of the WWW visited by a website crawler tends to be spread over several thousands hosts, it is difficult to find a closed section that allows a realistic behaviour of the tested crawlers.

Our experiments were run on a workstation that is equipped with two 2.8 Ghz Xeon processors and 4 Gb main memory. As a database system we used an ORACLE 9i database server hosted on the same machine. Both crawlers were implemented in Java 1.4, with the exception of the ranking algorithms and the guard module which were partly implemented in PL/SQL to improve the runtime performance.

Table 2: Classification results using 10-fold cross validation within the training database for the internal crawler and the homepage classifier.

topic	p_{err}	$P_{threshold}$	pages per site	prec. int crw.	rec. int crw	f-meas. int. crw.	prec homep.	rec. homep.	f-measure homepage	f-measure improve.
horses	0.85	0.9	6.9	0.84	0.97	**0.90**	0.49	0.88	**0.63**	0.27
astronomy	0.90	0.9	6.3	1.00	0.90	**0.95**	0.86	0.79	**0.83**	0.12
sailing	0.88	0.9	6.3	0.90	0.97	**0.94**	0.77	0.92	**0.84**	0.10
mountain biking	0.86	0.8	6.3	0.81	0.97	**0.88**	0.76	0.83	**0.79**	0.11
skate boarding	0.88	0.8	3.2	0.76	1.00	**0.86**	0.74	0.89	**0.81**	0.05
boxing	0.88	0.9	7.4	0.79	0.79	**0.79**	0.74	0.72	**0.73**	0.05

5.2. Evaluation of Website Classification

Our first experiment demonstrates the higher accuracy that can be achieved for website classification by using the internal crawler compared to a homepage classifier.
The homepage classifier uses the same centroid based *k*NN-classifier as the internal crawler, but is trained and tested on homepages only. The internal crawler used in these experiments terminates its crawl after a confidence threshold of $p_{threshold}$ is reached and does not continue the crawl to find interesting links. Since this test needs labelled test data, we performed 10-fold cross-validation on the topics stored in the training database (table 1). Table 2 displays the precision, recall and f-measure (as trade off between precision and recall) for the tested topics when employing the website classifier and the homepage classifier. Additionally, the table reports the classification error p_{err} and the average number of webpages that the website classifier downloaded per website. For the training of the page classifier of the internal crawler, we used the first 25 webpages of each training website when applying a breadth-first traversal.
For all of the tested topics, the internal crawler obtained significantly higher f-measures than the homepage classifier. For the topic horses, it even increased the f-measure from 0.63 to 0.9, i.e. by 0.27. Thus, by classifying the websites by more than one page, the classification accuracy was substantially increased. Let us note that a manual analysis of the crawled websites confirmed the hypothesis that especially commercial websites often do not provide a meaningful homepage. The average number of pages used for classification was between 3.2 and 7.4 indicating that website classification does not require large numbers of webpages per site for making more accurate predictions.

5.3 Evaluation of the Crawling Performance

To demonstrate the performance of the complete focused website crawler, we performed numerous crawls. Since we retrieved a total number of approximately 50,000 potentially relevant websites, we could manually verify only samples from each crawl. Table 3 displays a sample of relevant websites retrieved for the topic horses. The first five domains were retrieved after approximately 250 websites were visited, the last five at the end of the crawl after about 2500 relevant websites had been retrieved. This example illustrates that the crawler started to discover relevant websites early and kept his good accuracy until the end of the crawl.

Table 3: Example websites returned for the topic horses.

website	# visited pages	confi-dence
www.tbart.net	4	0.65
www.socalequine.com	6	0.59
www.thehalterhorse.com	4	0.65
www.thejudgeschoice.com	4	0.75
www.thehorsesource.com	3	0.68
…		
www.laceysarabians.com	5	0.71
www.baroquehorses.com	5	0.60
www.knightmagicfarms.com	4	0.67
www.pccha.com	7	0.64
www.danddhorsetransport.com	4	0.70

Our first crawling experiment compares the three different weightings introduced in section 4.2 for ranking the external frontier. Therefore, we started each crawler using the parameters achieving maximum accuracy for the internal crawler and stopped the crawler after approximately 2500 relevant websites were found. To compare the effect of each of the weightings, we compared the website harvest rate, i.e. the ratio of relevant websites to all websites that were screened. Figure 6 displays the average website harvest rate aggregated over the last 1000 pages. For the topics horses and astronomy all three weightings performed very similar, although the global edge weights achieved a small advantage, especially at the beginning of the crawl. However, for the topic sailing the combined edge weights were able to compensate some of the weaknesses of both underlying methods. The experiments for the topic mountain biking displayed a strong advantage for the local edge weights. However, the combined edge weights were still able to compensate some of the weaknesses of global edge weights. Though our experiments did not reveal that one of the mentioned weightings showed superior results, we advise to employ the combined edge

