Web Dynamics and their Ramifications for the Development of Web Search Engines

Yiping Ke Lin Deng Wilfred Ng Dik-Lun Lee
Department of Computer Science
The Hong Kong University of Science and Technology
Email: {keyiping, ldeng, wilfred, dlee}@cs.ust.hk

Abstract

The World Wide Web has become the largest hypertext system in existence, providing an extremely rich collection of information resources. Compared with conventional information sources, the Web is highly dynamic in the following four factors: size (i.e., the growing number of Web sites and pages), Web pages (page content and page existence), hyperlink structures and users' searching needs. As the most popular and important tools for finding information on the Web, Web search engines have to face many challenges arising from the Web dynamics. This paper surveys the research issues on Web dynamics and discusses how search engines address the four factors of Web dynamics. We then briefly discuss the main issues and directions of future development of Web search engines.

1 Introduction

The World Wide Web (or simply the Web) has become the largest and most important information resource for very diversified applications and users in recent years. However, the information on the Web is prolific and connected in a distributed way, thus making it difficult for users to find the relevant information. This motivates the development of *Web search engines* (or simply *search engines*), which are vital tools for the efficient and effective retrieval of information from the Web.

The Web consists of a set of documents (usually called Web pages) and hyperlinks (or simply links), which connect Web pages across the Web. Compared with conventional hypertext systems (such as library databases), the Web is highly dynamic in its size, pages, link structures and user interests, which are all important factors in the design and development of search engines. The dynamics of the Web reflect the ever-increasing utilization of the Web for the presentation and exchange of information and the ability of an individual or organization to publish almost anything on the Web at very little cost. We now highlight the four dynamic factors that we discuss in this paper and their impacts on search engine development.

- **Dynamics of Web size.** The Web has gradually become a universal platform for presenting and exchanging information in the commercial, government, and educational sectors, which has resulted in the growth of the size of the Web at an exponential rate, as demonstrated in several studies [23, 54, 55, 43]. The rapid growth in size in terms of Web sites and Web pages poses a serious challenge for search engines, since the effectiveness relies on their coverage of large parts of Web information. In

order to gain better coverage various approaches have been introduced, such as the development of meta-search engines, special-purpose search engines and so on.

- **Dynamics of Web pages.** Another direct consequence of increasing Web usage is that at different times new Web pages come into existence, while existing pages may migrate under other URLs [26, 27, 35, 34, 38]. Some may disappear for a period of time but reappear later, or some may be deleted and cannot be found any more. Moreover, the author or the publisher of Web sites may need to modify and update the page content in order to keep the information valuable. This facet of dynamics challenges a search engine in a different way: besides there being huge amounts of information on the Web to cover, search engines must be able to keep the Web pages fresh and to re-index the information residing in local repositories. The dynamics of Web pages indirectly reflects the dynamics of the Web size, which is also related to the creation and deletion of Web pages.

- **Dynamics of Web link structures.** As the Web is essentially a hypertext system, the dynamics of the link structures [43, 18, 17, 29] is of great importance for search engines as long as link structures are commonly employed as a major component in ranking search results. In reality, the links between Web pages are being established and removed constantly. A search engine needs to address this dynamics, especially when some ranking techniques, such as the PageRank algorithm [65] used by Google [5], rely on the link information to generate search results.

- **Dynamics of Web user interests.** It is not difficult to see that different users or communities have different information needs, even when they submit the same queries to a search engine. Returning the same results to all users who ask the same query may not be satisfactory. An interesting research problem is how to provide personalized search results for various users, which has recently attracted a great deal of attention in the research communities, with the focus on ranking function optimization, relevance measure adaptation and so on [57, 58, 42, 45, 36]. In the industrial sectors, Google launched the beta version of its personalized search service [7] in 2004. This service allows users to select a set of preferred categories and consequently the search results are rearranged with the emphasis on the selected categories.

We recognize that the complex dynamics of the Web pose interesting challenges to Web searching. Search engines must devise efficient techniques and strategies to face the challenges. We cannot cover all of the activities concerning Web dynamics in this short paper. Our main objective is to stimulate interest among the research communities by presenting an overview of Web dynamics. With this purpose in mind, this paper surveys the important issues related to Web dynamics and analyzes how they impact the design and development of search engines in Sections 2 to 5. Finally, we briefly discuss the important research issues and directions for the future development of search engines, taking into consideration of our analysis of Web dynamics in Section 6.

