7

Adaptive Focused Crawling

Alessandro Micarelli and Fabio Gasparetti

Department of Computer Science and Automation
Artificial Intelligence Laboratory
Roma Tre University
Via della Vasca Navale, 79 00146 Rome, Italy
{micarel, gaspare}@dia.uniroma3.it

Abstract. The large amount of available information on the Web makes it hard for users to locate resources about particular topics of interest. Traditional search tools, e.g., search engines, do not always successfully cope with this problem, that is, helping users to seek the right information. In the personalized search domain, focused crawlers are receiving increasing attention, as a well-founded alternative to search the Web. Unlike a standard crawler, which traverses the Web downloading all the documents it comes across, a focused crawler is developed to retrieve documents related to a given topic of interest, reducing the network and computational resources. This chapter presents an overview of the focused crawling domain and, in particular, of the approaches that include a sort of adaptivity. That feature makes it possible to change the system behavior according to the particular environment and its relationships with the given input parameters during the search.

7.1 Introduction

Traditional search engines allow users to submit a query suggesting, as output, an ordered list of pages ranked according to a particular matching algorithm. The underlying Information Retrieval model's goal is to allow users to find those documents that will best help them meet information needs and make it easier to accomplish their information-seeking activities. The query can be considered the user's textual description of the particular information request. If the engine works in the Web domain, a software system usually named *crawler* traverses it, collecting HTML pages or other kinds of resources [69, 83]. It exploits the hyperlink structure in order to find all the destination anchors (or targets) reachable from a given starting set of pages through the outgoing links. The crawler also keeps the snapshot and the internal index of the search engines up-to-date, periodically recrawling and updating the pages with fresh images.

General-purpose search engines employ crawlers to collect pages covering different topics. At query time, the engine retrieves subsets of pages that are more likely to be related to the user current needs expressed by means of sets of key-

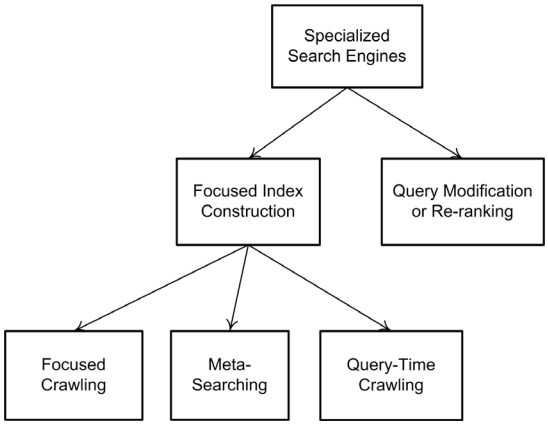

Fig. 7.1. A taxonomy of approaches to build specialized search engines, as shown in [80].

words. Models of the user needs are usually not employed, therefore the search results are not personalized for the user. Basically, two users, with different interests, knowledge and preferences, obtain the same results after having submitted the same query.

A first step toward a better search tool is developing *specialized search engines*, which provide tailored information on particular topics or categories for focused groups of people or individual users. The heavy constraints of a general-purpose search engine, i.e., indexing billions of pages and processing thousands of queries per second, are no longer required for these kinds of tools. New techniques can be included to represent Web pages and to match these representations against the interests of users, e.g., algorithms based on Natural Language Processing (NLP), usually avoided due to the computational resources needed.

There are several approaches to build specialized search engines [80], as shown in Fig. 7.1. Query modification and re-ranking exploit traditional search tools, filtering their content by augmenting user queries with keywords, or re-ordering the results, removing irrelevant resources. These techniques are widely discussed in Chapter 6 of this book [61]. Specialized search engines are also based on focused indexes, which contain only the documents related to the given topics of interest. To retrieve and index those documents, it is possible to meta-search specialized databases, or perform an autonomous search at query time. The most interesting technique is to perform focused crawling on the Web. It concerns the development of particular crawlers able to seek out and collect subsets of Web pages that satisfy some specific requirements. In particular, if the goal is to collect pages related to a given topic chosen by the user, the crawlers are usually named *focused* or *topical* [20, 17] (see Fig. 7.2). Focused crawlers are also employed in different domains from specialized IR-based search engines, but usually related to the retrieval and monitoring of useful hypertextual information, as shown later on in this chapter.

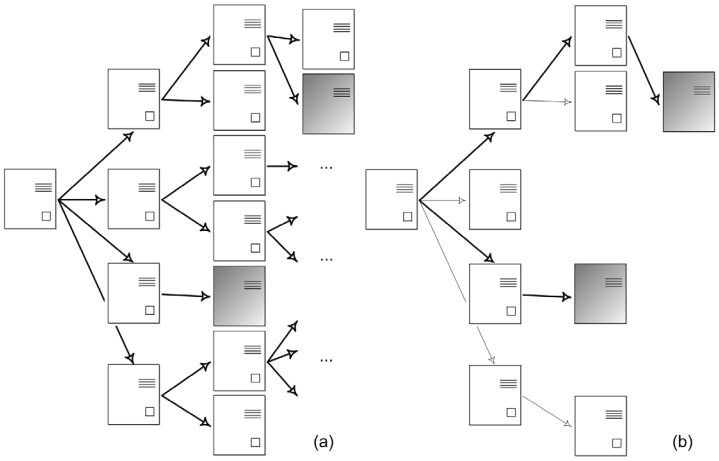

Fig. 7.2. Focused crawling attempts to find out and download only pages related to given topics (b), while standard crawlers employs traditional strategies (a), e.g., breadth-first search or further techniques to balance the network traffic on Web servers.

The focused crawling approach entails several advantages in comparison with the other approaches employed in specialized search engines. Performing an autonomous search at query time considerably delays the retrieval of result lists. Meta-searching provides results from existing general-purpose indexes that often contain outdated versions of the available documents. Due to the reduced storage requirements, focused crawling can employ techniques to crawl part of the deep Web (dynamically generated Web pages not accessible by search engines, see Sect. 7.2) or to keep the stored information fresh updating it frequently.

User queries, which usually consist of 2-3 keywords for traditional search engines [51, 77, 7], can be enriched by further information, such as subsets of taxonomy classes judged interesting by users, e.g. categories in the `Open Directory Project`[1] (ODP) or `Yahoo! Directory`[2], or by means of relevance feedback [75], see also Chapter 2 [42] and 6 [61] of this book for details. Better representations of the user needs help the focused crawlers to find out interesting pages, obtaining more accurate results.

If a focused crawler includes learning methods to adapt its behavior to the particular environment and its relationships with the given input parameters, e.g. the set of retrieved pages and the user-defined topic, the crawler is named *adaptive*. Non-adaptive crawlers are usually based on classifiers whose learning phase ends before the searching process starts. Even though some of them employ hypertextual algorithms, such as HITS - which lead to better results, making more information available - the adaptability is actually not manifest. Sometimes adaptive crawlers provide a sort of feedback mechanism where useful information is extracted from the analyzed documents, and the internal classifier updated

[1] www.dmoz.org
[2] www.yahoo.com

consequently [18]. Other approaches can explicitly model the set of pages around the topical ones. Such models capture important features that appear in valuable pages, and describe the content of the pages that are frequently associated with relevant ones. The searching process may benefit from those models, obtaining better overall crawl performance [72, 32, 1].

Adaptive focused crawlers are key elements in personalizing the human-computer interaction. Traditional non-adaptive focused crawlers are suitable for communities of users with shared interests and goals that do not change with time. In this case, it is easy to recognize the requested topics and start a focused crawl to retrieve resources from the Web. The adaptive crawler's advantage is the ability to learn and be responsive to potential alterations of the representations of user needs. This could happen when users do not know exactly what they are looking for, or if they decide to refine the query during the execution if the results are not deemed interesting. Therefore, adaptive focused crawlers are more suitable for personalized search systems that include a better model of the information needs, which keeps track of user's interests, goals, preferences, etc.. As a consequence, adaptive crawlers are usually trained for single users and not for communities of people.

A further advantage of adaptive crawlers is the sensitivity to potential alterations in the environment. Web pages are constantly being updated, as well as the related hyperlink structure. Therefore, a focused crawler should be able to monitor any change in order to look for new and interesting information.

The related domains that can really benefit from the focused crawling are the so-called vertical portals and studies on Web evolution. The former domain is related to Web sites that provide information on a relatively narrow range of goods and services for communities of users. Retrieving valuable, reliable and up-to-date information resources for the user is a typical purpose of these portals, as well as offering personalized electronic newspapers [10], personal shopping agents [34] or conference monitoring services [48]. Research activities on the study of the evolution of Web hypertextual environment [54, 66] usually requires a constant monitoring of specific portions of Web and of related changes during a given time interval, e.g., bursty communities of blogs [49]. Both domains could really benefit from employing focused crawling systems to retrieve information that meets the given input constraints, discovering properties that combine the topical content of pages and the linkage relationship between them.

The remainder of the chapter is structured as follows: research on the WWW and issues related to crawlers' development are discussed in Sect. 7.2. Section 7.3 includes references to focused crawlers where the adaptivity feature is not always explicitly included. Section 7.3.1 briefly introduces the topical locality phenomenon, used by focused crawlers to look for pages related a given topic. Section 7.4 converges on the adaptive focused crawling approaches explicitly based on AI agent or multi-agent systems, and particularly on genetic algorithms and Ant-paradigm based approaches. Section 7.5 presents an overview of further focused crawling approaches that aim at extracting important features from the

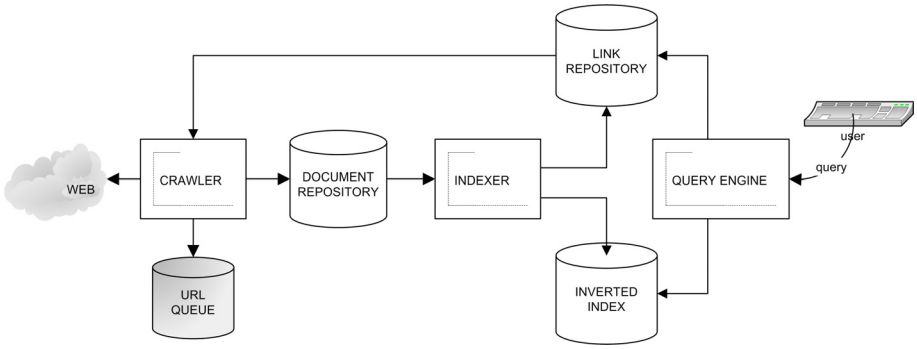

Fig. 7.3. A general overview of a search engine architecture [2].

crawled pages in order to adapt the search behavior. Finally, Sect. 7.6 gives a brief account on the methodologies chosen so far to evaluate focused crawlers.

