Available online at www.sciencedirect.com
ScienceDirect
Information Processing and Management 43 (2007) 609–623
INFORMATION PROCESSING & MANAGEMENT
www.elsevier.com/locate/infoproman

Architecture of a grid-enabled Web search engine

B. Barla Cambazoglu, Evren Karaca, Tayfun Kucukyilmaz, Ata Turk, Cevdet Aykanat *

Computer Engineering Department, Bilkent University, 06800 Bilkent, Ankara, Turkey

Received 1 January 2006; received in revised form 10 October 2006; accepted 13 October 2006
Available online 11 December 2006

Abstract

Search Engine for South-East Europe (SE4SEE) is a socio-cultural search engine running on the grid infrastructure. It offers a personalized, on-demand, country-specific, category-based Web search facility. The main goal of SE4SEE is to attack the page freshness problem by performing the search on the original pages residing on the Web, rather than on the previously fetched copies as done in the traditional search engines. SE4SEE also aims to obtain high download rates in Web crawling by making use of the geographically distributed nature of the grid. In this work, we present the architectural design issues and implementation details of this search engine. We conduct various experiments to illustrate performance results obtained on a grid infrastructure and justify the use of the search strategy employed in SE4SEE.

Keywords: Search engine; Web crawling; Text classification; Grid computing

1. Introduction

In this age of information, search engines act as important services, providing the community with the information hidden in the Web and, due to their frequent use, stand as an integral part of our lives. The last decade has witnessed design and implementation of several state-of-the-art search engines (Page & Brin, 1998). Today, there are search engines that have indexed more than four billion Web pages, processing millions of user queries per day over their local index.

A traditional search engine is typically composed of three pipelined components (Arasu, Cho, Garcia-Molina, & Raghavan, 2001): a crawler, an indexer, and a query processor. The crawler component is responsible for locating, fetching, and storing the content residing within the Web. The downloaded content is concurrently parsed by an indexer and transformed into an inverted index (Tomasic, Garcia-Molina, & Shoens, 1994; Zobel, Moffat, & Sacks-Davis, 2002), which represents the downloaded collection in a compact and efficiently queryable form. The query processor is responsible for evaluating user queries and returning to the users the pages

* Corresponding author. Tel.: +90 312 290 1625; fax: +90 312 266 4047.
E-mail addresses: berkant@cs.bilkent.edu.tr (B.B. Cambazoglu), ekaraca@cs.bilkent.edu.tr (E. Karaca), ktayfun@cs.bilkent.edu.tr (T. Kucukyilmaz), atat@cs.bilkent.edu.tr (A. Turk), aykanat@cs.bilkent.edu.tr (C. Aykanat).

0306-4573/$ - see front matter
doi:10.1016/j.ipm.2006.10.011

relevant to their query. Despite the fact that many research efforts were spent, effectiveness and efficiency still remain as the two major challenges in the Web search problem.

The effectiveness problem appears in both Web crawling and query processing. In Web crawling, effectiveness is related to the freshness of the indexed pages (Cho & Garcia-Molina, 2000), which is highly correlated with the crawling efficiency, i.e., if pages are more frequently downloaded, it is more probable that the cached copies of the pages are fresh. In query processing, effectiveness refers to the precision and recall measures, which evaluate the accuracy and coverage of the results, respectively (Clarke, Cormack, & Tudhope, 2000; Can, Altingovde, & Demir, 2004; Wilkinson, Zobel, & Sacks-Davis, 1995).

In addition to the effectiveness problem, both Web crawling and query processing have an efficiency problem. The efficiency problem in Web crawling (Cambazoglu, Turk, & Aykanat, 2004) is due to the large scale of the Web as well as the Web's constantly evolving nature, which require pages to be downloaded and indexed frequently. According to the results reported by Google, it takes around a month to recrawl the same page again on the average. The efficiency problem in query processing is due to the need to quickly evaluate a query over a rather large index (Cambazoglu & Aykanat, 2006; Can et al., 2004; Long & Suel, 2003), in the presence of many user queries being submitted concurrently. The state-of-the-art search engines attack this second problem using some algorithmic optimizations that may trade effectiveness for improved efficiency (Moffat, Zobel, & Sacks-Davis, 1994; Wong & Lee, 1993; Turtle & Flood, 1995) (e.g., short-circuit evaluation) or programming improvements (e.g., trying to keep the whole Web index in the volatile memory). But, in general, the primary method to cope with both problems is to employ parallel/distributed computing systems, which execute multiple crawler agents to crawl the Web (Cho & Garcia-Molina, 2002) and multiple query engines to evaluate queries over replicated/partitioned copies of the Web index (Baeza-Yates & Ribeiro-Neto, 1999; Ribeiro-Neto & Barbosa, 1998), increasing both page download rates and query processing throughput.

In this work, we present the design and implementation details of a grid-enabled search engine, Search Engine for South-East Europe[1] (SE4SEE), which somewhat differs from the above-mentioned, traditional search engines in both its design philosophy and functionality. In short, SE4SEE is a personalized, on-demand, country-specific, category-based search engine running on the grid infrastructure. It provides a Web search facility which combines crawling and classification. SE4SEE primarily addresses the page freshness and efficiency problems in Web crawling by utilizing the computational power and high bandwidth inherently available in the grid and the grid's geographically distributed nature. In this work, we conduct experiments to illustrate the performance of grid-enabled Web search and justify the features specific to SE4SEE.

The organization of the paper is as follows. In Section 2, we provide background information on Web crawling and text classification, which are the basic building blocks of SE4SEE, while justifying the use of the grid. In Section 3, we give a brief survey of the previous work on Web crawling, text classification, and distributed/gridified Web search. Section 4 presents the architecture of SE4SEE and its implementation details. We report the results of the conducted experiments in Section 5. Finally, in Section 6, we conclude and discuss some future work.

2. Preliminaries

2.1. Web crawling

Web crawling is the process of locating, fetching, and storing Web pages. A typical Web crawler, starting from a set of seed pages, locates new pages by parsing the downloaded pages and extracting the hyperlinks within. Extracted hyperlinks are stored in a FIFO fetch queue for further retrieval. Crawling continues until the fetch queue gets empty or a satisfactory number of pages are downloaded. Usually, many crawler threads execute concurrently in order to overlap network operations with CPU processing, thus increasing the throughput.

Although it seems to be a simple task, there are many challenges in Web crawling. The two important issues are coverage and freshness. The coverage refers to the size of the set of pages retrieved within a certain period

[1] SE4SEE homepage, http://se4see.grid.org.tr

of time. A successful crawler tries to maximize its coverage in order to provide a larger, searchable collection to the users. Similarly, the freshness of the collection is important in order to minimize the difference between the cached copies of pages and the originals on the Web, thus keeping the served information up-to-date.

Another important issue in Web crawling is the need for a large amount of computational resources. First, a high amount of processing power is necessary to parse the crawled pages, extract the hyperlinks, and index the pages' content. Second, a large amount of volatile memory is required to store and manage the data structures that grow quickly and continuously during the crawl. The final and most important resource requirement is a high network bandwidth. The network bandwidth determines the page download rate and affects the crawler's coverage as well as the page freshness.