2 Dynamics of Web Size

In this section, we review the studies on the size of the Web as well as the engine coverage and discuss how size dynamics influences the development of search engines. The Web size can be measured in terms of the number of Web sites or the number of Web pages.

The Web can be divided into two parts with respect to search engines: the *indexable Web* and the *non-indexable Web*. The indexable Web, also known as the *shallow* Web, is defined in [71] as *the part of the Web which is considered for indexing by the major engines, which excludes pages hidden behind search forms, pages with authorization requirements, etc.* By early 2005 [5], more than eight billion pages in the indexable Web were indexed by Google. In contrast, the non-indexable Web, also referred to as the *deep* Web [22], includes those pages that do not exist until they are created dynamically as the result of a specific search. In other words, the deep Web is hidden behind search interfaces, because traditional crawlers can not probe beneath the surface and reach the deep part. It is estimated in [22] that the size of the deep Web is 400 to 550 times larger than that of the indexable Web. To make the huge deep Web available to the users, a metasearch engine can be used as a middleware between the users and the search interfaces that hide the deep Web from the users (See Section 2.2.2). The metasearch engine has to characterize the contents behind the search interfaces in order to be able to route user queries to the most suitable search interfaces [44]).

2.1 Web Size and Search Engine Coverage

The Web has in fact been growing at a rapid rate, as is evidenced in Figure 1, which is extracted from the Netcraft Web Server Survey [12] that tested known host names. In the survey, the Web is measured in terms of the number of Web sites, starting from a small number of sites in August 1995 to more than 62 million in April 2005. The upper curve in Figure 1 represents the total number of Web sites, while the lower curve represents the *active Web sites* that are sufficiently important to be referenced by another site. This means that parked domains, personal Web sites not referenced anywhere, etc., are not classified as active sites.

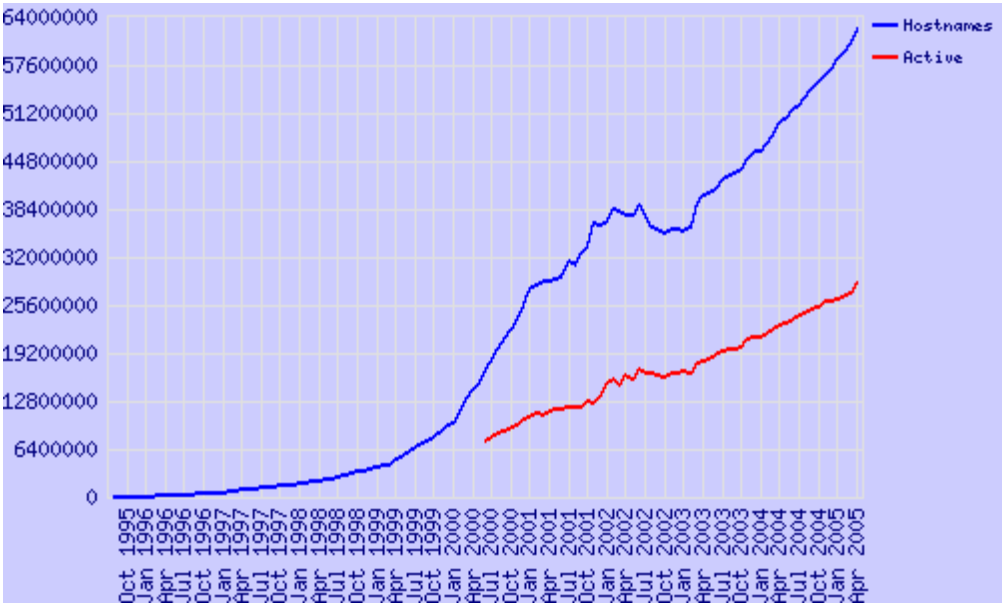

Figure 1: The Growth of the Web (Extracted from the Netcraft Web Server Survey)

Due to the fast expansion of the Web and the inherently limited resources in a search

engine, no single search engine is able to cover the entire Web. Web page coverage and the overlap of search engines was addressed by Lawrence and Giles [54]. Six major, full-text search engines were included in an extensive study, including AltaVista [1], Excite [4], HotBot [8], Infoseek [9], Lycos [10], and Northern Light [13]. Queries were issued to each search engine and then the overlaps were checked among the engines after normalizing the URLs in the search results. The size of the Web in terms of Web pages was estimated based on an analysis of the overlaps. There are several surprising but important findings in this study:

1. An estimated lower bound on the size of the Web was 320 million pages in December 1997.

2. The coverage of an individual engine was significantly limited: no single engine indexed more than one-third of the entire Web.