7.2 Crawlers and the World Wide Web

Analyzing a general search engine's architecture, as pictured in Fig 7.3, it is possible to recognize some sub-systems. An internal repository stores a copy of the retrieved documents, which will be used to extract links and convert the plain text into an *ad hoc* representation, called *Inverted Index*, used to provide quick responses for a given query. This index makes it possible to build lists of documents containing a specific keyword in a very short time. The global result list is drawn simply by merging all these sublists, and by ranking each resource with a given weighting algorithm.

The crawler is a very important module of a search engine. It is a high-performance system, expected to download hundreds of millions of pages over several weeks, minimizing the time spent for crash recoveries. Therefore, its design presents many challenges, such as I/O and network efficiency, robustness and manageability [15, 45, 64, 76, 33]. Moreover, it is meant to ensure a *politeness policy* that sets an interval between two connections to the same Web server, in order to avoid excessive traffic that could cause network congestions or software crashes.

Due to the peculiar characteristics of the Web domain, its vastness, heterogeneity and owing to the authors quickly changing the page contents, many different research fields focusing on the characteristics of this environment have been drawing a lot of interest over the past years. Some of these works are important to understand and develop focused crawlers, for this reason we report some references in the following sections.

7.2.1 Growth and Size of the Web

Estimating the size of the Web is a tough task. Central repositories of the Web content are missing, therefore it is impossible to draw statistics of the size and growth. A recent study revealed that at the beginning of 2005, the part of the Web considered potentially indexable by major engines was at least 11.5 billion pages [43]. Previous studies came to the conclusion that the size of the Web doubled in less than two years [52, 53]. A further consideration that has been widely accepted is that the network speed has improved less than storage capacities, from the single user to big organizations.

Other works try to estimate how often the Web pages change during a given period. For example, after having monitored a corpus of more than 0.7 million pages on a daily basis for four months, it has been estimated that 40% of the page set changed within a week, and 23% of the pages that fell into the `.com` domain changed daily [26]. An extended analysis on 150 million pages can be found in [36].

In order to keep results as fresh as possible, the search engines' local copies of remote Web pages must be kept up-to-date. It is possible to develop refresh policies that guarantee a low consumption of computational and network resources by estimating the change frequency of individual pages. Such estimation techniques are usually based on analysis and models that predict the change frequency of pages, monitoring how often it is updated over a given period and the last date of change [23, 24, 26]. Freshness of the index and the size of the indexable Web are two interrelated parameters to take under consideration when developing and tuning crawlers. A crawl may last for weeks, so at some point crawlers should begin revisiting previously retrieved pages, to check for changes. This strategy means that some pages could never be retrieved.

An attempt to automatically notify the search engine when particular pages change or a new page is added to a Web site is Google Sitemaps[3]. The webmaster has to keep up-to-date a simple XML file that lists the pages on the Web site and how often they change. Pushing this information from the Web site to search engines reduces the time they spend to find resources and adapts the re-crawling process optimizing the network resources. This approach is particularly suitable for large Web sites of dynamically generated pages, where the XML report can be periodically drawn up by the system.

Distributed Web crawling is another important research area that aims to develop infrastructures where single computational units collaborate and distribute the crawling in order to speed up the download and let the whole system tolerate possible crashes [76, 11]. An important subtopic of Web crawling architectures is that of parallelization policies. They regard the assignment of different sets of URLs to single crawling processes in order to maximize the download rate, avoiding multiple downloads of the same resources and minimizing the overhead from parallelization [25].

[3] http://www.google.com/webmasters/sitemaps/

Nevertheless, after taking into consideration the estimate results on the size and the growth of the Web, it is becoming a common opinion that crawling the Web is not trivial because it is growing faster than our search engine technology. The frequent update of some Web pages, which forces the crawler to re-download a great amount of data in a short time is not the only matter.

7.2.2 Reaching the Web: Hypertextual Connectivity and Deep Web

Bailey *et al.* named *dark matter* all the information that a search engine is not able to access [3]. The possible reasons range from the simple robot exclusion protocol[4] (a simple textual file stored on the Web server used by page authors to avoid undesired visits by the search engines' crawlers), to the lack of connectivity in the link graph [16].

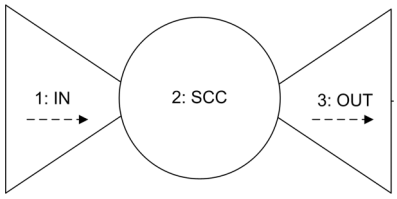

Fig. 7.4. A study on the Web's hypertextual connectivity shows three large sets, where from the IN set a user can reach a page in OUT passing through SCC. The complete bow-tie diagram can be found in [16].

Figure 7.4 shows a diagram where 200 million pages and 1.5 billions of links in a 1999 crawl were analyzed discovering four distinct page sets. The Strongly Connected Component (SCC) is the larger component, accounting for 27%, consisting of pages that can reach one another along directed links. The second and third sets are called IN and OUT, each accounting for 21% approximately, and consist of pages that can reach the SCC but cannot be reached from it, or that are accessible from the SCC but do not link back to it respectively. The rest of the pages, a further 29%, cannot be reached and do not reach SCC. It is obvious how some parts of the Web are not easily reachable. If the crawler starts its navigation from the OUT set, it can reach just a fifth of whole Web.

Another important restriction that reduces the part of the Web available for crawlers is due to the existence of dynamic page generators. For example, commercial Web sites usually offer a text box where the user can submit a query in order to retrieve a list of products. This phenomenon appears each time a page is generated by user interaction with a Web server, an interaction which is more than a simple click of a link. Usually the pages are built by querying an internal database where the information is stored. *Deep Web* is used to indicate the set of all those pages, but sometimes the meaning of the deep Web is expanded

[4] http://www.robotstxt.org/

to include non-textual files, such as audio, movies, etc., for which a traditional search engine would have problems assigning a content representation. It has been estimated that public information on the deep Web is currently up to 550 times larger than the normally accessible Web [8]. Crawlers can collect those product pages in two different ways: generating and submitting a set of keywords capable of retrieving most of the dynamically generated pages [71, 30, 67], and simulating the user's surfing behavior through agents [50].

Studies on search engine coverage confirm how the indexes contain only a small fraction of the Web. In January 2005, among the top 4 search engines, the best was Google, reaching more than 76% of the indexable Web, while Ask/Teoma reached approximately 58% approximately [43]. A previous study [53] assigned less than 40% of the indexable Web to the best search engine. According to most of the search engine experts, dynamically-generated Web pages and the related techniques used to uncover that information during the crawls are currently ignored by search engines, keeping them from being part of the query results. In other words, most of the huge amount of information on the Web is not accessible to the user through search engines.

The vastness of the Web, and in particular of the Deep Web, which is not indexed by traditional search tools, and the low freshness and coverage of search engines suggest how different kind of crawlers have the chance to increase the performance and the accuracy of the search results. Before analyzing focused crawlers in detail, we shall briefly introduce some of the strategies evaluated on the Web domain to partially overcome the aforesaid issues.

7.2.3 Crawling Strategies

Assuming that a search engine is unable to keep its index fresh due to the extent and growth of the Web, research is focusing on the best strategies to prioritize the downloads. In other words, the crawlers can retrieve only a fraction of the Web pages within a given period, so it becomes important to download the most important pages first, that is, the pages that satisfy most of the queries submitted by users.

The implementation of a crawler includes a queue of URLs to be retrieved, fed during the surfing with URLs extracted from pages, the URL QUEUE in Fig. 7.3. The download strategy used by the crawler affects the queue ordering, moving ahead the resources to be crawled first. In abstract terms, the strategy assigns a score to each resource that denotes an estimate of the importance for the current crawl. Of course, at any given moment that score is based only on the information gathered by the crawler until then, that is, the part of the Web that has been crawled.

Cho *et al.* [27] compared three ordering metrics on a corpus of 179,000 pages collected from an university Web site. The metrics are: Breadth-first search, Backlink count and PageRank. The first metric coincides with the well-known search algorithm. If the Web is represented by a graph of pages and a hyperlink structure of edges, the neighbors of a starting edge are considered before ana-

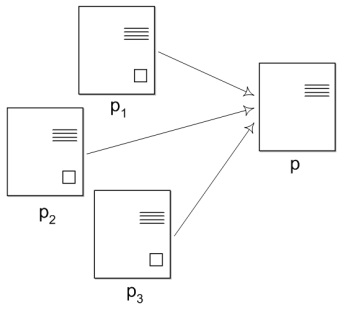

Fig. 7.5. The backlink count metric assigns an importance to a page p based on the number of Web pages pointing to it.

lyzing the neighbors of the visited edges, and so forth. Obviously, the edges are never visited twice.

The *backlink count* metric assigns an importance to each page to be downloaded as a function of the number of the crawled pages that point to it, see Fig. 7.5. Intuitively, if many Web authors decide to put a link to a given site, that site should be judged more important than one that is seldom referenced. This simple metric derives from bibliometry research, that is, the analysis of the citation graph of scholarly articles, and from social network studies. The same research influenced HITS [47] and PageRank [68], two iterative algorithms employed in the Web domain to assign relevance measures to each resource, taking into account the hyperlink structures among pages. In huge collections of documents, query results are often too numerous to be analyzed by users, therefore it becomes important to accurately rank the most important pages. HITS and PageRank try to cope with these problems exploiting the presence of the link structure between documents to unearth the new information used during the ranking process. A better description of these algorithms is available in Chapter 5 of this book [62].

The evaluation in [27] shows that PageRank outperforms the backlink ordering if the goal is to crawl the pages with the highest in-degree, that is, the most popular. The authors show that backlink is more biased by the choice of the starting points, usually reaching locally interesting pages instead of globally interesting ones, focusing more on single clusters. Its behavior does not improve as the crawl proceeds and more information becomes available. If the goal is to crawl pages with content matching a given topic, breadth-first search ensures better performance. It is possible to explain such a result by observing how pages usually point to other pages that share the same topic.

The assets of the breadth-first search are discussed also in [65] where an extended corpus of 328 million of pages has been examined. This strategy is able to discover the highest quality pages assessed by means of the PageRank metric. The authors speculate that there are usually many hosts pointing to high quality pages, and for this reason, regardless of the host or page from which the crawl originates, they can be found sooner during the search.