We believe that all these computational requirements make Web crawling a suitable target for grid computing (Foster & Kesselman, 2003). In general terms, the grid can be defined as "a type of a parallel and distributed system that enables sharing, selection, and aggregation of geographically distributed autonomous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements".[2] The grids contain computationally powerful nodes, which have the resources necessary for running a Web crawling application. Furthermore, in cases where the spatial locality of the pages is important, the geographically distributed nature of the grid can be utilized to increase page download rates, as is the case in the design of SE4SEE.

2.2. Text classification

Informally, text classification is the problem of assigning a category to a document from a predefined set of categories. In the literature, various machine learning techniques are employed to solve this problem. Most of these techniques are based on the supervised learning approach, where the classifier is trained by a set of previously labeled set of documents and then is used to predict categories for unseen test documents. The accuracy of the classification depends on the choice of the underlying machine learning algorithm as well as the quality of the documents used for training the classifier.

Most search engines rely on keyword-based search, where a query, consisting of a number of keywords, is evaluated over an inverted index, and the top k documents are returned to the user in decreasing order of their similarity to the query (Lee, Chuang, & Seamons, 1997). However, there are also approaches employing text classification in querying of document collections and/or presentation of the results. The use of text classification in search engines is mainly in the form of pre-classification (e.g., engines providing topic directories manually created by human experts) or post-classification (e.g., engines providing automated classification of the query results). While the former of these increases precision, the latter enhances the presentation of the results. SE4SEE adopts the post-classification approach, where the crawled pages are classified under several topic categories before being presented to the user.

3. Related work

In this section, we survey the previous works on Web crawling, text classification, and search engines. In the literature, there are many research studies concentrating on different issues in Web crawling, such as URL ordering for retrieving high-quality pages earlier (Baeza-Yates, Castillo, Marin, & Rodriguez, 2005; Cho, Garcia-Molina, & Page, 1998; Najork & Wiener, 2001), partitioning the Web for efficient multi-processor crawling (Cambazoglu et al., 2004; Teng, Lu, Eichstaedt, Ford, & Lehman, 1999), distributed crawling (Boldi, Codenotti, Santini, & Vigna, 2002; Zeinalipour-Yazti & Dikaiakos, 2002), and focused crawling (Altingovde & Ulusoy, 2004; Chakrabarti, van den Berg, & Dom, 1999; Diligenti, Coetzee, Lawrence, Giles, & Gori, 2000). Despite this large amount of effort, due to the commercial value of the developed applications, it is hard to obtain robust and customizable crawling software (Heydon & Najork, 1999; Shkapenyuk & Suel, 2002).

For text classification (Lam, Ruiz, & Srinivasan, 1999), an abundance of machine learning algorithms (Sebastiani, 2002; Yang, 1999) such as k-nearest neighbor (Han, Karypis, & Kumar, 2002), naive Bayesian

[2] Grid Computing Info Centre, http://www.gridcomputing.com/gridfaq.html

(McCallum & Nigam, 1998), neural networks (Ng, Goh, & Low, 1997), decision trees (Lewis & Ringuette, 1994), and support vector machines (Sun, Lim, & Ng, 2002) are used in the literature. In Web page classification (Kan, 2004), due to its performance and quality, naive Bayesian classifier is usually preferred. A number of machine learning tools such as Weka (Witten & Frank, 2005), Grid Weka (Khoussainov, Zuo, & Kushmerick, 2004), and the Harbinger machine learning toolkit (Cambazoglu & Aykanat, 2005) are readily available for use in text classification.

Although there are many different Web search engines,[3] the market is dominated by three major engines.[4] These engines have huge multi-processor computing infrastructures consisting of thousands of PCs. However, they are mostly centralized systems, not suitable for crawling geographically distributed Web sites. There are a number of information retrieval works on peer-to-peer environments (Bender, Michel, Triantafillou, Weikum, & Zimmer, 2005), distributed systems (Melnik, Raghavan, Yang, & Garcia-Molina, 2001), and the grid (Scholze, Haya, Vigen, & Prazak, 2004).

MINERVA (Bender et al., 2005) is a peer-to-peer Web search engine, in which each peer independently executes a Web crawler. This peer-to-peer system lacks a central coordinator, and hence there is no control over the coverage of each peer. Consequently, the same pages may be crawled multiple times by different peers, resulting in an overlap of pages. This overlap is a crucial problem in peer-to-peer Web search. MINERVA offers techniques that aim to solve this overlap problem and tries to aggregate the results of independent crawls to generate a global result.

The use of the grid for information retrieval is relatively new. To the best of our knowledge, GRACE[5] is the only attempt to develop a grid-enabled search engine (Scholze et al., 2004). The aim of GRACE is to build a search and categorization tool over the grid. GRACE can use both local directories and the query results of other search engines as a knowledge repository. The main objective of GRACE is to analyze the search results and categorize them via linguistic analysis. In this perspective, GRACE is an unsupervised categorization tool rather than a search engine. In GRACE, the utilization of the grid resources is achieved via parallelism based on the distributed nature of the grid. A user can concurrently run multiple queries over the grid. GRACE, in turn, analyzes the query results, categorizes them, and aggregates the results of multiple queries.

Although GRACE and SE4SEE architectures both aim to utilize the grid resources, their motivations are different. While GRACE categorizes the query results that are based on the results obtained from other search engines, SE4SEE does not depend on the results of other search engines. Instead, the query results are retrieved directly from the Web utilizing geographical closeness in country-specific search. Furthermore, GRACE does not provide a facility for category-specific search, whereas SE4SEE allows users to select and search in a specific category as well as perform a keyword-based search.

4. The SE4SEE Architecture

4.1. Features

Search Engine for South-East Europe (SE4SEE) is an attempt towards developing a grid-enabled search engine that specifically targets the countries in the South-East Europe. It is one of the two selected regional applications developed as a part of the EU-funded SEE-GRID FP6 project,[6] which is the primary initiative for establishing a grid infrastructure in the South-East European countries. As stated in Section 1, SE4SEE is a personalized, on-demand, country-specific, category-based, grid-enabled search engine, currently running on the grid infrastructure formed by the SEE-GRID project. Below, we briefly describe the distinguishing features of SE4SEE.

- Personalized crawling: In traditional search engines, the entire Web is crawled, and the pages are indexed for public search. In SE4SEE, a different crawling approach is taken. For each user query, an individual crawl is

[3] http://www.searchenginewatch.com
[4] http://www.google.com, http://search.yahoo.com, http://search.msn.com
[5] Grace project homepage, http://www.grace-ist.org
[6] SEE-GRID project homepage, http://www.see-grid.org

started over the Web, and the relevant pages are selected from the fresh copies on the Web. This way, up-to-date versions of the pages are evaluated and accuracy of the resulting answer set of pages is enforced.