3. Combining the results of multiple engines can significantly increase coverage: six major search engines tested collectively covered about 60% of the Web.

As a follow-up study, Lawrence and Giles [55] repeated and extended their experiments. The number of analyzed search engines was then increased to 11 (AltaVista [1], EuroSeek [3], Excite [4], Google [5], HotBot [8], Infoseek [9], Lycos [10], Microsoft [11], Northern Light [13], Snap [14], and Yahoo [16]) and the number of queries used in the experiments was expanded from 575 to 1,050. The estimated size of the Web increased significantly from 320 million to 800 million pages, which more than doubled in fourteen months. Meanwhile, they found that no search engine indexed more than about 16% of the Web, which indicated that the search engines were increasingly falling behind in their efforts to index the Web.

Another attempt to measure Web page coverage and the overlap of search engines was carried out by Bharat and Broder [23] using random queries in November 1997. Instead of measuring directly the size and overlaps of the four tested search engines (AltaVista [1], Excite [4], Infoseek [9] and HotBot [8]), they generated random URLs from the database of a particular search engine and checked whether they were also indexed by the other engines. Their study was performed almost in the same period as Lawrence and Giles's first study. However, the size estimates differ. Bharat and Broder concluded that the Web had about 200 million pages, while Lawrence and Giles claimed it had about 320 million pages. Bharat and Broder also found that the Web page overlap between the tested search engines was very small.

In Table 1 we present the main results of the three studies mentioned above, which shows the estimated Web size measured in terms of the number of Web pages, the largest search engine coverage, and the joint coverage derived from the set of the tested engines. There are two important messages here. First, a search engine covers only a small fraction of the Web in practice, and its coverage fails to keep up with the rapid growth of the Web. Second, combining several major search engines can significantly increase the coverage of the entire Web. This motivated the development of the metasearch engine in order to combine the searching power of individual search engines. Unfortunately, there have been, to the best of our knowledge, no similar follow-up studies to access the current Web size. However, as the number of Web sites is known to grow rapidly, as shown in Figure 1, it is reasonable to estimate that the number of Web pages also increases by following the same trend.

Study	Web Pages	Largest Engine	Joint Coverage
Bharat and Broder (1997)	200 million	AltaVista (50%)	80%
Lawrence and Giles (1997)	320 million	HotBot (34%)	60%
Lawrence and Giles (1999)	800 million	Northern Light (16%)	42%

Table 1: Comparison of Three Studies on Web Size (pages) Estimation

2.2 Impact of Web Size Dynamics on Search Engines

In this section, we discuss three important strategies that search engines use to deal with Web size dynamics. We first present the scalable architectures of Google and FAST. Then we discuss the strategies developed in metasearch engines and special-purpose engines in order to tackle Web size dynamics.

2.2.1 Scalable Architecture

A powerful search engine should be capable of covering as many existing Web pages as possible. Thus, it is necessary for a search engine to have a scalable architecture to deal with the dynamic Web environment. Herein, we consider how Google and FAST, two important search engines commonly used today, designed their systems in a scalable way.

As a well-known example, Google's repository [5, 28, 21, 70] stores the full HTML texts of every Web page and employs the compression library *zlib* to store the indexes of Web pages. The compact encoding is able to scale up the maintenance costs in an effective way as the Web grows. Google also has a fast distributed crawling system in order to scale up to hundreds of millions of Web pages. A number of crawlers are run in parallel and each of them keeps roughly 300 connections open at once to retrieve Web pages at a fast enough pace. Furthermore, Google is designed with crawling, indexing, and sorting operations that are efficient enough to build an index of a substantial portion of the Web (24 million pages at 1998) in less than one week.