7.3 Focused Crawling

This section provides an overview of the most important references to focused crawling systems. Even though some of them do not include any adaptivity form, it is useful to analyze the proposed techniques and algorithms before introducing other approaches.

One of the features that characterizes a focused crawler is the way it exploits hypertextual information. Traditional crawlers convert a Web page into plain text extracting the contained links, which will be used to crawl other pages. Focused crawlers exploit additional information from Web pages, such as anchors or text surrounding the links. This information is used to predict the benefit of downloading a given page. Because we do not know anything about the page's content, this prediction avoids to waste CPU and network resources to analyze irrelevant documents.

Basically, all the focused crawlers are based on the topical locality phenomenon. According to this phenomenon, Web pages on a given topic are usually clustered together, with many links that connect one page to another. Once a good page is found, the crawler can analyze its cluster to retrieve pages on the same topic.

A brief description of the topical locality phenomenon and how the hypertextual information is exploited by crawlers in the search activity opens this section.

7.3.1 Exploiting the Hypertextual information

The Web consists of hypertextual resources, typically Web pages, connected by means of links. Traditional IR content-based approaches take into consideration the textual information inside each page in order to assign representations and match them with the user needs. The *hyper information* [56], that is, the information related to a given page that takes into account also the Web structure it is part of, is usually ignored. Besides improving the performance of traditional search engines [15], the information extracted from the link structure is also employed by focused crawlers to navigate among pages in order to find interesting pages. The topical locality phenomenon, the anchors and two hypertextual algorithms, HITS and PageRank, are without doubt important topics to examine in this context.

Topical Locality and Anchors An important phenomenon that lays the foundations of almost all the focused crawlers is *Topical locality*. Davison makes a deep empirical evaluation along with a study on the *proximal cues* or *residues* extracted from links [29], while Menczer extends the study, also considering the link-cluster conjecture, that is, pages about the same topic clustered together [57]. Topical locality occurs each time a page is linked to others with related content, usually because Web authors put a link to another page, to allow users to see further related information or services.

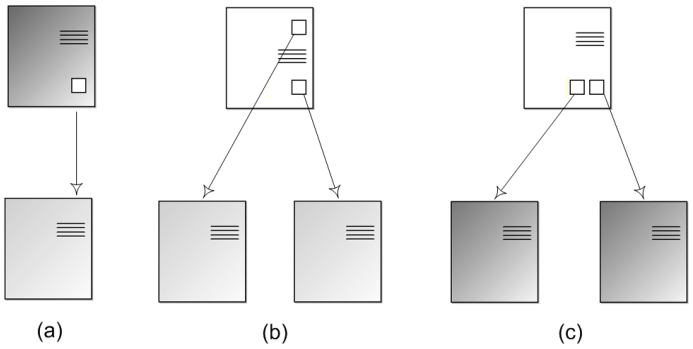

Fig. 7.6. Some phenomena occurring in the hypertextual Web environment: A page is significantly more likely to be topically related to the pages which point to it, (a), and sibling pages are more similar when the links from the parent are located closer together (c) than far away (b).

Focused crawlers exploit this phenomenon crawling clusters of pages each time they find an interesting one. A cluster consists of pages that can be reached by the current page, extracting its links. The crawl usually stops when pages not correlated with selected topics are discovered.

Empirical evaluations acknowledged that this phenomenon is largely common on the Web (see also [28]). A page is much more likely to be topically related to the pages which point to it, as opposed to other pages selected at random or other pages that surround it. Further results show how sibling pages, namely those that share a page that points to both of them, are more similar when the links from the parent are closer together [29], as shown in Fig. 7.6.

The notion of proximal cues or residues corresponds to the imperfect information at intermediate locations exploited by users to decide what paths to follow in order to reach a target piece of information [37, 70]. In the Web environment, text snippets[5], anchor text or icons are usually the imperfect information related to a certain distant content.

Davison [29] shows how anchor text is often very informative about the contents of documents pointed to by the corresponding URL, therefore it may be useful in discriminating unseen child pages. That phenomenon has been exploited for several tasks, such as inferring user information needs [41, 22], document summarization [31] and query translation [55].

Some search engines associate the text of a link with the page the link points to [15], including a partial representation of the linked page in the index, even though that page has not been crawled, nor indexed yet, and despite it does not contain any textual information to be indexed, e.g., Macromedia Flash-based homepages. Moreover, the chance to include text usually written by different

[5] Snippets usually correspond with the short textual description that search engines associate with each query results.

authors often provides more accurate descriptions of Web pages and increases the retrieval performance; for details see the related vocabulary problem [38].

HITS and PageRank The wide research activity on citation analysis started by Garfield in the 1950s is the foundation of two famous hypertextual algorithms, PageRank [68] and HITS [47]. They all aim to assign measures of relevance relying only on the structure of links extracted from sets of Web pages, ignoring the textual content. Their effectiveness has been proved when coupled with traditional IR systems, e.g., [15]. They are employed in more than one crawling algorithm to assign a hypertextual-based rank to the resources to be crawled, and finding new *Seed sets*, i.e., initial pages where the search starts [27, 20, 74].

In general terms, when a huge collection of hypertextual documents needs to be analyzed to retrieve a small set of resources that satisfy a given query, the ranking becomes a very important process. The user usually scans a small number of documents. In a large study on queries submitted to a search engine, it has been noticed that 28.6% of users examine only one page of results, and 19% look at two pages only, therefore half of the users do not check more than 20 URLs [78]. If there are many documents that satisfy a query, the system may assign the same rank to the first results. For this reason, further relevance measures that take into account the popularity, the prestige or the authority of a page must be included in the ranking process.

These kinds of hypertextual algorithms can be successfully employed in focused crawling systems. One of the goals of this type of crawler is to optimize the computation resource to analyze a few pages, ignoring the part of the Web that is not interesting or hardly relevant for the user. Moreover, unlike general-purpose search engines, the number of retrieved pages per query is not an important parameter. Short result lists of highly ranked documents are undoubtedly better than long lists of documents that force the users to sift through them in order to find the most valuable information.

A description of the two link analysis algorithms, HITS and PageRank, is to be found in Chapter 5 of this book [62].

7.3.2 An Overview of the Focused Crawling Approaches

Although one of the first focused crawlers dates back to 1994, we had to wait many years before seeing other approaches in the literature. It is worth mentioning the *Fish-search* algorithm [14, 13] because of its key principle that enhances the traditional crawling algorithms.

The algorithm takes as input one or more starting URLs and the user's query. The starting pages could be collected from the first results of search engines or user's bookmarks, and correspond to the seed URLs. The crawler's queue of pages to be downloaded becomes a priority list initialized with the seed URLs. At each step, the first URL is popped from the queue and downloaded. When a text page becomes available, it is analyzed by a scoring component, evaluating whether it is relevant or irrelevant with respect to the search query. Based on that score, a heuristic decides whether to pursue the exploration in the

related direction or not. If it is not relevant, its links will be ignored by further downloads. See Fig. 7.2 as an example of a focused crawl compared with a crawl based on a breadth-first search.

In the Fish search, whenever a document is fetched, it is scanned for links. A depth value is assigned to each linked page. If the parent is relevant, the depth of the children is set to some predefined value. Otherwise, the depth is set to be one less than the parent's depth. When the depth reaches zero, the direction is dropped and none of its children are inserted into the queue.

It is possible to limit the search to either a fixed period of time or to the retrieval of a certain number of relevant pages. Although the authors provided a number of heuristics to optimize the search, Web server operators noted that Fish-search is not far from an exhaustive search. If all users were to employ a fish-search for every search, the overall bandwidth demand would impose a heavy burden on Web servers.

Further details of the Fish algorithm plus the Shark-search description, which uses a better measure of relevance to draw the similarity between documents and the given query and several other improvements, can be found in [14, 13, 44].

This section also includes other important non-adaptive focused crawling approaches proposed in the literature. A first approach exploits Web taxonomies, such as `ODP` or `Yahoo!`, and the HITS algorithm to predict the importance of the pages to be crawled. Tunneling and Contextual crawling aim at discovering important features to understand where the topical pages are located, following the paths they belong to. Finally, a further approach targets the Semantic Web environment.

Taxonomies and Distillation A popular focused crawler approach is proposed by Chakrabarti *et al.* [20, 19]. This focused crawling system makes use of two hypertext mining programs that guide the crawl: a *classifier* that evaluates the relevance of hypertext documents regarding the chosen topics, and a *distiller* that identifies hypertext nodes that are considered great access points to many relevant pages through a few links, by means of the HITS algorithm.

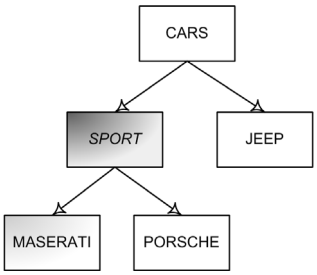

Fig. 7.7. A link contained in a page on `Maserati` cars is visited if the user has selected the `Maserati` topic, or one of its ancestors in the taxonomy, such as `Sport`, during the training of the classifier.

After having collected a set of interesting pages, the user selects the best matching nodes in the category tree of a classifier trained on a given taxonomy, e.g., `Yahoo!`. The hierarchy helps filtering the correct pages during the crawl. Basically, after having retrieved a document, an algorithm finds the leaf node of that tree with the highest probability of similarity. If some ancestor of these nodes has been marked as interesting (see Fig. 7.7), the URLs found in the document will be crawled in future, otherwise the crawl is pruned at the document or a low priority is assigned to them.

Tunneling Focused crawlers heavily rely on the topical locality phenomenon discussed in Sect.7.3.1. A page about a given topic is more likely to point to other pages about the same topic. Even though very useful and effective during the crawl, such a phenomenon might generate some drawbacks. Sometimes, pages of the same topic do not point directly one anther and therefore it is necessary to go through several off-topic pages to get to the next relevant one. Bergmark *et al.* [9] suggest to allow the crawl to follow a limited number of bad pages in order to reach the good ones, naming this technique *tunneling*.

The authors use the terms *nugget* and *dud* to indicate pages correlated with at least one of the given topic collections judged interesting, and the pages that do not match any of the collections; 500,000 pages were downloaded and analyzed in order to find patterns of document-collection correlations along link paths. The idea is to statistically estimate an adaptive cutoff for the tunneling strategy, that is, how far the crawler is allowed to go when starting from an interesting page. Once the cutoff is reached, the crawl in that direction is halted.