- On-demand crawling: Unlike traditional search engines, which crawl the Web continuously, in SE4SEE, the crawling task is initiated upon the arrival of a user query. Depending on various factors, this type of on-demand crawling may be time-consuming. However, we believe that this approach is acceptable if (1) the information sought for is fresh and is not indexed yet by traditional search engines (e.g., querying the result of a sport event that finished just 5 min ago) or (2) the user initiating the crawl has no time constraints (e.g., looking for some computer graphics papers to be cited in a PhD thesis).
- Category-based search: SE4SEE has support for category-based search in addition to keyword-based search. In this approach, pages downloaded by the crawler are categorized using a previously trained text classifier. At the completion of the crawl, only the set of pages relevant to the topic category selected by the user is presented.
- Country-specific search: Since one of the initial motivations behind SE4SEE is to develop a socio-cultural search engine, SE4SEE provides country-specific search. In general, country-specific search can be performed based on the language of the page, the country domain of the page URL, or the geographical locality of the hosting site. Currently, in SE4SEE, the pages are resolved according to the top-level domain names, e.g., the user may request only the links in the ''.tr'' domain to be downloaded during the crawl.
- Gridification: SE4SEE is fully enabled to the grid. The computational burden of Web crawling to an individual user is alleviated by the utilization of resources (computational power, storage capacity, and the network bandwidth) available in the grid. In particular, SE4SEE runs on the grid infrastructure established as a part of the SEE-GRID project. By submitting country-specific queries to the servers residing in the target country, SE4SEE aims to exploit the geographical locality of Web pages and grid sites, thus increasing the page crawling throughput.

4.2. Overview of query processing over the grid

Basically, there are two alternatives for parallelism in grid-enabled Web crawling: intra-query or inter-query parallelism. In intra-query parallelism, a query is submitted to multiple grid nodes, and a crawling task is started at the nodes, each crawling a portion of the Web. The crawled pages are than merged into a global answer set. Although this approach offers good performance in reducing the crawling time, issues such as avoiding overlap in local answer sets or communicating inter-node links between crawlers must be addressed (Cho & Garcia-Molina, 2002). Inter-query parallelism, on the other hand, is a coarse-grain parallel approach, targeting high throughput in query processing. In this approach, each computing node completes the whole crawling task on its own. Although we have an ongoing work on intra-query parallelism, the inter-query parallelism approach is currently employed in SE4SEE.

SE4SEE uses the Globus[7] and LCG[8] middleware to interact with the grid infrastructure. As the underlying grid middleware is able to distribute the work evenly, load balancing is not an issue for the current system. Unfortunately, this distribution is only based on the availability of computational resources in the system. Ideally, we also want it to take the maximum and currently available network bandwidths into consideration. Such a distribution is not possible as the middleware is not network-aware. Unless this difficult problem has been solved, a better, bandwidth-based load distribution mechanism is not possible for our application.

The deployment diagram of the SE4SEE application is given in Fig. 1. A user requires a computer with a browser to connect to the Web portal running on the SE4SEE server. In order to prevent the misuse of grid resources, the user is expected to have a valid SE4SEE account, which is verified by the authentication module in the server. The Web portal acts as a mediator between the user and the grid. That is, it converts the user query into a grid job and submits it through a user interface node (UI) to a worker node (WN). UI nodes in the LCG architecture are entry points to the grid; jobs are submitted and their results are received from these. WNs, on the other hand, are responsible for executing the jobs. The crawler and the classification tasks are executed on the WN and the generated crawling/classification output is stored in the resource broker (RB),

[7] Globus homepage, http://www.globus.org/

[8] LCG middleware homepage, http://lcg.web.cern.ch/LCG/activities/middleware.html

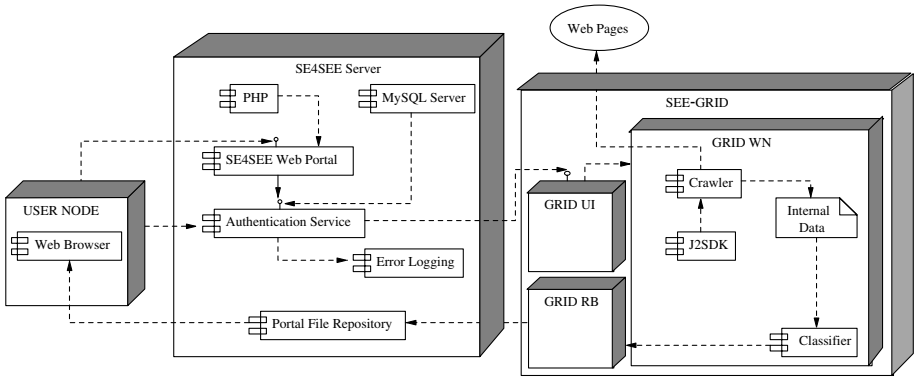

Fig. 1. Deployment diagram of SE4SEE describing the relationship between software and hardware components.

a computer which not only coordinates the jobs and handles their assignments, but is also responsible for the temporary storage of the jobs' input and output. After a time period, the user may transfer the output from the RB to the result repository in the SE4SEE server so that the results can be visualized and permanently stored.

In Fig. 2, we exemplify the job execution in SE4SEE. In the figure, edges show the data flow over the network between different computing systems. In our sample scenario (indicated by bold edges), a user living in Romania performs a search for the hotels located in Croatia. The user connects to the SE4SEE portal located in Ankara through her Web browser and submits the query. The portal transforms the query into an executable grid job and submits the job to an available computing node located in Zagreb, which is highly likely to be geographically close to the target Web pages. A number of hotel pages in the Croatian Web space are located, fetched, and stored in the grid node. When the crawling and classification jobs terminate, the resulting set of pages are retrieved back to the portal. At any time, the user can connect to the Web portal and access the results.

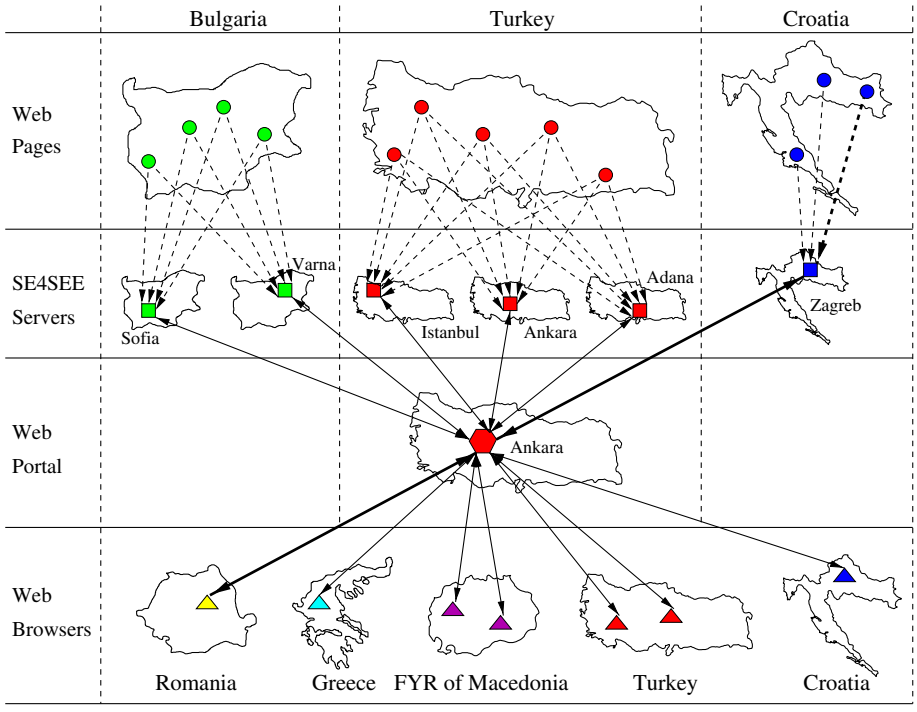

Fig. 2. A sample search scenario over the SE4SEE architecture.