Another example is the FAST Search Engine architecture, which was the subject of a case study in 2002 [67]. The FAST crawler consists of a cluster of interconnected machines, each of which is responsible for retrieval, processing, and storage of a partition of Web space. All crawler machines work in a relatively independent manner and only exchange discovered hyperlinks with all the other machines in a star network. This architecture makes the engine linearly scalable with both document storage and retrieval capacity. FAST adopts a distributed architecture for the searcher and the indexer. There are two classes of nodes: *search nodes* and *dispatch nodes*. Each search node holds a portion of the index, I_i, and a searcher that searches I_i and returns search results. There are m replications of each search node to match with the query rate and balance the workload. A dispatch node is responsible for receiving queries, routing them to a set of underlying search nodes, and collecting and merging search results. They are organized in multilevels and any number of levels can be built to accommodate larger size and heavier traffic.

2.2.2 Metasearch Engines

As discussed in Section 2.1, the coverage of a search engine is limited, and the overlap of the coverage of search engines is relatively low. The simplest way of improving Web coverage is to combine the results obtained from multiple engines, which is the basic principle of a

metasearch engine.

A metasearch engine is a system that provides unified access to multiple existing search engines [61]. Unlike the crawler-based search engines, metasearch engines neither crawl the Web themselves, nor build and maintain their own indexes of documents. When a user issues a query, a metasearch engine first sends the query out to the appropriate underlying search engines in parallel, and then collects, aggregates, and often post-processes results in a unifying framework before presenting them to the users.

Metasearch engines significantly increase the coverage and scalability of the search. One of the most popular metasearch engines in use is Dogpile [2], which sends the query to a customizable list of search engines, directories and specialty search sites and then displays results from each search engine individually. An important research issue that arises from building a metaseach engine is how to mix and rank the results derived from the underlying engine components. For more details on metasearch engine design, the readers may refer to the survey in [61].

2.2.3 Special-Purpose Search Engines

While a metasearch engine seeks to cover a much larger Web space than a single search engine does, a special-purpose search engine (also known as a vertical search engine) limits the coverage in order to tailor the search results to some well-defined application domains. The *focused crawler* (also called a topical crawler or a topic-driven crawler) of a special-purpose search engine aims to selectively seek out pages that are relevant to a pre-defined set of topics, rather than to exploit all regions of the Web. This *focused crawling* technique enables a search engine to operate efficiently within a topically limited document space.

The basic procedure of running a focused crawler is as follows. The crawler starts with several seed pages, which are topic-relevant. Whenever it fetches a Web page, the unvisited URLs are extracted from that page and scored by their relevance to the topics. The crawler then picks up the URL with the highest score to crawl. Various methods can be used to score the unvisited URLs, which results in different kinds of focused crawlers. Similarity measures, such as cosine similarity used in the BestFirst crawler [59], can be simply employed. A classifier can also be trained to guide the crawler to relevant pages, such as *Focused Crawler* by Chakrabarti et al. [30] and *Context Focused Crawler* by Diligenti et al. [37]. Variations of link-based methods using topical weights can also be chosen as discussed in [24, 31]. Ontology can also be used for focused crawling [39].

3 Dynamics of Web Pages

In this section, we discuss the dynamics of Web pages, which includes the change in the existence and in content of a page. According to Baeza-Yates et al. [19], Web pages are dynamic in three ways and all of them affect the service quality of Web searching.

- **Creation.** New pages are continuously created and added into a Web site. Page creation produces new information on the Web and search engines are expected to capture this new information in a timely manner.
- **Updates.** Updates refer to content changes on pages. It is very challenging for search engines to decide how frequently to refresh their page references.

- **Deletion.** A Web page becomes non-existent if it is removed, or if all the links to it are removed. Undetected deletions are more damaging to a search engine's reputation than undetected updates, since it is an immediate frustration to users when they are directed to non-existent Web pages.

3.1 Characterizing Web Page Dynamics

There has been much work characterizing Web page dynamics. Koehler performed a six-year experiment of the weekly retrieval of a set of sampled Web pages from 1996 to 2003. The findings were successively reported in [49, 50, 51, 52]. It is interesting to find that about 33% of the sampled Web pages still existed after six years, but only 3% of the pages had not changed after a year. Over the six-year period, Koehler observed a decline in content changes on pages. In other words, as the page collection ages, it is less likely that Web authors will make modifications to the same degree as they once did. A limitation in Koehler's work is that only deletion and updates to an existing set of Web pages are studied, while the creation of new pages is not in the scope.