The evaluation results show how a focused crawl with adaptive cutoff downloads resources at a higher rate, therefore, exploring the Web faster than a fixed cutoff crawler. At the same time, the adaptive version can limit the amount of irrelevant material it has to crawl through. Briefly, the tunneling technique seems to improve the effectiveness of searching by recognizing and pruning paths which look hopeless.

Contextual Crawling Diligenti *et al.* [32] try a different approach address the very problem of assigning the right importance to different documents along a crawl path following a different approach. Given a set of interesting documents, by querying some search engines they are able to build a representation of the pages that occur within a certain link distance of those documents. In particular, the Web is *backcrawled*; the analyzed pages are the ones that point to the set of interesting documents. All the retrieved information is stored in a structure named *context graph* which maintains for each page the related distance, defined as the minimum number of links necessary to traverse in order to reach a page in the initial set.

The context graph is passed to a set of Naïve Bayes classifiers that identify the categories according to the expected link distance from the page to the target documents. In this way, given a generic document, the crawler predicts how many steps away from a target document it is likely to be. Of course, documents that are expected to lead more quickly to the interesting documents are crawled

sooner. The only relevant drawback of the context-focused crawling approach is the need for search engines providing the reverse link information needed to build the context graph.

Semantic Web A further focused crawling approach [35] is related to the Semantic Web[6], the current effort to enhance the traditional Web presentation languages such as HTML and help grasp the meaning of data, and let autonomous agents understand and interact with information sources. Ontologies are one of the basic elements needed to describe static domain knowledge in terms of common domain structures to be reused by different systems. Basically, the crawling approach aims to define a relevance measure to map a Web page content with an existing ontology provided by users. Textual matches against the ontology's lexicon and taxonomic relationships between super and sub-concepts for calculating accurate relevance scores are undoubtedly the core elements of that approach. The comparison between standard focused crawling approaches and simple keyword matching shows relevant improvements in global performance.

7.4 Agent-based Adaptive Focused Crawling

In recent years, the research community has tried to propose new original approaches in order to build focused crawlers. Some of them speculate on the Web and the analogy with huge environments where single autonomous units live and keep moving, looking for interesting resources. From this point of view, the AI field has developed a wide range of architectures, models and algorithms providing a solid foundation where it has been possible to build interesting prototypes. This section provides information on two algorithms, a genetic-based one and an ant paradigm-based one, including the related focused crawling techniques.

7.4.1 Genetic-based crawling

One of the most renowned adaptive focused crawlers is *InfoSpiders*, also known as *ARACHNID* (Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery) [58] based on genetic algorithms. Even though the prototype is developed using reinforcement learning [81], which will be discussed in Sect. 7.5.2, we decided to discuss it in this section because it is also based on an agent-based approach.

Genetic algorithms have been introduced in order to find approximate solutions to hard-to-solve combinatorial optimization problems. Those kinds of algorithms are inspired by evolutionary biology studies. Basically, a population of chromosomes encoded by means of a particular data structure evolves towards a potential solution through a set of genetic operators, such as inheritance, mutation, crossover, etc.. The chromosomes that get closer to best solutions have

[6] http://www.w3.org/2001/sw/

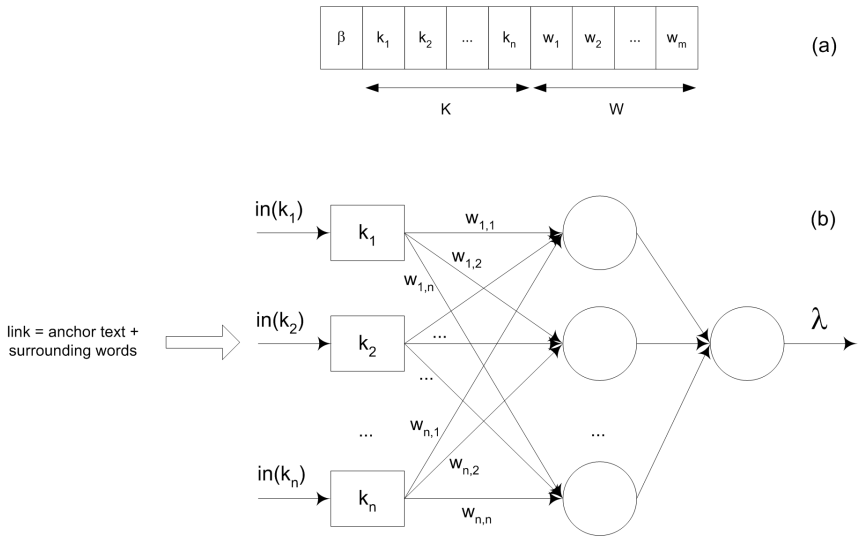

Fig. 7.8. An Infospiders's genotype (a) composed of a set of keywords K and weights W of a neural network used to evaluate a link, that is, the anchor text plus the surrounding words, contained in a Web page (b).

more chances to live and reproduce, while the ones that are ill-suited for the environment die out. The initial set of chromosomes is usually generated randomly and for this reason the first solutions are more likely to be rather poor.

The random mutations between two chromosomes is an important feature of these algorithms in preventing solution sets from converging to sub-optimal or local solutions. For the same reason, even pairs of organisms that do not receive high fitness, namely, not close to good solutions, have a chance to breed.

As in a real environment, if a lot of changes occur all of a sudden, many individuals of the population risk dying out. On the contrary, if those changes take place gradually, the species can gradually evolve along with it. The whole evolution process is repeated until a solution that is judged fairly close to the optimal solution is found.

In InfoSpiders an evolving population of intelligent agents browses the Web driven by user queries, imitating the humans during their browsing activities. The agents are able to assess the relevance of resources with a given query and to reason autonomously about future actions regarding the next pages to be retrieved, with little or no interaction among them.

Each agent is built on top of a *genotype* (shown in Fig. 7.8), basically a set of chromosomes, which determines its searching behavior. The same genotype takes part in the offspring generation process. The first component of an agent's genotype is a parameter β that represents the extent to which an agent trusts the textual description about the outgoing links contained in a given page. The other two components are: a set of keywords K initialized with the query terms and a vector of real-valued weights W, initialized randomly with uniform distribution.

The latter component corresponds to the information stored in a feed-forward neural network used for judging what keywords in the first set best discriminate the documents relevant to users. Basically, for each link in a page, an agent extracts the set of keywords that appear around it and that are also included in the genotype set. The set is the input to the neural network. The function *in()*, within the network, weights each word, counting their occurrences and calculating their positions on the anchor text. More importance is given to terms that are closer to the link.

The adaptivity is both supervised and unsupervised. If the user has previously submitted a relevance feedback on the document chosen by the agent, the related content is used to evaluate the retrieved documents (supervised learning). Otherwise, the relevance assessments are employed for training the neural network's weights W, comparing the outcome of the agent's behavior with the initial query terms corresponding to the profile of user interests. The network's weights are updated through the backpropagation of error [73] according to the agents' actions, e.g., irrelevant pages are retrieved, in order to alter the subsequent behavior of the agent.

Two functions, *benefit()* and *cost()* are employed to determine the energy gain related to a link selection. The former gets high values if the keywords K are contained in the visited document, while the *cost* function is correlated to resources used during the retrieving of documents, e.g., network load or document size. The whole crawling process is summarized in Algorithm 1.

Algorithm 1 Pseudo-code of the InfoSpiders algorithm.

```
{initialize each agent's genotype, energy and starting page}
PAGES ← maximum number of pages to visit
while number of visited pages < PAGES do
  while for each agent a do
    {pick and visit an out-link from the current agent's page}
    {update the energy estimating benefit() − cost()}
    {update the genotype as a function of the current benefit}
    if agent's energy > THRESHOLD then
      {apply the genetic operators to produce offspring}
    else
      {kill the agent}
    end if
  end while
end while
```

The second form of adaptivity to the environment is achieved by mutations and crossovers among agents. They guarantee the inclusion of relevant resources, they spawn new offspring and adapt agents to the environment. Each single agent learns and exploits important local features of the environment. The multi-agent system as a whole captures more general features, which represent resources that are relevant to users.

The keyword vector K is subjected to mutation during the reproduction of the agent population. A stochastic approach takes into consideration both local context, selecting a word that describes documents that led to the energy increase, and global context, selecting words from documents where the user has submitted relevance feedback. According to the authors, that form of keyword evolution, operated through local selection, mutation and crossover, implements a kind of selective query expansion. The neural net is mutated by adding some random noise to the set of weights W.

Basically, a value corresponding to the agent's energy is assigned at the beginning of the search, and it is updated according to the relevance of the pages visited by that agent. Both neural networks and genetic algorithms select terms from those documents that are considered relevant. They are also responsible for using agents' experience to modify the behavior of the agents, and for identifying new links leading to the energy increase. The energy determines which agents are selected for reproduction and the ones to be killed amidst the population.

The agent architecture based on a set of computational units facilitates a multi-process scalable implementation that can be adapted to the current resources, in terms of available network bandwidth and CPU.

Chen *et al.* [21] compared a genetic-based focused crawler, called *Itsy Bitsy spider*, with a Best-first search approach, where a function assigns a score to each outgoing link from the collection of retrieved pages, corresponding to the priority to visit the link. The links with the highest score are the ones the crawler visits first.

Starting from a set of pages related to the current user information needs, the genetic-based crawler extracts words and links from that set. By means of a particular similarity measure among sets, it is possible to compare two different search space states and evaluate the outcome of a system's action, that is, the download of a particular link. The similarity measure is the same one used as a priority function in the best-first search approach.

The mutation operator is implemented performing a search on the YAHOO! database in order to suggest new, promising unknown pages for further explorations. The crossover operates on sets of pages connected to the seed set. The probability to perform one of the two operators during the crawl is a system's parameter. In this way, depending on whether the user prefers to locally crawl a part of the Web or aims at the whole Web, the probability to use the mutation operator gets lower or higher values with respect to crossover.

During the evaluation, the genetic approach does not outperform the best first search. The recall values of the former are significantly higher than the best first search, but precision is not statistically different. Nevertheless the mutation operator includes resources that would be hardly retrievable through conventional search processes. This kind of generation of new configurations is also considered in the Yang and Chen's crawler [82], where a hybrid version of simulated annealing is employed instead of genetic algorithms. Simulated annealing is a generalization of the Monte Carlo method for combinatorial problems inspired by the annealing process of liquids and metals [60]. A further evaluation, where

an improved version of InfoSpiders is compared against the best-first search is discussed in [59].