4.3. Components

SE4SEE is composed of three main components: a crawling component, a text classification component, and a Web portal. We provide the details of these components in the following sections.

4.3.1. Web crawler

Since SE4SEE is a ''personal'' search engine, which serves a large number of users each with specific, personal crawling needs, an easily customizable crawler is required. Furthermore, in order to be able to adapt to the heterogeneous nature of the grid infrastructure, a platform independent crawler should be preferred. Such a crawler is capable of executing on different architectures, thus preventing the recompilation overhead and compatibility issues.

The Web crawling component of SE4SEE is implemented in Java utilizing the WebSPHINX[9] interactive development environment for Web crawlers. WebSPHINX is designed to enable and ease the development of personally customized, Web-site-specific, relocatable crawlers and also provides libraries for HTML parsing, pattern matching, and common Web transformations.

The crawler in SE4SEE retrieves the pages in a breadth-first manner (Najork & Wiener, 2001). This approach is more suitable for processing category-based queries, compared to depth-first traversal of pages. Unless a seed URL is provided by the user, the crawls are started from seed pages which contain links to relevant pages for each topic category. Seed pages are selected by human experts from the sites that provide up-to-date links to pages specific to each topic category. The stopping conditions for the crawls are determined by the user, who may specify either the duration of the download or the maximum number of pages crawled.

4.3.2. Text classifier

The Harbinger machine learning toolkit[10] (Cambazoglu & Aykanat, 2005) is used as the text classifier in SE4SEE. This toolkit provides implementations for a number of machine learning algorithms, readily available for use in text classification. There is also built-in support for instance selection, feature selection, and class balancing, which all help in improving the accuracy of classification. In particular, SE4SEE uses the naive Bayesian classifier in this toolkit for Web page classification.

The naive Bayesian classifier tries to capture the global properties of a dataset. It operates on input attributes, which is the vocabulary of the set of training pages in our case. In the training phase of the classifier, the probability of an input attribute being observed in each category is calculated. In the test phase, for each crawled page, the probability of the page belonging to a certain category is determined using the word distribution. For each page and category pair, the classifier generates a probability indicating the degree of relevance between the page and the category. The category with the highest probability is chosen as the category of a page. Despite its assumption that words appear independent of each other, naive Bayesian performs well for Web page classification (Kan, 2004).

The searchable categories in SE4SEE are mostly socio-cultural in nature. The currently provided topic categories are Banks, Dining, Festivals, Hotels, Politics, Sports, Transportation, and Universities. An important issue in successful classification is the selection of high quality Web pages for training. In order to train the classifier, for each category, an equal number of training pages are manually collected from the Web by human experts. Currently, the training pages are only available for Turkey and Croatia, but the training sets for several other countries are expected to be added to the system.

During the training, training pages are passed through several filters. First, whitespace, non-alphanumeric characters, and all HTML tags are eliminated from the pages. Language-specific stemmers were not available at the time of the implementation; hence, no stemming is applied. But, since stopword lists were available for each supported country, stopwords are eliminated. To further reduce the number of non-representative terms in the training pages, feature selection (Lewis, 1992) based on the Chi-square technique is applied. The naive Bayesian classifier is trained with the remaining terms, and a classification model is generated. This classification model is

[9] WebSPHINX homepage, http://www.cs.cmu.edu/~rcm/websphinx
[10] Harbinger homepage, http://bmi.osu.edu/~barla/coding/HMLT

used during the tests to predict a category for each downloaded page whose category is unknown. The pages in a category are ranked in decreasing order of the probabilities before being displayed to the user.

The execution of the classifier is pipelined with the crawler. The crawled pages are passed to the classifier for classification. The classifier is concurrently executed as a separate process, which wakes up regularly and checks if there are pages to be classified. The classifier terminates if there are no new pages after a period of time. The concurrent execution allows the network-bound operation of the crawler to be overlapped by the CPU-bound execution of the classifier, thus reducing the total query execution times.

4.3.3. Web portal

As the only interaction point between the user and the SE4SEE back-end, the Web portal is a major component of the search engine. It has to be user-friendly, even though it requires a more complex interface than classic search engines due to the application's added capabilities. There are several SE4SEE-specific issues that are addressed in the design of the Web portal. The concept of multiple users and jobs has led to the implementation of an authentication system. The inherent batch-like behavior of the crawling task resulted in the addition of a result maintenance mechanism. Finally, the nature of the grid environment led to the introduction of error checking and logging mechanisms.

The long execution times of a typical crawling session, especially when combined with the high task initiation costs of the grid environment, prevent the creation of a real-time search engine. A significant amount of time passes between the submission of a query and the availability of the result, making it impractical for a user to wait for that amount of time. Furthermore, since crawling is a time-consuming task which requires a significant amount of network resources, the retrieved results should be stored for later access. To address this issues, SE4SEE implements a job management system.

There are two types of queries that can be submitted: category- and keyword-based queries. These differ in the seed page selection and page acceptance methodologies. Category-based queries aim to gather pages relevant to a certain topic category by starting from a set of category-specific seed pages, performing classification on all retrieved pages, and returning those whose similarity to the training pages exceed a certain threshold. Keyword-based queries are similar to those in traditional search engines; here, the crawl starts from a user-entered URL and returns the pages that contain the keywords given by the user. No classification is performed on keyword-based queries. Both query types are restricted to user-specified top-level domains to ensure that the crawler stays within a country's Web space. A stopping condition is given along with the query; the procedure continues either until a specified time has passed since the beginning of the crawl or a specified number of pages have been processed.

After a user query is submitted to the portal, the job management system creates an appropriate JDL (Job Description Language) file and a shell script containing the statements to be executed. A copy of the query parameters are saved for future reference. Then, the system locates a computing node where the query can be processed. In country-specific queries, the closest grid nodes are tried to be selected by the system. Once a grid node is determined, the executables of the crawler and text classifier are transferred to the target node. The crawler and text classifier binaries are executed at the target grid node until the user-specified stopping criterion is met. When the job execution completes, the crawled pages are automatically retrieved from the resource broker to the Web portal. The user can then view the results of the search. The results can be saved and recalled multiple times later on, thereby preventing the waste of grid resources by re-querying.

To prevent the extensive use of grid resources, an authorization-based system is implemented. Users need to log on to the system before any grid interaction takes place. A user, once authorized, has the ability to submit queries, manage the crawling tasks and view the results of completed crawls. Both category- and keyword-based queries result in the submission of grid jobs that can be examined and, if desired, aborted. The results for completed crawls are presented in a manner similar to common search engines, along with an option to view the page in the form it was retrieved by the crawler, effectively forming a time-stamped local cache of the results. A keyword search can also be performed in the crawled results, allowing the refinement of presented results without having to resort to additional searches.