Another study was by Lim et al. in 2001 [56]. They developed two novel measures, namely *distance* and *clusteredness*, in order to quantify Web page dynamics. Intuitively, distance characterizes the degree of change between two versions of a Web page, while clusteredness characterizes how these changes spread out within a Web page. They performed a test on 6000 Web pages and found that most of the changes were small in distance and were clustered, which suggests that index updates in search engines based on an incremental approach can be much more efficient compared with naive methods requiring re-scanning of all Web data. Remarkably, the scale of the experiments by both Lim et al. and Koehler was small. The main difference between their work is that Lim et al. developed metrics to characterize the *degree of page change*, rather than studying the *existence or frequency of change*.

A recent and large-scale study on the Web page dynamics was by Fetterly et al. [40]. In their study, 151 million Web pages were crawled once a week for eleven weeks. A number of significant findings are stated as follows:

1. 88% of pages were still available during the final crawl, which indicates that the degree of page deletion in eleven weeks is not high.

2. By counting the number of changed "shingles", 65.2% of pages did not change at all. For those changed pages, the changes were usually located in the HTML markup or were trivial.

3. Large pages changed more often and more obviously than smaller ones.

4. Changes were highly correlated. If a Web page changed in the past, it is more likely that it will change again in the near future.

The above-mentioned results have practical implications for developing incremental Web crawlers that seek to maximize the freshness of a collection of Web pages. A shortfall of this work is that the authors only measured the deletion and update rate of pages, but did not include the rate of page creation.

3.2 The Impact of Web Page Dynamics on Search Engines

A typical way to handle the problems created by Web page dynamics in a search engine is first to develop an analytical model that characterizes changes in Web pages, and then to develop an update strategy that maximizes the freshness of pages.

3.2.1 Web Page Dynamics Modelling

Poisson Model

Most of the studies concerning page dynamics modelling assume that the Web page changes follow independent Poisson processes [26, 27, 35, 34]. That is, each Web page, P_i, is updated according to a Poisson process at an average change rate, λ_i, and the change rates are independent.

Brewington and Cybenko [26, 27] downloaded about 100,000 Web pages per day for seven months. They studied changes by recording the last-modified time stamp, the time of download, and various stylistic information. A change of a page in their study was defined as any alteration of the page. Their analysis shows that only 4% of pages were totally dynamic (changed each observation time), while 56% of pages were completely unchanged. Based on their empirical analysis, they modelled changes on the growing Web as an exponential distribution, combining the effects of page creation and updates. This is essentially a variation of the Poisson process. They further proposed a novel measure of freshness, termed (α, β)-*currency* to characterize how up-to-date a search engine is. A page is defined to be β-*current* if it is not changed between the last time it was indexed and β time units ago. A search engine is said to be (α, β)-current if the probability of a randomly chosen page in it being β-current is at least α. By specifying reasonable values for α and β, the re-indexing period for a search engine to achieve (α, β)-current can be computed. They then estimated that a search engine containing 800 million pages must refresh at least 45 million pages once every 18 days to maintain (0.95, 1 week)-currency. Their work quantifies what "current" means for search engines and estimates the rate at which search engines must re-index the Web. However, the proposed model ignored the deletion of pages and did not distinguish the degree of page changes. Another limitation is that their re-indexing scheme is based on a single revisit period for all pages.

Cho and Garcia-Molina have done a set of comprehensive work on how Web pages are updated and on crawling strategies for search engines. Their earlier work in [35] presented and compared several scheduling policies for updating local databases. They modelled Web page changes as an independent Poisson process. They also introduced the notions of *freshness* and *age* in order to measure how up-to-date the local database is. The *freshness* of the local database is the fraction of pages that are up-to-date at a given time. *Age* of a page is 0, if the local copy is up-to-date at time t, and otherwise is (t − last-modification time of the page). The *age* of the local database is the average age of pages in the database. Based on their model and two metrics, they studied the effectiveness of various updating policies, and proposed an optimal one that can improve freshness and age very significantly. The optimal scheduling policy states that all pages are visited in the same order repeatedly for each crawl cycle and at the same rate.