It should be pointed out that the agent architecture and the adaptive representation of InfoSpiders, consisting of a set of computation units with single genotypes, differ considerably from the Itsy Bitsy spider approach. The neural network and the other components of the genotype make it possible for a single agent to autonomously determine and adapt effectively its behavior according to both the local context of the search and the personal preferences of the user. That feature is basically missing in the Itsy Bitsy spider.

Balabanović [6, 5] combines the collaborative and content-based approaches in a single multi-agent recommender system named *Fab*. *Collection* agents search the Web according to profiles representing their current topics. The users are grouped into clusters according to the similarity of their profiles, which do not usually correspond to the collection agents' profile. The topic of a set of search agents can be of interest to many users, while one user can be interested in many topics. The retrieved pages are forwarded to those users whose profiles they match beyond some threshold. If a page is highly rated by one user, it is passed to the users with similar profiles (user's cluster).

Collection agents are based on topical locality, and perform a best-first search of the Web. An agent follows the links in a given page if it is pertinent to the given topic. Other kinds of agents construct queries to be submitted to various search engines and randomly collect links from human-suggested *cool* Web sites.

The population of collection agents evolves by adapting to the population of users. The agents that retrieve pages rarely seen or that receive low feedback scores by users are killed. If the agent is able to retrieve good pages, it is given the chance to reproduce itself. Hence, the population dynamically adapts to the user requests, to the environment and to the available computing resources.

7.4.2 Ant-based crawling

Research on ant foraging behaviors inspired a different approach for building focused crawlers. In [39, 40] an adaptive and scalable Web search system is described, based on a multi-agent reactive architecture, stemming from a model of social insect collective behavior [12]. That model has been created by biologists and ethologists to understand how blind animals, such as ants, are able to find out the shortest ways from their nest to feeding sources and back. This phenomenon can be easily explained, since ants can release a hormonal substance to mark the ground, the *pheromone*, leaving a trail along the followed path. This pheromone allows other ants to follow such trails, reinforcing the released substance with their own.

Intuitively, the first ants returning to their nest from the feeding sources are those which have chosen the shortest paths. The back-and-forth trip (from the nest to the source and back) allows them to release pheromone on the path twice. The following ants leaving the nest are attracted by this chemical substance. If they have to select a path, they will prefer the one which has been frequently covered by the other ants that followed it previously. This is the reason why

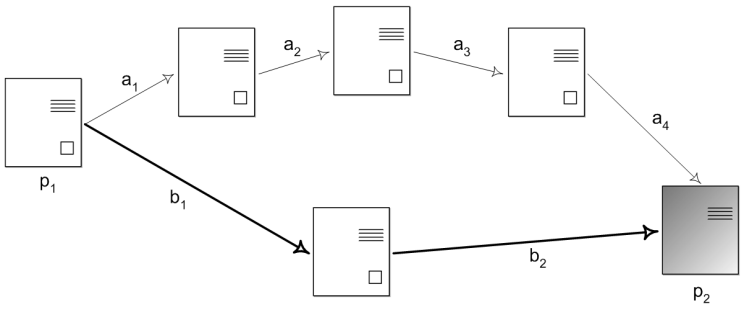

Fig. 7.9. If two paths lead to an interesting page, the first agents that reach that page are the ones that have followed the shortest path $\langle b_1, b_2 \rangle$, and therefore they are the first to release the pheromone that attracts the following agents to the same path.

they will direct themselves towards the shortest paths, which are the first to be marked (see Fig. 7.9).

The decision-making of each computational unit is realized through a set of simple behaviors, which allows the agents to wander from one document to another, choosing the most promising paths, i.e., those leading to resources relevant for users. The outcome of the agent exploration is indirectly handed out to other agents, which can exploit it to improve future explorations. This form of social ability ensures the indispensable adaptivity needed to to work in complex and dynamic environments, such as the Web.

Aside from the topical locality phenomenon (Sect. 7.3.1), a further observation widely discussed in [63] is taken into consideration: to reach the current page, Web page authors suppose that surfers have followed a special path and, therefore, have visited and read certain pages too. Therefore, the contents of Web pages are often not self-contained: they do not always contain all the information required to completely understand and explain what they deal with. In order to satisfy user information needs, the sequences of connected pages should be considered as a virtual information unit that could be suggested to users [46]. For this reason, during an exploration, the pages on the path through which a surfer reaches the page under examination, i.e., the current *context* of the exploration, are not to be ignored [9].

Each agent corresponds to a virtual ant that has the chance to move from the hypertextual resource where it is currently located url_i, to another url_j, if there is a link in url_i that points to url_j. At the end of each exploration, the pheromone trail is released on the agent's route, that is, sequence of URLs of the followed path. When the agent is located in a certain resource, it can match the related content with the user's needs, and measure the amount of pheromone on the paths corresponding to the outgoing links.

The pheromone trails allows ants to make better local decisions with limited local knowledge both on environment and group behavior. The ants employ them to communicate the exploration results one another: the more interesting resources an ant finds, the more pheromone trail it leaves on the followed path.

As long as a path carries relevant resources, the corresponding trail will be reinforced and the number of attracted ants will increase.

The system execution is divided into cycles; in each one of them, the ants make a sequence of moves among hypertextual resources. The maximum number of allowable moves depend proportionally on the value of the current cycle, that is, $maxmoves = k \cdot currentcycle$ where k is a constant. At the end of a cycle, the ants update the pheromone intensity values of the followed path as a function of the retrieved resource scores.

This increases the number of the allowed movements and, consequently, the number of the visited resources, as the number of cycles increases too. During the first cycles, the trails are not so meaningful, due to the small set of crawled resources, so the exploration is basically characterized by a random behavior. But after a certain number of cycles, the paths that permit to find interesting resources will be privileged with a major intensity of pheromone, therefore, a great number of ants will have the chance to follow them. In other words, the process is characterized by a positive feedback loop where the probability to follow a particular path depends on the number of ants that followed it during the previous cycles.

It is possible to describe the operations that the ant-agents perform with these *task accomplishing behaviors*:

1. at the end of the cycle, the agent updates the pheromone trails of the followed path and places itself in one of the start resources
2. if an ant trail exists, the agent decides to follow it with a probability which is function of the respective pheromone intensity
3. if the agent does not have any available information, it moves randomly

Intelligent behavior emerges from the interaction of these simple behaviors, and from the interaction that agents establish with their environment. The resources from which the exploration starts can be collected from the first results of a search engine, or the user's bookmarks, and correspond to the seed URLs.

To select a particular link to be followed, a generic ant located on the resource url_i at the cycle t, draws the *transition probability* value $P_{ij}(t)$ for every link contained in url_i that connects url_i to url_j. The $P_{ij}(t)$ is considered by the formula:

$$P_{ij}(t) = \frac{\tau_{ij}(t)}{\sum_{l:(i,l)\in E} \tau_{il}(t)} \tag{7.1}$$

where $\tau_{ij}(t)$ corresponds to the pheromone trail between url_i and url_j, and $(i, l) \in E$ indicates the presence of a link from url_i to url_l (see Fig. 7.10).

To keep the ants from following circular paths, and to encourage page exploration, each ant stores a list L containing the visited URLs. A probability related to the path from url_i to url_j, if the url_j belongs to L, is 0. At the end of every cycle, the list is emptied out.

When the limit of moves per cycle is reached, the ants start the *trail updating process*. The *updating rule* for the pheromone variation of the k-ant corresponds to the mean of the visited resource scores:

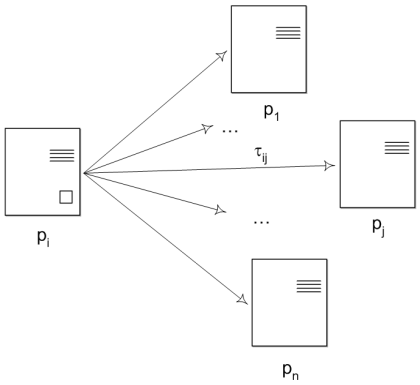

Fig. 7.10. When an agent selects a link to follow, it draws the transition probabilities P_{ij} of all the out-going links $i \rightarrow 1, i \rightarrow 2, \ldots, i \rightarrow n$ from the current page p_i, i.e., $\forall j : (i,j) \in E$, according with the pheromone trails τ_{ij}, as in Eq. 7.1.

$$\Delta\tau^{(k)} = \frac{\sum_{j=1}^{|p^{(k)}|} score(p^{(k)}[j])}{|p^{(k)}|} \tag{7.2}$$

where $p^{(k)}$ is the ordered set of pages visited by the k-ant, $p^{(k)}[i]$ is the i-th element of $p^{(k)}$, and $score(p)$ is the function that, for each page p, returns the similarity measure with the current information needs: a $[0,1]$ value, where 1 is the highest similarity.

The process is completed with the τ value updates. The τ_{ij} trail of the generic path from url_i to url_j at the cycle $t+1$ is affected by the ant's pheromone updating process, through the computed $\Delta\tau^{(k)}$ values:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \sum_{k=1}^{M} \Delta\tau^{(k)} \tag{7.3}$$

where ρ is the trail evaporation coefficient. It must be set to a positive value less than 1 to avoid unlimited accumulation of substance caused by the repeated positive feedback. The summation widens to a subset of the N ants living in the environment. In order to avoid the implementation of the third accomplishing behavior, at the beginning of the execution, all the $\tau_{ij}(0)$ values are set to a small constant τ_0. Algorithm 2 shows each step of the crawling process.

The developed architecture and, in particular, the trail updating process, permit to take into consideration two empirical observations: Web pages on a given topic are more likely to link to those on the same topic, and the content of Web pages is often not self-contained. It is easy to notice how the pheromone attraction bears the ants' exploration toward pages linked to the most interesting ones. Moreover, if a path leads to a page that is not deemed very interesting, the probabilities to follow its outgoing links could be relevant just the same, because the pheromone trail on a link is function of all the pages' scores in the path. In

Algorithm 2 Pseudo-code of the Ant-based crawler.

```
{initialize each agent's starting page}
PAGES ← maximum number of pages to visit
cycle ← 1
t ← 1
while number of visited pages < PAGES do
  while for each agent a do
    for move = 0 to cycle do
      {calculate the probabilities P_ij(t) of the out-going links as in Eq. 7.1}
      {select the next page to visit for the agent a}
    end for
  end while
  {update all the pheromone trails}
  {initialize each agent's starting page}
  cycle ← cycle + 1
  t ← t + 1
end while
```

this way, the context of a page, that is the content of the pages from which it is possible to reach a page, is also inspected.