Finally, to ensure the durability and security of the system, additional considerations are made. A robust authentication mechanism is implemented, preventing the unprotected storage of passwords. All queries and database accesses are logged. We have mechanisms for intercepting and handling both the errors due to the

failures in the grid infrastructure, reported by the grid middleware, and the errors generated by the application itself. Constraints are placed on certain parameters of the application to prevent misuse of resources and to make the application behave like a ''good citizen'' of the grid community. Hence, the number of crawl jobs that can be performed by a user has been restricted and the stopping conditions of the crawls are capped at sane values.

The pages of the Web portal are prepared using PHP, user actions on these pages invoking external applications that perform the desired tasks. All grid-interaction is over command-line utilities, relying on the robustness of these utilities in unforeseen circumstances. This method also provides a layer of abstraction between the grid and the application code, preventing any changes on grid side having an immediate effect on the application. Any data used in the invocation of these utilities is stored in a regularly backed-up MySQL database, again providing a robust solution for critical information.

5. Experiments

5.1. Platform

As the hardware platform, SE4SEE utilizes the resources available in the grid infrastructure established throughout the SEE-GRID project. These resources, in conformance with the grid philosophy, is composed of a variety of heterogeneous, geographically distributed computational resources. The SEE-GRID infrastructure is essentially a large network of computers that, although located in different regions of South-East Europe, work together to perform a common task. All of our experiments presented in this section are conducted utilizing this infrastructure.

Table 1 summarizes hardware/software characteristics of the grid sites available in the SEE-GRID infrastructure, used in our experiments. In general, it is hard to mention a typical configuration as the individual sites that form the grid have a variety of hardware resources, sometimes even having different configurations within a site. However, broadly speaking, we can say that experiments are conducted computers with an x86 processor clocked at 2.4 GHz or higher, and having at least 512 MB RAM. Although reported in the table, disk capacity is not much of a concern in the experiments since all nodes met the minimum requirement, which has been determined to be 2 GB.[11] Network connectivity of the grid sites was uncertain and had to be measured through experiments. The grid site at the last row of the table is tagged as UI since this site provides the primary interface to the SEE-GRID infrastructure. All other sites are tagged according to their geographical locality.

5.2. Setup

The experiments were performed using the application's command-line back-end. The typical approach of letting the grid infrastructure decide at which site the application runs is avoided for supervised experimentation. Instead, specific sites were chosen manually and jobs are directly submitted to them. Running times for the crawler and classifier were measured by utilizing the executing system's measurement mechanisms and are typically accurate to the millisecond. Scheduling times for the task were derived from the timestamps found on the execution logs provided by the grid middleware. As the nodes on the grid are synchronized using the Network Time Protocol, the derived times are accurate to the order of seconds.

5.3. Results

Five sets of experiments are conducted, where each experiment tries to justify or investigate one of the search features provided by SE4SEE (Section 4.1). First, efficiency of personalized crawling is investigated

[11] According to our experiments, a typical page is 20 KB on the average. For a 100,000-page crawl, this translates to a maximum of 2 GB temporary disk space. However, in practice, this value is much lower since, after fetching, pages are concurrently processed by the text classifier and most are discarded.

Table 1
Characteristics of the grid sites used in the experiments

Tag	Grid site	CPU (GHz)	RAM (GB)	Disk (TB)	Middleware	OS
BA	grid01.pmf.unsa.ba	Intel P4 2.4	0.5	0.036	SL 3.0.5	LCG-2.6.0
HR	grid1.irb.hr	Intel Xeon 2 × 2.8	2	0.03	SL 3.0.3	LCG-2.4.0
MK	grid-ce.ii.edu.mk	Intel P4 3.0	0.5	0.12	SL 3.0.3	LCG-2.4.0
BG	ce001.grid.bas.bg	Intel P4 2.4	0.5	0.1	SL 3.0.3	LCG-2.6.0
TR	grid2.cs.bilkent.edu.tr	Intel P4 3.0	1	0.08	SL 3.0.3	LCG-2.3.0
UI	ce.ulakbim.gov.tr	Intel P4 3.0	1	0.2	SL 3.0.3	LCG-2.6.0

via experiments to have an understanding of the overhead that crawling introduces. Second, experiments are carried out on page freshness to justify the on-demand crawling strategy employed in SE4SEE. Third, we conducted experiments to reveal the benefits of geographically distributed Web crawling. Fourth, we experimented on the overheads introduced by grid-enabled Web search. Finally, we investigated the effectiveness of the category-based search provided by SE4SEE. The following sections present these experiments.

5.3.1. Efficiency

Personalized Web search requires a different crawling/classification task to be initiated over the Web. This is a computationally costly and time-consuming task. In this set of experiments, we try to investigate the efficiency of personalized Web crawling. For this purpose, we crawled and classified varying numbers of pages from the ".edu.tr" domain (Turkish educational sites) and The University of Split. In the experiments, the classifier is executed separately after the crawler finished downloading pages, thus enabling us to measure the relative overheads of the two components more accurately.

Fig. 3 displays the times obtained in crawling and classifying varying number of pages using the grid site denoted with tag UI. The times for archiving/compressing the resulting set of pages are relatively negligible and hence not displayed. According to the figure, although the crawling and classification components have similar overheads at low number of pages, the crawling overhead dominates as the number of pages increases. The results show that personalized search is practical for crawling a fair number of pages. Moreover, in SE4SEE, the crawler and classifier are concurrently executed in a pipelined fashion. Hence, the classification is overlapped with network transfer; the actual total execution time is bounded from above by the sum of the reported execution times of these two components and from below by the maximum of the two values. As also illustrated by this experiment, crawling multiple sites is usually faster than crawling a single site.

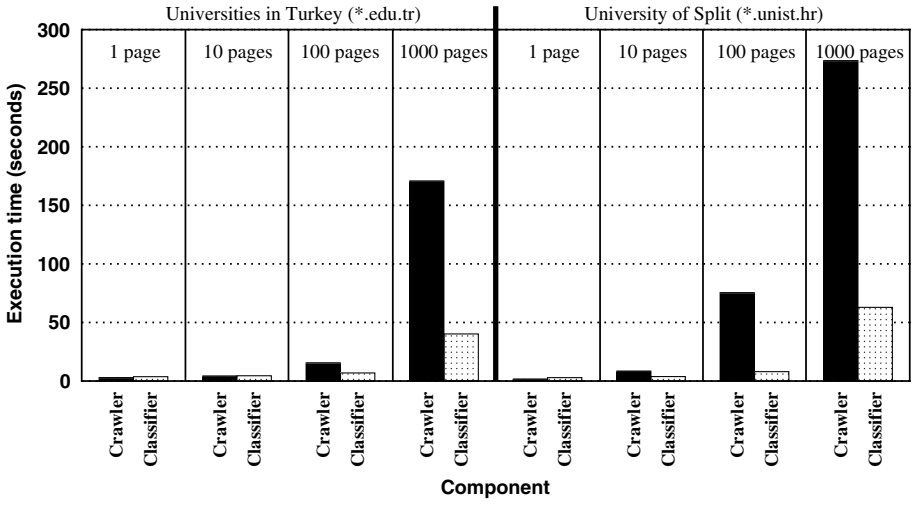

Fig. 3. Performance of Web crawling/classification with increasing number of pages.