Later in [34], Cho and Garcia-Molina compared two design choices for Web crawlers and discussed their trade-offs. An *incremental crawler* runs continuously without any pause, updates the crawled page in-place, and revisits the page based on its changing frequency. Alternatively, a *periodic crawler* periodically updates all pages, performs shadowing (keeping the newly crawled pages in a separate space and replacing the current

collection instantaneously with this new one), and revisits pages at the same frequency, regardless of how often they change. These two designs have different advantages. The incremental crawler provides high freshness and places low peak loads on networks and servers. On the other hand, the periodic crawler is easier to implement and provides high availability of the current collection. Finally, they also proposed an architecture for the incremental crawler, which combines the best design choices they discussed earlier. Compared with the work by Brewington and Cybenko [26, 27], Cho and Garcia-Molina handled the page updates in a local database of fixed size and do not take page creation into consideration.

A Non-Poisson Model

Edwards et al. [38] proposed another refreshing strategy for an incremental Web crawler. Instead of using a Poisson process, their model makes no assumptions on the statistical behavior of Web page changes. They modelled the whole problem for controlling the crawl strategy as a nonlinear programming problem, whose objective function is the minimization of weighted sums of obsolete pages, subject to a set of constraints (such as available bandwidth and load balance). By solving this nonlinear programming problem, the outputs tell how many existing but old URLs should be re-crawled and how many newly discovered URLs should be fetched. Their use of the term "obsolete" not only covers the pages that have been crawled but are out-of-date later, but also considers "new" pages either discovered or exogenously supplied during the crawl. However, their proposed methodology is relatively expensive: the algorithm for solving the problem becomes rather complex over time and has to be periodically reset and started from scratch.

3.2.2 Experiments on Web Page Dynamics

A very recent work on Web page dynamics from a search engine perspective was carried out by Ntoulas et al. [64]. They extensively studied the creation and deletion of Web pages, as well as the frequency and degree of updates. They chose 154 "popular" Web sites from the Google Directory [6], and then downloaded all the pages for each Web site every week for 51 weeks. Their findings are summarized as follows. (1) New pages were created at the rate of 8% per week. (2) About 80% of today's Web pages will be inaccessible after one year. Thus, search engines should detect page deletions to avoid broken links in search results. (3) The vast majority of page changes were minor, as measured by the "TF.IDF Cosine Distance" and the "Word Distance," which indicates that search engines should consider the degree, not just the frequency, of change when deciding which pages to revisit.

3.2.3 Summary

Table 2 gives a summary of the issues of Web page dynamics covered in the above-mentioned studies. It presents the aspects of Web page dynamics covered and the metrics proposed to measure the freshness of search engines. The study by Ntoulas et al. is the most complete one, since it covers the issues of page creation, updates and deletion. However, they did not propose any metric. The three other studies all ignored the deletion of Web pages. Cho and Garcia-Molina [35, 34] only considered updates, without taking creation and deletion into account. There are three metrics proposed.

Some useful conclusions according to all these studies can be drawn as follows. First, the creation and deletion of pages are at a relatively high rate. Therefore, search engines should periodically fetch in new pages and detect page deletions to cope with these changes

Study	Creation	Updates	Deletion	Freshness Metric
Brewington and Cybenko [26, 27]	✓	✓	×	(α, β) - currency
Cho and Garcia-Molina [35, 34]	×	✓	×	freshness, age
Edwards et al. [38]	✓	✓	×	×
Ntoulas et al. [64]	✓	✓	✓	×

Table 2: Comparison of Web Page Dynamics Studies

on the Web. Second, most of the existing pages change either very frequently or very infrequently. Moreover, the majority of changes are "minor" ones, which however does not imply that these changes are not important. The changes are minor in the sense that search engines only need to expend little effort to reincorporate them into the index. Hence, intelligent scheduling and re-indexing are of great importance for developing an efficient search engine.

4 Dynamics of Web Link Structures

In this section we discuss the dynamics of Web link structures. We first introduce studies on Web link structure modelling and then discuss its impact on search engines.