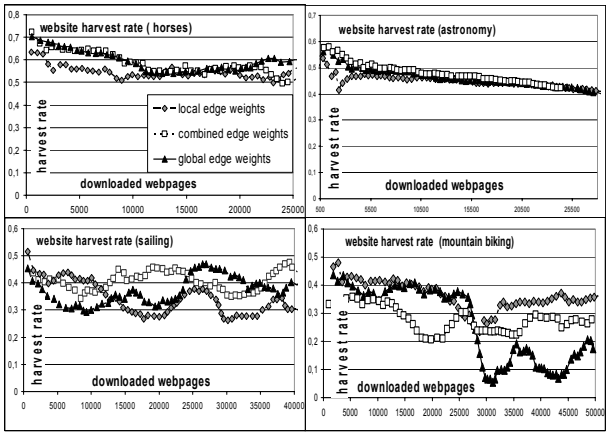

Figure 6: Website harvest rates (average of last 1000 pages) for each topic and each weighting.

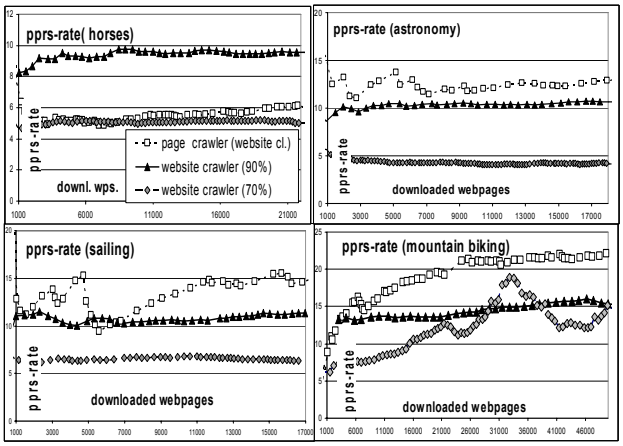

Figure 7: pprs-rates (average of last 5000 pages) for each topic and each crawler.

weights function, since it was always at least the second best and sometimes outperformed the other methods.

The next series of experiments was conducted to back up our claim that common focused (webpage) crawlers are unsuitable for retrieving websites and that the proposed focused website crawler overcomes the problems of page crawlers providing a more efficient and accurate retrieval of relevant websites.

In our first experiment, we have already demonstrated that the accuracy of the internal crawler is superior to the accuracy achieved by the homepage classifier. Thus, the post-processing counting relevant homepages is unlikely to produce the same quality of results either. To show that applying a website classifier like [15] is not sufficient for providing comparable accuracy, we determined the percentage of websites that were classified by one single webpage. For all four examples approximately 50 % of the resulting websites where classified by using one page only. Thus, in half of the cases applying a more sophisticated website classifier to the websites being aggregated from the results of a page crawl cannot perform any better than the homepage classifier.

This behaviour of the page crawler can be explained as follows. Most transversal links referencing a new site are directed at one special entry page (usually the homepage) and most other webpages found within this website are linked via internal links only. A page crawler examining a website visits this entry page first and classifies it. The ranking score of the other webpages within the website now strongly depend on the confidence value of the entry page. If the confidence w.r.t. the target class is rather high, then additional pages are examined also. If the classification result is rather uncertain, however, the ranking scores tends to be rather low and it is likely that the additional pages won't be visited during the crawl. For the task of website retrieval this behaviour is unsuitable. If the relevance of the entry page is hard to decide, it would make sense to examine additional pages from the site in order to achieve more reliable classification. On the other hand, if the relevance of the entry page is very certain, it is wasteful to proceed crawling to discover the obvious. Our proposed website crawler handles candidate sites that cannot reliably be classified based on the entry page more carefully than those where a certain classification can immediately be obtained.

To demonstrate this difference, we ran the focused webpage crawler for each of the first 4 topics listed in table 1 and applied a website classifier to the results. Additionally, we performed two different website crawls to demonstrate the capability of the website crawler to find a suitable trade-off between accuracy and efficiency by adjusting the confidence threshold. The first one uses again the parameter setting providing maximum accuracy ($p_{threshold} \approx 90\%$). Thus, we can judge the overhead for the additional accuracy. The second crawl used a confidence value of 70%. Due to this rather soft breaking condition, the second crawl usually visited very few pages per website, but provided less reliable results.

Figure 7 displays the average pprs-rate over the last 5000 webpages for the first four topics displayed in table 1. Recall that the pprs-rate measures the average number of additional webpages that are downloaded until a new relevant website is discovered. Let us note that the crawls vary in length, since we terminated crawling after reaching at least 2500 relevant websites regardless of how many webpages where downloaded.

For three out of four topics, even the website crawler aiming at more accurate results ($p_{threshold} \approx 90\%$) achieved a lower pprs-rate than the page crawler. For the topic mountain biking, e.g., it needed approximately 7 pages less than the page crawler to find an additional relevant domain at the end of the crawl. Thus, even when returning more reliable results, the website crawler in most cases gained an efficiency advantage compared to the page crawler.