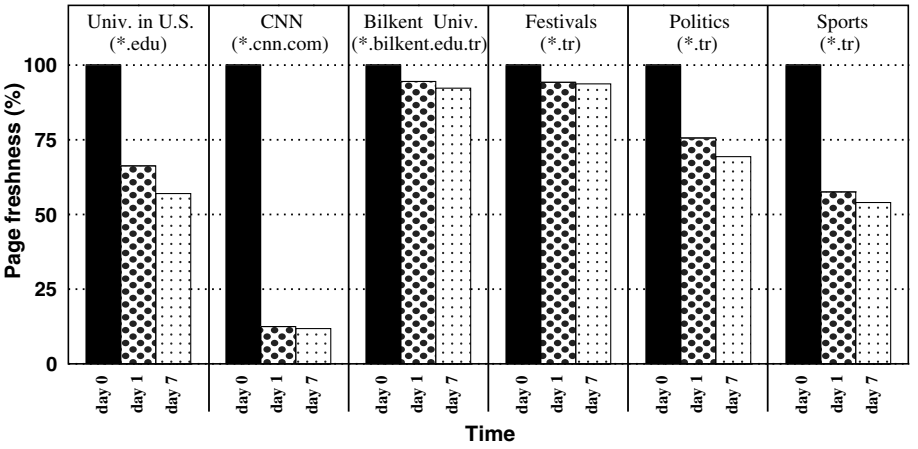

Fig. 4. The variation of page freshness in time for different sites or topic categories.

5.3.2. Page freshness

Since obtaining high page freshness is the one of the motivations behind SE4SEE, we tried to figure out the importance of page freshness via experiments and observed the rate of change in the textual material found in the Web pages (ignoring the HTML content and other information). For this purpose, we first made an initial crawl over a set of Web sites to obtain an initial collection. Throughout a week, the pages in the initial collection were daily recrawled. The freshness $F(t)$ of a crawl at time t is measured by the $F(t) = 100 \times (I - M(t))/I$ formula, where I is the number of pages in the initial collection and $M(t)$ is the number of pages whose content is modified (i.e., updated or deleted) and hence differs from the initial download.

Fig. 4 displays the change of page freshness after $t = 1$ and $t = 7$ days. At the top of the figure, the sites or topic categories are given. The topic categories include sites picked from the training set of pages we manually created. According to Fig. 4, a considerable portion of the pages seems to be modified frequently. Especially, in the CNN Web site, only 12.50% of the pages remain the same after a day. Similarly, after a week, almost half of the educational pages are modified. A similar behavior is not observed in the crawl made over the Bilkent University since this crawl includes pages deep in the directory hierarchy, which have a tendency to be modified less frequently.

Page freshness also shows variation among the topic categories, i.e., while pages belonging to a category remain untouched, pages in some other category may be modified frequently. For example, according to our experiments, the festival pages remain rather static, whereas sports pages are updated more frequently. Overall, we believe that these experiments justify the need for the on-demand crawling strategy employed in SE4SEE, but not available in the traditional search engines.

5.3.3. Geographical locality

A primary benefit of the use of the grid infrastructure in SE4SEE is the geographically distributed nature of the grid sites. Hence, experiments are conducted to investigate the effect of utilizing the grid for geographically distributed Web crawling, where pages are tried to be downloaded by geographically closer servers. Specific sites were chosen as test sites based on their location, and jobs were directly submitted to them. In the experiments, crawling tasks were initiated at five different grid sites, located in Bosnia-Herzegovina (BA), Bulgaria (BG), Croatia (HR), FYROM (MK), and Turkey (TR).

Fig. 5 displays the page crawling throughput (number of pages crawled per minute) achieved by the grid sites for different sets of pages. In this experiment, we first aimed to figure out the typical bandwidth of the individual sites. Note that a closer site with a low network bandwidth might perform worse than a site that is geographically far to the pages, even though the latter has a higher latency with respect to the crawled pages. To avoid misinterpretation of the other results due to the differences in the bandwidth, an approximation of the bandwidth is required. To obtain such a value, a crawl was performed on a Website geographically distant

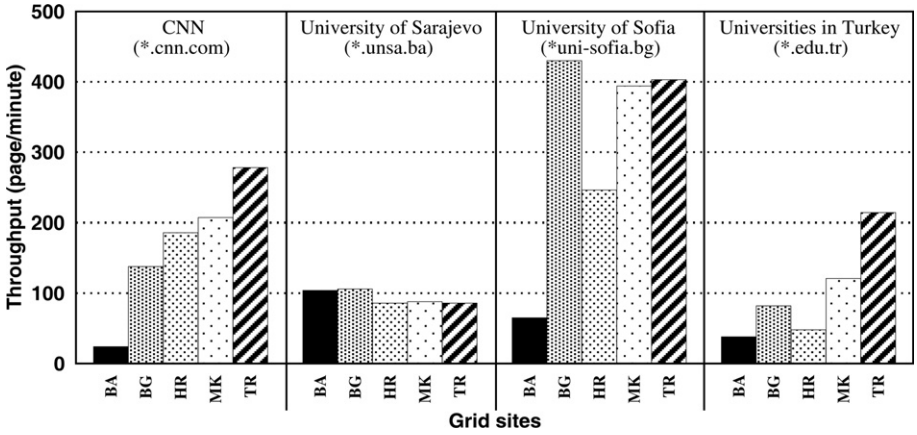

Fig. 5. Effect of geographical locality on crawling throughput.

to all sites, far enough to make any advantages due to the proximity negligible. For this purpose, the CNN site, located in US, is chosen and crawled by all grid sites. This experiment shows that the network capacity of the grid site BA is problematic, whereas the TR site performs relatively better than the rest. However, we must note that even these approximate bandwidths may be misleading since there is a possibility that some sites may have direct satellite connection to target sites, rendering geographical proximity less important.

According to Fig. 5, as expected, each grid site performs well in downloading the pages geographically nearby. Even the BA site, which has a limited bandwidth, achieves a fair throughput in crawling pages from the Web server of the University of Sarajevo. Similarly, the BG and TR sites achieve the highest throughput in crawling pages located Bulgaria and Turkey, respectively. Note that, if the throughputs were normalized with respect to the estimated site bandwidths, in the third experiment (the University of Sofia), the throughput gap between the BG site and the others would be more significant in favor of the BG site. These experimental results indicate that the spatial proximity between the crawling sites and the target pages plays an important role in the crawling throughput, thus justifying the geographically distributed crawling approach of SE4SEE.