As for the adaptability, there is a twofold instance: the first concerns the opportunity for users to refine queries during the execution when the results are not so satisfactory or, in general, when users do not know how to input a query to express what they want. The second form regards the possibility of analyzed hypertextual resource changes due to environment instability. Of course, this adaptability does not directly concern the user and variations of his/her needs, but it is related to the environment's dynamics. These two types of adaptivity are possible because at every cycle the value of pheromone intensities $\tau_{ij}(t)$ is updated according to the visited resource scores.

Once the query, or the content of the visited resources changes, the similarity measure changes affects the $\Delta\tau_{ij}(t)$ variation which influences, in its turn, $\tau_{ij}(t)$ (see the pheromone updating process), that is, the results of past explorations. Moreover, as cycles go by, the current values of pheromone intensities tend to become ideal, i.e. function of the current environmental status, with the pheromone evaporation effects.

For instance, given a particular path P, in case of a negative variation of the $\Delta\tau_{ij}^{k}(t)$ values (due to, for instance, a change of the user query), where the path from url_i to url_j belongs to P, the trails $\tau_{ij}(t+1)$ are subjected to a feedback which is reduced if it is compared to the one in the former cycles. For this reason, a smaller number of ants will be attracted, therefore, the $\Delta\tau_{ij}^{k}(t+2)$ increments are still further reduced, and so forth. In other words, each change in the environment causes a feedback consequently modifying the system behavior in order to adapt itself to the new conditions.

The major drawback of this approach is that agents need to start the crawl from the same set of pages at the end of each cycle. As the crawl widens, many pages are re-crawled many times before discovering new resources on the frontier,

wasting network and computational resources. Periodically updating the seed set with new pages is a possible way to tackle this problem.

The proposed architecture is a reactive one, where the system's rational behavior emerges from the interaction and the cooperation of an agent population; therefore, it must be attempted to completely understand the relationship between local and global behavior. In particular, it is useful to be aware of the changes of the global behavior in function of some parameter variations, such as the initial values and the decay of the pheromone intensities, or the number of agents employed. For this purpose, the agents' behavior has to be analyzed, i.e. how agents choose paths in function of interesting resource positions, to see if they behave properly under all the possible circumstances.

7.5 Machine Learning-based Adaptive Focused Crawling

In the last section, two approaches based on genetic algorithms and animal foraging models have been described. Other adaptive focused crawling approaches exploit different techniques in order to represent and recognize features that can drive the search toward the interesting resources. In this section, some of these approaches based on machine learning are briefly introduced.

7.5.1 Intelligent Crawling statistical model

Aggarwal *et al.* [1] introduce an interesting adaptive crawling framework, called *Intelligent Crawling*. It aims to statistically learn the characteristics of the Web's linkage structure while performing the search. In particular, given an unseen page, a set of customizable features named *predicates* (e.g., the content of the pages which are known to link to the unseen URL, or tokens in the unseen page's URL), are taken into consideration. Those features are analyzed in order to understand how they are connected to the probability that the related unseen page satisfies the information needs.

The feature set and the linkage structure of that portion of the Web which has already been crawled are the input of the statistical model. At the beginning of the search, no statistical information is available and each single feature gets the same importance, therefore the crawler virtually behaves randomly. As soon as an interesting page is downloaded, its features are analyzed in order to find any possible correlation. For example, if the fraction of pages about music composer *Bach* is 0.3%, and it is found that 10% of the pages that point to them contain the word *Bach* in their content, following the terminology in [1], we may as well assert that:

$$P(C|E) > P(C) \tag{7.4}$$

the particular knowledge E about a candidate URL, e.g., the 10% of pages that contain the word *Bach*, increases the prior probability P(C) that a given Web page will be related to user needs. In order to understand if a given feature,

namely, knowledge E, is favorable to probability $P(C)$, the *interest ratio* function is defined:

$$I(C,E) = \frac{P(C|E)}{P(C)} = \frac{P(C \cap E)}{P(C)P(E)} \tag{7.5}$$

An approximation of $P(C \cap E)$ and $P(E)$ can be obtained using the information that the crawler has retrieved so far, as well as $P(C)$, fraction of retrieved pages that satisfy the user defined predicate. The interest ratio gets values greater than 1 if a feature E is favorable to the probability that the unseen page satisfies the user needs. Otherwise, if event E is unfavorable, the interest ratio will be within the range $(0, 1)$.

For example, the interest ratio for the event corresponding to the word *Bach* occurring in the in-linking page is $0.1/0.03 = 3.33$. Once those ratios identify the best features to be taken into consideration, the crawler uses them to decide the candidate pages that are more likely to satisfy the user's needs.

The interesting point of this framework is that it does not need any collection of topical examples for training. At the beginning, users specify their needs by means of some predicates, e.g., the page content or the title must contain a given set of keywords, and the crawler adapts its behavior learning correlations among the given features. Nevertheless, choosing the right set of predicates and/or adding new predicates besides the ones proposed by authors might not be an easy task to accomplish.

7.5.2 Reinforcement Learning-based Approaches

The idea of using textual information contained in pages that point to the ones to be evaluated during the download is exploited also by Chakrabarti *et al.* [18]. Their work starts from a traditional focused crawler [20], where a classifier evaluates the relevance of a hypertextual document with respect to the chosen topics. The training of the classifier is done at the beginning of the crawl, where the user selects the best matching nodes in the category tree of a hierarchical classifier. The enhancement of the crawler concerns the inclusion of a second classifier called *apprentice*, which assigns priorities to unvisited URLs in the crawl frontier. In order to draw these priorities, the classifier extracts some features from a particular representation of structured documents, named Document Object Model (DOM)[7], of the pages that point to unvisited URLs.

The original focused crawler's classifier, whose role was to assign a similarity measure to crawled pages given some user needs, now trains instances for the apprentice. Basically, for each retrieved page v, the apprentice is trained on information from the original classifier and on some features around the link extracted from crawled pages that point to v. Those predictions are then used to calculate if it is worth traversing a particular URL, and therefore order the queue

[7] The Document Object Model is an interface that allows programs and scripts to dynamically access and process the document's content and structure, see `http://www.w3.org/DOM/` for details.

of URLs to be visited. The evaluation shows how the false positives decrease significantly between 30% and 90%.

In [72] reinforcement learning techniques are employed to map what actions have generated a benefit during the crawling. That map becomes important each time the system decides the next action to undertake, evaluating the future reward expected from executing it.

The interesting documents found in the environment are the rewards, while following a particular link corresponds to an action. The authors make some simplifying assumptions concerning the representation of an environment state, that is, the remaining set of interesting documents to be found, and the set of links that have been discovered, in order to deal with the problem. Basically, the idea is to learn during the crawl the text in the neighborhood of hyperlinks that are most likely to point to relevant pages. For this reason, each link is replaced with the surrounding words, so it is possible to generalize across different hyperlinks by comparing the related text. A collection of Naïve Bayes text classifiers perform the mapping by casting this regression problem as classification.

7.6 Evaluation Methodologies

Defining an evaluation methodology for a *standard* crawler does not require a great effort. Once a subset of the Web is available, it is possible to run an instance of the crawl on a workstation and monitor the most important parameters to measure its effectiveness, such as: computation time to complete the crawl, or the number of downloaded resources per time unit. Since the Web corpus is fixed, different crawling strategies and the related results can be directly compared. Some technical issues must be addressed in order to construct a collection that is a good sample of the Web [4]. Many important elements should be considered when defining an evaluation methodology regarding a *focused* crawling system. In this section, we briefly mention some of these elements, referring to literature for a deeper analysis.

One of the first evaluation parameters to take into consideration is the soundness of the retrieved documents. The traditional crawlers' goal is to download as many resources as possible, whereas a focused crawler should be able to filter out the documents that are not deemed related to the given topics of interest. All the research in the information retrieval domain can help us define measures of soundness for the retrieved results, but some of them, such as precision and recall, become meaningless in the Web domain.

In order to evaluate the retrieval effectiveness, precision P_r corresponds to the fraction of top r ranked documents that are relevant to the query over the total number of retrieved documents, interesting and not:

$$P_r = \frac{found}{found + false\ alarm} \tag{7.6}$$

while recall R_r is the proportion of the total number of relevant documents retrieved in the top r over the total number of relevant documents available in

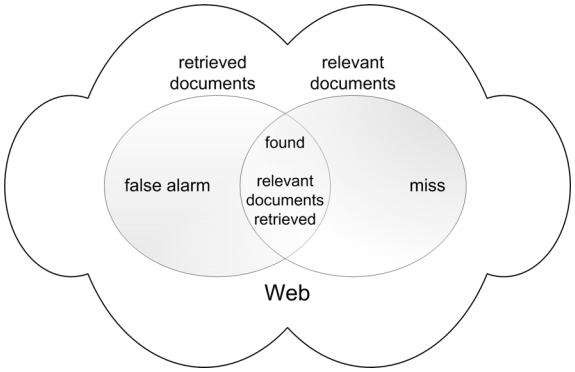

Fig. 7.11. The Web and the two sets, retrieved and relevant documents, which sometimes do not correspond. Precision and recall aim at measuring this deviation.

the environment:

$$R_r = \frac{found}{found + miss} \tag{7.7}$$

see also the diagram in Fig. 7.11.

As pointed out in [20], the recall indicator is hard to measure because it is impossible to clearly derive the total number of relevant documents present on the Web.

The Information Retrieval community has identified many approaches based on very large collections to provide objective testbeds for evaluations. A standard collection guarantees the *reliability* propriety, essential to obtain the same results when tests are repeated under the same initial conditions and the chance to compare the results among different systems. Moreover, it is possible to clearly point out the sets of relevant documents and therefore calculate the traditional IR evaluation measures. Nevertheless, focused crawlers should access the Web directly, to avoid that during the search some paths be ignored because of some pages not being included in the collection. For this reason, standard collections are rarely employed for evaluations of focused crawlers, and it is almost impossible to make comparisons from results obtained by the different algorithms.