For all topics, the website crawler with a 70 % confidence threshold clearly outperformed the two comparison

partners with respect to efficiency. For the topic astronomy, e.g., it visited only about five additional webpages until it retrieved another relevant site. Due to the large number of results, we could not verify the entire result set, but a manual analysis of a sample supported our claim of more reliable results even for the 70% website crawler. To conclude, our experimental evaluation demonstrates that a focused website crawler is, for similar accuracy requirements, clearly more efficient for retrieving relevant websites than a focused webpage crawler with website post-processing. In an alternative scenario, when achieving a comparable pprs-rate, the focused website crawler returns more accurate results.

6. CONCLUSIONS

When searching the web, there are many applications targeting whole websites rather than single webpages. For that purpose, we introduced a focused crawler directly searching for relevant websites instead of webpages. The proposed two-level architecture allows us to control the number of pages to be downloaded from each website and to find a good trade-off between accurate classification and efficient crawling. The external crawler views the web as a graph of linked websites, selects the websites to be examined next and invokes internal crawlers. An internal crawler views the webpages of a single given website and performs focused page crawling within that website. In our experimental evaluation we demonstrated that reliable website classification requires to visit more than one but less than all pages of a given site. Furthermore, we compared our proposed crawler to a focused webpage crawler that handles the concept of websites in a corresponding step of post-processing. For the same efficiency (measured by the number of pages downloaded per relevant site), the website crawler achieved significantly higher classification accuracy than its comparison partner. For comparable accuracy, the website crawler needed a considerably smaller rate of pages visited per relevant site. These results support our claim that in order to achieve high classification accuracy and efficiency of crawling, a focused website crawler requires a two-level architecture and corresponding crawl strategies with an explicit concept of websites.

For future work, we plan to develop more specific internal and external crawling strategies. Furthermore, we will investigate the crawling of commercial websites that are not strongly linked to each other due to their competitive nature. In these cases, spotting hub pages seems to be more important than for crawling broad topics on the web. Another highly interesting direction is the use of website crawling to provide a filter for crawling highly specific content from the web. The idea is to first spot websites that are likely to contain the specific information. Afterwards these websites can be scanned for a webpage containing the specific information.

REFERENCES

[1] Bharat K.,Henziger M.R.: ”Improved Algorithms for Topic Distillation in a Hyperlinked Environment”, Proceedings of SIGIR-98, 1998.

[2] Chakrabarti S., van den Berg M., Dom B.: “Distributed Hypertext Resource Discovery Through Examples”, Proceedings VLDB 99.

[3] Chakrabarti S., van den Berg M., Dom B.: “Focused Crawling: a new Approach to Topic-Specific Web Resource Discovery”, Proceedings WWW 1999.

[4] Chakrabarti S., Dom B. and Indyk P.: ”Enhanced hypertext categorization using hyperlinks”, Proceedings ACM SIGMOD, 1998.

[5] Chakrabarti S., Punera K., Subramanyam M.: “Accelerated Focused Crawling through Online Relevance Feedback”, Proceedings WWW 2002.

[6] Chakrabarti S. :” Mining the Web”, (pp. 17-43), Morgan Kaufmann Publishers, 2003.

[7] Craven M., DiPasquo D., Freitag D., McCallum A., Mitchell T., Nigam K., and Slattery S.: ”Learning to Construct Knowledge Bases from the World Wide Web”, Artificial Intelligence, Elsevier, 1999.

[8] Cho J., Garcia-Molina H., Page L.: “Efficient Crawling Through URL Ordering”, Proceedings WWW 1998.

[9] Diligenti M., Coetzee F., Lawrence S., Giles C.L., Gori M.: “Focused Crawling Using Context Graphs”, Proceedings VLDB 2000.

[10] DMOZ open directory project, http://dmoz.org

[11] Ester M., Kriegel H.-P., Schubert M.: “Website Mining : A new way to spot Competitors, Customers and Suppliers in the World Wide Web”, Proceedings ACM SIGKDD 02.

[12] Google Search Engine, http://www.google.com

[13] E.-H. Han and G. Karypis. ”Centroid-Based Document Classification: Analysis and Experimental Results”. In Proc. PKDD’00, 2000.

[14] Joachims T.: “Text Categorization with Support Vector Machines: Learning with Many Relevant Features”, Proceedings European Conference on Machine Learning, 1998.

[15] Kriegel H.-P., Schubert M.: „Classification of websites as sets of feature vectors“, Proc. IASTED DBA 2004.

[16] Kleinberg J.: “Authoritative sources in a hyperlinked environment“, Proc. ACM-SIAM, 1998.

[17] Rennie J., McCallum A.: “Using Reinforcement Learning to Spider the Web Efficiently”, Proceedings ICML 99.

[18] Yahoo! Directory Service, http://www.yahoo.com

[19] Yang Y., Liu X.: “A Re-Examination of Text Categorization Methods”, Proceedings ACM SIGIR 1999.