5.3.4. Gridification

The overhead of the grid architecture had to be determined to be able to make time-comparisons to classic search engines. To this effect, several crawls of different sizes were made from the same grid site. Execution times for four job phases were extracted from the grid logs: ready, scheduled, running, and fetching. The ready time is the time it takes for a job to be assigned to a site once it has been submitted to the system. The scheduled time indicates how long the job waits at the grid node. The running time is the execution time of the application, and the fetching time is the time it takes for the output to be retrieved form the resource broker. Note that the time it takes for the output to be transmitted from the grid node to the resource broker could not be timed.

The results in Fig. 6 demonstrate the high start-up costs of the grid infrastructure. The startup overhead of the jobs take a dominating amount of time for smaller crawls and are still a significant source of delay even for the larger crawls sizes. Most of this overhead comes from the delays introduced at the crawling nodes. The time to fetch the results form the resource broker is negligible, but increases linearly with the number of fetched pages, as expected.

5.3.5. Effectiveness

One of the benefits provided by the SE4SEE application is that it assigns categories to the retrieved pages. Selection of good seed pages for topic categories is important, as the crawling task is started from these pages and continued in a breadth-first manner. In this set of experiments, we try to investigate the quality of seed page selection and the behavior of classification. For this purpose, 100-page and 1000-page crawls are initiated for two different topic categories (banks and sports) and the distribution of pages into categories are investigated.

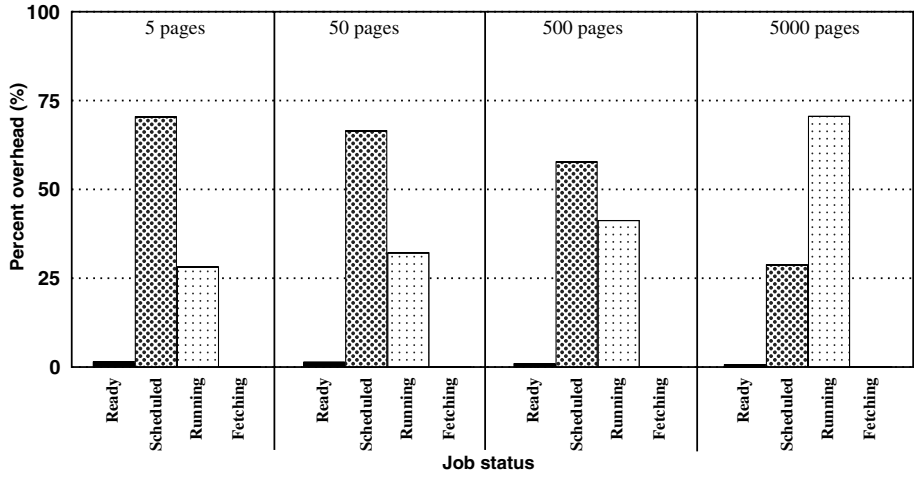

Fig. 6. The percent dissection of duration for different phases of query execution on the grid.

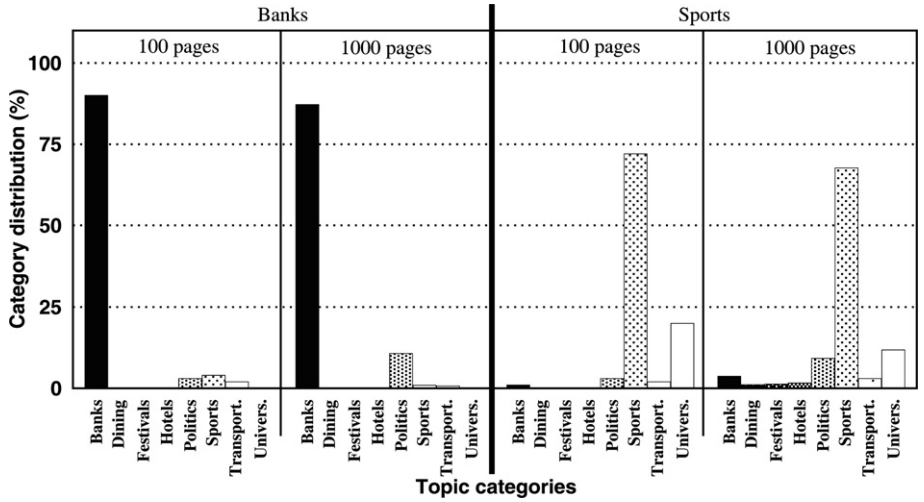

Fig. 7. Effect of seed page selection in classification of crawled pages.

Fig. 7 shows the results obtained in these experiments. As expected, as the pages are more distant in the link structure from the starting set of seed pages, the probability of classifying pages into categories other than the target category increases. This is because either the classification accuracy degrades or pages belonging to irrelevant categories are crawled. For example, in the 100-page crawl performed over the sports pages, 72.0% of the to pages are classified as sports pages, whereas the rate of relevance is 67.7% in the 1000-page crawl case. The behavior of the classification also depends on the characteristics of the topic category. For example, the bank pages are more easily distinguished (a similar behavior is also observed for the politics and universities categories) even though some portion of them are classified as politics pages. Accurately classifying sports pages seems to be harder, probably because textual features identifying sports pages overlap with the features identifying other categories.

6. Conclusion and future work

In the current version of SE4SEE, the usage of grid resources is via an inter-query-parallel approach. One other perspective could be to use an intra-query-parallel approach where each query is decomposed into

subqueries running on multiple machines. As an improvement over the current SE4SEE architecture, the future direction of the SE4SEE infrastructure is to support intra-query parallelism to make a better use of the grid resources.

One of the assets of the SE4SEE is its socio-cultural value. Grid, by its very nature is a domain of cultural integration. As a part of the grid infrastructure, SE4SEE aims to promote the establishment of the cultural foundations of the grid infrastructure and serve as a basis for socio-cultural interaction and integration. In order to achieve its goal, SE4SEE provides the grid community with tools for country- and category-specific search options. Hence, the categories selected so far are picked according to their emphasis on the cultural variations within the grid community. We hope this to be a good opportunity to enhance the inter-cultural relations in South-East European region.

Acknowledgements

This publication is based on the work performed in the framework of the FP6 project SEE-GRID, which is funded by the European Community. The SEE-GRID consortium consists of eleven contractors: ten representatives or incubators of National Grid Initiatives (NGIs) from SE European countries and CERN. The consortium contractors that represent NGIs are: GRNET (Greece), SZTAKI (Hungary), ICI (Romania), CLPP (Bulgaria), TUBITAK (Turkey), ASA (Albania), BIHARNET (Bosnia Herzegovina), UKIM a(FYR-oM), UOB (Serbia-Montenegro), RBI (Croatia).

This work is also partially supported by The Scientific and Technological Research Council of Turkey under grant EEEAG-106E069.

References

Altingovde, I. S., & Ulusoy, O. (2004). Exploiting interclass rules for focused crawling. *IEEE Intelligent Systems, 19*(6), 66–73.

Arasu, A., Cho, J., Garcia-Molina, H., & Raghavan, S. (2001). Searching the Web. *ACM Transactions on Internet Technologies, 1*(1), 2–43.

Baeza-Yates, R., Castillo, C., Marin, M., & Rodriguez, A. (2005). Crawling a country: better strategies than breadth-first for Web page ordering. In *Special interest tracks and posters of the 14th international conference on World Wide Web*. Chiba, Japan.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. New York: Addison-Wesley.