In order to evaluate the goodness of the retrieved resources, many evaluations use the same measures employed in the focused crawler's algorithms. For example, the classifiers' outcome in [20], the VSM-based similarity measures in [39], and the importance metrics in [27] are employed both to guide the search of crawlers and to evaluate the relevance of the retrieved resources. That approach may appear improper, but actually the measures are defined without considering the document set to be evaluated, therefore, the results do not affect the definition of evaluation measures.

A further evaluation measure often employed during the evaluation is the percentage of important pages retrieved over the progress of the crawl. The goal is to select relevant pages only, avoiding to spend time crawling the uninteresting regions of the Web. If the percentage is high, the crawler is able to save its

computational resources and retrieve a good page set. Chakrabarti *et al.* [20] consider the average quality measures of the retrieved pages for different time slices of the crawl. In this way, it is possible to see if crawlers get lost during the search or if they are able to constantly keep the crawl over the relevant documents.

The time spent for crawling and analyzing the documents is another important element to measure. A focused crawler usually collects thousands of URLs per hour. Of course, computational resources, such as the available network bandwidth, memory and CPU, required for the processing, affect the crawl time. Some algorithms, such as HITS, based on the link graph, need plenty of time to draw their relevance measures, especially for complex graphs. Focused crawlers based on these algorithms, e.g., [27, 20, 74], are expected to reduce the rate of page downloads as the crawl goes on.

In [79] an extended analysis of the literature about focused crawling suggests a general framework for crawler evaluation research. That framework considers important issues, such as the choice of seed pages where the crawls start. Of course, if seed pages are related to the topic of the crawl, the search for related pages is much easier because of the topical locality phenomenon (see Sect.7.3.1). The framework also provides a mechanism to control the level of difficulty of the crawl task by means of the hypertextual structure among pages.

At present, focused crawling evaluations that also include adaptivity analysis are not available. One of the reasons could be the difficulty to measure the reaction of crawlers to user needs refinements, or alterations of sub-sets of Web pages. How long does it take to adapt the crawl to a user relevance feedback and provide new interesting documents? How many environment alterations are tolerable before the crawling performance falls below a given threshold? Standard methodologies to assess those features, thus allowing to compare results, are yet to be developed.

7.7 Conclusions

Focused crawling has become an interesting alternative to the current Web search tools. It concerns the development of particular crawlers able to seek out and collect subsets of Web pages related to given topics. Many domains may benefit from this research, such as vertical portals and personalized search systems, which provide interesting information to communities of users. Some of the most interesting approaches have been described, along with important hypertextual algorithms, such as HITS and PageRank.

Particular attention has been given to adaptive focused crawlers, where learning methods are able to adapt the system behavior to a particular environment and input parameters during the search. Evaluation results show how the whole searching process may benefit from those techniques, enhancing the crawling performance. Adaptivity is a must if search systems are to be personalized according to user needs, in particular if such needs change during the human-computer interaction.

Besides new crawling strategies that take into consideration peculiar characteristics of the Web, such as topical locality, we can expect future research to head in several directions. Some techniques used to look into the dark matter (hidden Web), and Natural Language Processing (NLP) analysis can help understand the content of Web pages and identify user needs. In this way, the effectiveness of crawlers can be improved both in terms of precision and recall.

Extending the focused crawlers' range to consider resources available on the Semantic Web is a further research topic, which will gain importance as this new technology that allows data to be shared and reused across applications becomes more popular in the Web community. Current prototypes of focused crawlers do not consider the reuse of past experience - namely, what crawlers have discovered and analyzed from past crawls - in new explorations of the Web. The chance to exploit such data in the crawling process could help retrieving information more quickly, reducing the network and computational resources.

References

1. Aggarwal, C.C., Al-Garawi, F., Yu, P.S.: Intelligent crawling on the world wide web with arbitrary predicates. In: Proceedings of the 10th World Wide Web Conference (WWW10), Hong Kong (2001) 96–105 `http://www10.org/cdrom/papers/110/`.
2. Arasu, A., Cho, J., Garcia-Molina, H., Paepcke, A., Raghavan, S.: Searching the web. ACM Transactions on Internet Technology (TOIT) **1**(1) (2001) 2–43
3. Bailey, P., Craswell, N., Hawking, D.: Dark matter on the web. In: Poster Proceedings of the 9th World Wide Web Conference(WWW9), Amsterdam, Netherlands (2000) `http://www9.org/final-posters/poster30.html`.
4. Bailey, P., Craswell, N., Hawking, D.: Engineering a multi-purpose test collection for web retrieval experiments. Information Processing and Management **39**(6) (2003) 853–871
5. Balabanović, M.: Exploring versus exploiting when learning user models for text recommendation. User Modeling and User-Adapted Interaction **8**(1-2) (1998) 71–102
6. Balabanović, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Communications of the ACM **40**(3) (1997) 66–72
7. Beitzel, S.M., Jensen, E.C., Chowdhury, A., Grossman, D., Frieder, O.: Hourly analysis of a very large topically categorized web query log. In: SIGIR '04: Proceedings of the 27th annual international conference on Research and development in information retrieval, New York, NY, USA, ACM Press (2004) 321–328
8. Bergman, M.K.: The deep web: Surfacing hidden value. The Journal of Electronic Publishing **7**(1) (August 2001)
9. Bergmark, D., Lagoze, C., Sbityakov, A.: Focused crawls, tunneling, and digital libraries. In: ECDL '02: Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries, London, UK, Springer-Verlag (2002) 91–106 `http://citeseer.ist.psu.edu/bergmark02focused.html`.
10. Bharat, K., Kamba, T., Albers, M.: Personalized, interactive news on the web. Multimedia Systems **6**(5) (1998) 349–358
11. Boldi, P., Codenotti, B., Santini, M., Vigna, S.: Ubicrawler: a scalable fully distributed web crawler. Software, Practice and Experience **34**(8) (2004) 711–726

12. Bonabeau, E., Dorigo, M., Theraulaz, G.: Inspiration for optimization from social insect behavior. Nature **406** (2000) 39–42
13. Bra, P.D., Houben, G.J., Kornatzky, Y.: Information retrieval in distributed hypertexts. In: Proceedings of the 4th RIAO, Intelligent Multimedia, Information Retrieval Systems and Management, New York, NY, USA (1994) 481–491 http://citeseer.ist.psu.edu/debra94information.html.
14. Bra, P.D., Post, R.: Searching for arbitrary information in the www: The fish-search for mosaic. In: Proceedings of the 2nd World Wide Web Conference (WWW2), Chicago, USA (1994) http://citeseer.ist.psu.edu/172936.html.
15. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems **30** (1998) 107–117
16. Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J.: Graph structure in the web. In: Proceedings of the 9th World Wide Web Conference (WWW9), Amsterdam, Netherlands (2000) 309–320 http://www9.org/w9cdrom/160/160.html.
17. Chakrabarti, S.: Recent results in automatic web resource discovery. ACM Computing Surveys **31**(4es) (1999) 17
18. Chakrabarti, S., Punera, K., Subramanyam, M.: Accelerated focused crawling through online relevance feedback. In: WWW '02: Proceedings of the 11th international conference on World Wide Web, New York, NY, USA, ACM Press (2002) 148–159 http://www2002.org/CDROM/refereed/336/.
19. Chakrabarti, S., van den Berg, M., Dom, B.: Distributed hypertext resource discovery through examples. In: VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1999) 375–386 http://www.vldb.org/conf/1999/P37.pdf.
20. Chakrabarti, S., van den Berg, M., Dom, B.: Focused crawling: A new approach to topic-specific web resource discovery. In: Proceedings of the 8th World Wide Web Conference (WWW8), Toronto, Canada (1999) 1623–1640 http://www8.org/w8-papers/5a-search-query/crawling/index.html.
21. Chen, H., Chung, Y.M., Ramsey, M., Yang, C.C.: A smart itsy bitsy spider for the web. Journal of the American Society for Information Science **49**(7) (1998) 604–618
22. Chi, E.H., Pirolli, P., Chen, K., Pitkow, J.: Using information scent to model user information needs and actions on the web. In: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI2001), Seattle, WA, USA (2001) 490–497
23. Cho, J., Garcia-Molina, H.: The evolution of the web and implications for an incremental crawler. In: VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2000) 200–209
24. Cho, J., Garcia-Molina, H.: Synchronizing a database to improve freshness. In: SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM Press (2000) 117–128
25. Cho, J., Garcia-Molina, H.: Parallel crawlers. In: WWW '02: Proceedings of the 11th international conference on World Wide Web, New York, NY, USA, ACM Press (2002) 124–135 http://www2002.org/CDROM/refereed/108/index.html.
26. Cho, J., Garcia-Molina, H.: Estimating frequency of change. ACM Transactions on Internet Technology (TOIT) **3**(3) (2003) 256–290
27. Cho, J., Garcia-Molina, H., Page, L.: Efficient crawling through url ordering. Computer Networks and ISDN Systems **30**(1–7) (1998) 161–172

28. Chung, C., Clarke, C.L.A.: Topic-oriented collaborative crawling. In: CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management, New York, NY, USA, ACM Press (2002) 34–42
29. Davison, B.D.: Topical locality in the web. In: SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM Press (2000) 272–279
30. de Carvalho Fontes, A., Silva, F.S.: Smartcrawl: a new strategy for the exploration of the hidden web. In: WIDM '04: Proceedings of the 6th annual ACM international workshop on Web information and data management, New York, NY, USA, ACM Press (2004) 9–15
31. Delort, J.Y., Bouchon-Meunier, B., Rifqi, M.: Enhanced web document summarization using hyperlinks. In: HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia, New York, NY, USA, ACM Press (2003) 208–215
32. Diligenti, M., Coetzee, F., Lawrence, S., Giles, C.L., Gori, M.: Focused crawling using context graphs. In: VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2000) 527–534 http://www.vldb.org/conf/2000/P527.pdf.
33. Diligenti, M., Maggini, M., Pucci, F.M., Scarselli, F.: Design of a crawler with bounded bandwidth. In: WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, New York, NY, USA, ACM Press (2004) 292–293 http://www2004.org/proceedings/docs/2p292.pdf.
34. Doorenbos, R.B., Etzioni, O., Weld, D.S.: A scalable comparison-shopping agent for the world-wide web. In: AGENTS '97: Proceedings of the first international conference on Autonomous agents, New York, NY, USA, ACM Press (1997) 39–48
35. Ehrig, M., Maedche, A.: Ontology-focused crawling of web documents. In: SAC '03: Proceedings of the 2003 ACM symposium on Applied computing, New York, NY, USA, ACM Press (2003) 1174–1178
36. Fetterly, D., Manasse, M., Najork, M., Wiener, J.: A large-scale study of the evolution of web pages. In: WWW '03: Proceedings of the 12th international conference on World Wide Web, New York, NY, USA, ACM Press (2003) 669–678 http://www2003.org/cdrom/papers/refereed/p097/P97sources/p97-fetterly.html.
37. Furnas, G.W.: Effective view navigation. In: CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, ACM Press (1997) 367–374
38. Furnas, G.W., Landauer, T.K., Gomez, L.M., Dumais, S.T.: The vocabulary problem in human-system communication. Communications of the ACM **30**(11) (1987) 964–971
39. Gasparetti, F., Micarelli, A.: Adaptive web search based on a colony of cooperative distributed agents. In Klusch, M., Ossowski, S., Omicini, A., Laamanen, H., eds.: Cooperative Information Agents. Volume 2782., Springer-Verlag (2003) 168–183
40. Gasparetti, F., Micarelli, A.: Swarm intelligence: Agents for adaptive web search. In: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004). (2004) 1019–1020 http://citeseer.ist.psu.edu/738711.html.
41. Gasparetti, F., Micarelli, A.: User profile generation based on a memory retrieval theory. In: Proc. 1st International Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces (WPRSIUI'05). (2005) 59–68 http://citeseer.ist.psu.edu/gasparetti05user.html.