Bender, M., Michel, S., Triantafillou, P., Weikum, G., & Zimmer, C. (2005). Improving collection selection with overlap awareness in P2P search engines. In *Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 67–74). Salvador, Brazil.

Boldi, P., Codenotti, B., Santini, M., & Vigna, S. (2002). Ubicrawler: a scalable fully distributed Web crawler. In *Proceedings of AusWeb02, the eighth Australian World Wide Web conference*.

Cambazoglu, B. B., & Aykanat, C. (2005). Harbinger machine learning toolkit manual. Technical Report, BU-CE-0502, Bilkent University, Department of Computer Engineering. Ankara, Turkey.

Cambazoglu, B. B., & Aykanat, C. (2006). Performance of query processing implementations in ranking-based text retrieval systems using inverted indices. *Information Processing & Management, 42*(4), 875–898.

Cambazoglu, B. B., Turk, A., & Aykanat, C. (2004). Data-parallel Web crawling models. *Lecture Notes in Computer Science, 3280*, 801–809.

Can, F., Altingovde, I. S., & Demir, E. (2004). Efficiency and effectiveness of query processing in cluster-based retrieval. *Information Systems, 29*(8), 697–717.

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

Cho, J., & Garcia-Molina, H. (2000). The evolution of the Web and implications for an incremental crawler. In *Proceedings of the 26th international conference on very large data bases* (pp. 200–209). Cairo, Egypt.

Cho, J., & Garcia-Molina, H. (2002). Parallel Crawlers. In *Proceedings of the seventh World-Wide Web conference* (pp. 124–135).

Cho, J., Garcia-Molina, H., & Page, L. (1998). Efficient crawling through URL ordering. In *Proceedings of the 7th international World Wide Web conference* (pp. 161–172). Brisbane, Australia.

Clarke, C. L. A., Cormack, G. V., & Tudhope, E. A. (2000). Relevance ranking for one to three term queries. *Information Processing and Management, 36*(2), 291–311.

Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. L., & Gori, M. (2000). Focused crawling using context graphs. In *Proceedings of the 26th international conference on very large data bases* (pp. 527–534). Cairo, Egypt.

Foster, I., & Kesselman, C. (2003). *The grid 2: Blueprint for a new computing infrastructure*. San Francisco: Morgan Kaufman.

Han, E., Karypis, G., & Kumar, V. (2002). Text categorization using weight adjusted *k*-nearest neighbor classification. In *Proceedings of the 5th Pacific-Asia conference on knowledge discovery and data mining* (pp. 53–65).

Heydon, A., & Najork, M. (1999). Mercator: a scalable, extensible Web crawler. *World Wide Web, 2*(4), 219–229.

Kan, M.-Y. (2004). Web page categorization without the Web page. In *Proceedings of the 13th international World Wide Web conference* (pp. 262–263).

Khoussainov, R., Zuo, X., & Kushmerick, N. (2004). Grid-enabled Weka: a toolkit for machine learning on the grid. *ERCIM News 59*.

Lam, W., Ruiz, M. E., & Srinivasan, P. (1999). Automatic text categorization and its applications to text retrieval. *IEEE Transactions on Knowledge and Data Engineering, 11*(6), 865–879.

Lee, D. L., Chuang, H., & Seamons, K. (1997). Document ranking and the vector-space model. *IEEE Software, 14*(2), 67–75.

Lewis, D. D. (1992). Feature selection and feature extraction for text categorization. In *Proceedings of speech and natural language workshop* (pp. 212–217).

Lewis, D. D., & Ringuette, M. (1994). A comparison of two learning algorithms for text categorization. In *Proceedings of the third annual symposium on document analysis and information retrieval* (pp. 81–93).

Long, X., & Suel, T. (2003). Optimized query execution in large search engines. In *Proceedings of the 29th international conference on very large databases*. Berlin, Germany.

McCallum, A., & Nigam, K. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on learning for text categorization*.

Melnik, S., Raghavan, S., Yang, B., & Garcia-Molina, H. (2001). Building a distributed full-text index for the Web. *ACM Transactions on Information Systems, 19*(3), 217–241.

Moffat, A., Zobel, J., & Sacks-Davis, R. (1994). Memory efficient ranking. *Information Processing and Management, 30*(6), 733–744.

Najork, M., & Wiener, J. L. (2001). Breadth-first crawling yields high-quality pages. In *Proceedings of the 10th international conference on World Wide Web* (pp. 114–118). Hong Kong, Hong Kong.

Ng, H. T., Goh, W. B., & Low, K. L. (1997). Feature selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of the 20th international conference on research and development in information retrieval* (pp. 67–73).

Page, L., & Brin, S. (1998). The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the seventh World-Wide Web conference* (pp. 107–117).

Ribeiro-Neto, B. A., & Barbosa, R. A. (1998). Query performance for tightly coupled distributed digital libraries. In *Proceedings of the third ACM conference on digital libraries* (pp. 182–190).

Scholze, F., Haya, G., Vigen, J., & Prazak, P. (2004). Project GRACE: a grid based search tool for the global digital library. In *7th international conference on electronic theses and dissertations*. Lexington, KY.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys, 34*(1), 1–47.

Shkapenyuk, V., & Suel, T. (2002). Design and implementation of a high-performance distributed Web crawler. In *International conference on data engineering* (pp. 357–368).

Sun, A., Lim, E. P., & Ng, W. K. (2002). Web classification using support vector machine. In *Proceedings of the 4th international workshop on Web information and data management* (pp. 96–99).

Teng, S., Lu, Q., Eichstaedt, M., Ford, D., & Lehman, T. (1999). Collaborative Web crawling: information gathering/processing over Internet. In *32nd Hawaii international conference on system sciences*.

Tomasic, A., Garcia-Molina, H., & Shoens, K. (1994). Incremental updates of inverted lists for text document retrieval. In *Proceedings of the 1994 ACM SIGMOD international conference on management of data* (pp. 289–300). Minneapolis, Minnesota.

Turtle, H., & Flood, J. (1995). Query evaluation: strategies and optimizations. *Information Processing and Management, 31*(6), 831–850.

Wilkinson, R., Zobel, J., & Sacks-Davis, R. (1995). Similarity measures for short queries. In *Fourth text retrieval conference (TREC-4)* (pp. 277–285). Gaithersburg, Maryland.

Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques* (2nd ed.). San Francisco: Morgan Kaufman.

Wong, W. Y. P., & Lee, D. K. (1993). Implementations of partial document ranking using inverted files. *Information Processing and Management, 29*(5), 647–669.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval, 1*(1/2), 67–88.

Zeinalipour-Yazti, D., & Dikaiakos, M. D. (2002). Design and implementation of a distributed crawler and filtering processor. In *Proceedings of the next generation information technologies and systems* (pp. 58–74).

Zobel, J., Moffat, A., & Sacks-Davis, R. (1992). An efficient indexing technique for full-text database systems. In *Proceedings of the 18th international conference on very large databases* (pp. 352–362). Vancouver, Canada.