42. Gauch, S., Speretta, M., Chandramouli, A., Micarelli, A.: User profiles for personalized information access. In Brusilovsky, P., Kobsa, A., Nejdl, W., eds.: The Adaptive Web: Methods and Strategies of Web Personalization. Volume 4321 of Lecture Notes in Computer Science. Springer-Verlag, Berlin Heidelberg New York (2007) this volume
43. Gulli, A., Signorini, A.: The indexable web is more than 11.5 billion pages. In: WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web, New York, NY, USA, ACM Press (2005) 902–903 http://www.cs.uiowa.edu/~asignori/web-size/.
44. Hersovicia, M., Jacovia, M., Maareka, Y.S., Pellegb, D., Shtalhaima, M., Ura, S.: The shark-search algorithm an application: tailored web site mapping. In: Proceedings of the 7th World Wide Web Conference (WWW7), Brisbane, Australia (1998) 317–326 http://www7.scu.edu.au/1849/com1849.htm.
45. Heydon, A., Najork, M.: Mercator: A scalable, extensible web crawler. World Wide Web **2**(4) (1999) 219–229
46. Joachims, T., Freitag, D., Mitchell, T.M.: Webwatcher: A tour guide for the world wide web. In: Proceedings of the 15h International Conference on Artificial Intelligence (IJCAI1997). (1997) 770–777 http://citeseer.ist.psu.edu/16829.html.
47. Kleinberg, J.: Authoritative sources in a hyperlinked environment. In: Proceedings of the 9th annual ACM-SIAM symposium on Discrete algorithms, San Francisco, CA, USA (1998) 668–677 http://www.cs.cornell.edu/home/kleinber/auth.pdf.
48. Kruger, A., Giles, C.L., Coetzee, F.M., Glover, E., Flake, G.W., Lawrence, S., Omlin, C.: Deadliner: building a new niche search engine. In: CIKM '00: Proceedings of the ninth international conference on Information and knowledge management, New York, NY, USA, ACM Press (2000) 272–281 http://citeseer.ist.psu.edu/kruger00deadliner.html.
49. Kumar, R., Novak, J., Raghavan, P., Tomkins, A.: On the bursty evolution of blogspace. In: WWW '03: Proceedings of the 12th international conference on World Wide Web, New York, NY, USA, ACM Press (2003) 568–576 http://www2003.org/cdrom/papers/refereed/p477/p477-kumar/p477-kumar.htm.
50. Lage, J.P., da Silva, A.S., Golgher, P.B., Laender, A.H.F.: Automatic generation of agents for collecting hidden web pages for data extraction. Data and Knowledge Engineering **49**(2) (2004) 177–196
51. Lau, T., Horvitz, E.: Patterns of search: analyzing and modeling web query refinement. In: UM '99: Proceedings of the seventh international conference on User modeling, Secaucus, NJ, USA, Springer-Verlag New York, Inc. (1999) 119–128
52. Lawrence, S., Giles, L.C.: Searching the world wide web. Science **280** (1998) 98–100
53. Lawrence, S., Giles, L.C.: Accessibility of information on the web. Nature **400** (1999) 107–109
54. Levene, M., Poulovassilis, A.: Web dynamics. Software Focus **2**(2) (2001) 60–67
55. Lu, W.H., Chien, L.F., Lee, H.J.: Translation of web queries using anchor text mining. ACM Transactions on Asian Language Information Processing (TALIP) **1**(2) (2002) 159–172
56. Marchiori, M.: The quest for correct information on the web: Hyper search engines. In: Proceedings of the 6th World Wide Web Conference (WWW6), Santa Clara, CA, USA (1997) 1225–1235 http://www.w3.org/People/Massimo/papers/WWW6/.
57. Menczer, F.: Lexical and semantic clustering by web links. Journal of the American Society for Information Science and Technology **55**(14) (2004) 1261–1269

58. Menczer, F., Belew, R.K.: Adaptive retrieval agents: Internalizing local context and scaling up to the web. Machine Learning **31**(11–16) (2000) 1653–1665
59. Menczer, F., Pant, G., Srinivasan, P.: Topical web crawlers: Evaluating adaptive algorithms. ACM Transactions on Internet Technology **4**(4) (2004) 378–419
60. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A., Teller, E.: Equations of state calculations by fast computing machines. Journal of Chemical Physics **21**(6) (1953)
61. Micarelli, A., Gasparetti, F., Sciarrone, F., Gauch, S.: Personalized search on the world wide web. In Brusilovsky, P., Kobsa, A., Nejdl, W., eds.: The Adaptive Web: Methods and Strategies of Web Personalization. Volume 4321 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, and New York (2007) this volume
62. Micarelli, A., Sciarrone, F., Marinilli, M.: Web document modeling. In Brusilovsky, P., Kobsa, A., Nejdl, W., eds.: The Adaptive Web: Methods and Strategies of Web Personalization. Volume 4321 of Lecture Notes in Computer Science. Springer-Verlag, Berlin Heidelberg New York (2007) this volume
63. Mizuuchi, Y., Tajima, K.: Finding context paths for web pages. In: Proceedings of the 10th ACM Conference on Hypertext and Hypermedia: Returning to Our Diverse Roots (HYPERTEXT99), Darmstadt, Germany (1999) 13–22
64. Najork, M., Heydon, A.: High-performance web crawling. In Abello, J., Pardalos, P.M., , Resende, M.G., eds.: Handbook of massive data sets. Kluwer Academic Publishers, Norwell, MA, USA (2002) 25–45
65. Najork, M., Wiener, J.L.: Breadth-first search crawling yields high-quality pages. In: Proceedings of the 10th World Wide Web Conference (WWW10), Hong Kong (2001) 114–118 `http://www10.org/cdrom/papers/208/`.
66. Ntoulas, A., Cho, J., Olston, C.: What's new on the web?: the evolution of the web from a search engine perspective. In Feldman, S.I., Uretsky, M., Najork, M., Wills, C.E., eds.: Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004, ACM (2004) 1–12 `http://www2004.org/proceedings/docs/1p1.pdf`.
67. Ntoulas, A., Zerfos, P., Cho, J.: Downloading textual hidden web content through keyword queries. In Marlino, M., Sumner, T., III, F.M.S., eds.: ACM/IEEE Joint Conference on Digital Libraries, JCDL 2005, Denver, CA, USA, June 7-11, 2005, Proceedings, ACM (2005) 100–109
68. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project (1998) `http://dbpubs.stanford.edu/pub/1999-66`.
69. Pinkerton, B.: Finding what people want: Experiences with the webcrawler. In: Proceedings of the 2nd World Wide Web Conference(WWW2), Chicago, USA (1994) 821–829
70. Pirolli, P., Card, S.K.: Information foraging. Psychological Review **106** (1999) 643–675
71. Raghavan, S., Garcia-Molina, H.: Crawling the hidden web. In: VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2001) 129–138
72. Rennie, J., McCallum, A.: Using reinforcement learning to spider the web efficiently. In: ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1999) 335–343 `http://citeseer.ist.psu.edu/7537.html`.

73. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In Rumelhart, D.E., McClelland, J.L., eds.: Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations. MIT Press, Cambridge, MA, USA (1986) 318–362
74. Rungsawang, A., Angkawattanawit, N.: Learnable topic-specific web crawler. Journal of Network and Computer Applications **28**(2) (2005) 97–114
75. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill, New York (1983)
76. Shkapenyuk, V., Suel, T.: Design and implementation of a high-performance distributed web crawler. In: Proceedings of the 18th International Conference on Data Engineering (ICDE'02), Washington, DC, USA, IEEE Computer Society (2002) 357
77. Spink, A., Jansen, B.J.: A study of web search trends. Webology **1**(2) (December 2004) `http://www.webology.ir/2004/v1n2/a4.html`.
78. Spink, A., Wolfram, D., Jansen, M.B.J., Saracevic, T.: Searching the web: the public and their queries. Journal of the American Society for Information Science **52**(3) (2001) 226–234
79. Srinivasan, P., Menczer, F., Pant, G.: A general evaluation framework for topical crawlers. Information Retrieval **8**(3) (2005) 417–447
80. Steele, R.: Techniques for specialized search engines. In: Proc. Internet Computing 2001, Las Vegas (June 25–28 2001) `http://citeseer.ist.psu.edu/steele01techniques.html`.
81. Sutton, R.S., Barto, A.G.: Introduction to Reinforcement Learning. MIT Press, Cambridge, MA, USA (1998)
82. Yang, C.C., Yen, J., Chen, H.: Intelligent internet searching agent based on hybrid simulated annealing. Decision Support Systems **28**(2) (2000) 269–277
83. Yuwono, B., Lam, S.L.Y., Ying, J.H., Lee, D.L.: A World Wide Web resource discovery system. In: Proceedings of the 4th World Wide Web Conference (WWW4), Boston, Massachusetts, USA (1995) 145–158 `http://www.w3.org/Conferences/WWW4/Papers/66/`.