4.1 Web Link Structure Modelling

Several studies aim to understand the model of Web linkage [43, 18, 17, 29]. In all these studies, the Web is naturally perceived as a directed graph with Web pages (or Web sites) as nodes and hyperlinks as edges.

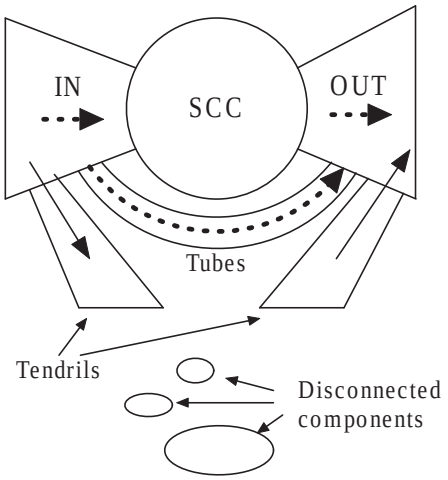

Figure 2: Modelling Web Link Structure

A detailed analysis of Web link structures was done by Broder et al. [29]. This study focuses on the connectivity of Web pages and suggests a conceptual view of a Web link structure as shown in Figure 2. As seen from this figure, the "bow-tie" like view consists of four main components and each of them has roughly the same size as detailed as follows.

1. The SCC (Strong Connected Component) represents the pages that are *strongly connected* in the Web graph. The pages in SCC can reach one another via some directed links.

2. The IN represents the pages that are able to reach the SCC but not the other way around.

3. The OUT represents the pages that are reachable from the SCC but not the other way around.

4. The Others represent the tendrils that contain pages that are reachable from portions of IN, or that can reach portions of OUT, without passage through the SCC, and also some disconnected components.

Broder et al.'s experiments also confirm that the power law holds for in-degree and out-degree distributions of Web pages, which means that the probability that a page has in-degree (or out-degree) i is proportional to $1/i^x$, for some positive $x > 1$. Furthermore, they found in their experiments that x equals 2.1 for in-degree distribution and 2.72 for out-degree distribution.

There are also a variety of random graph models and stochastic models proposed for the structure of the Web graph. We refer interested readers to [60] for a complete description of random models and [53] for stochastic models of the Web graph.

Another work related to the Web link structures and search engines was by Chakrabarti et al. [32], who studied the evolution of the Web graph influenced by the existence of search engines. They modelled the Web as an undirected and relatively static graph, where out-links are not modified after creation and pages do not disappear. They proved theoretically that the presence of search engines limit the page author's attention to a small set of "mainstream" Web pages, to which the authors are likely to link new pages. As a result, new pages become harder and harder to enter a well-connected, entrenched Web communities.

Regarding the *dynamics* of Web link structures, little work has actually been done. To our knowledge, only one study on characterizing the dynamics of the link structure was conducted by Ntoulas et al. in [64], which was introduced in Section 3.2.2. They concluded that the Web link structure is significantly more dynamic than Web pages according to experimental results. After a one-year study, they found that only 24% of the initial links were still available, which indicates a severe deletion of Web links. On the other hand, on average, 25% new links were created every week, which is much larger than 8% new pages. This result gives useful implications for search engines that their link-based ranking metrics (such as the PageRank) may need to be updated very often because a week-old ranking may not reflect the current page ranking very well.

4.2 Search Engines to Cope With Web Link Dynamics

Common search engines rely heavily on the knowledge of Web link structures in order to select relevant and important pages as returned answers to a search query. The most well-known example is Google [5], which adopts the PageRank algorithm [65] to generate effective ranked search results. PageRank makes efficient use of the link structure of the Web graph, which takes into account a page's citation importance. The essential idea is as follows. Assume a page, A, is referenced by a set of citation pages, $\{T_1 \dots T_n\}$. There is a damping parameter, d, between 0 and 1, which tunes the weight of the citation pages.

Let $C(A)$ be the number of links going out from A. The PageRank of a page A, denoted as $PR(A)$, is given by the following equation:

$$PR(A) = (1-d) + d\,(PR(T_1)/C(T_1) + \ldots + PR(T_n)/C(T_n))\,.$$

The PageRank is based on the probability of a "random surfer" on the Web with two possible actions at any given time. The surfer may either randomly choose one of the links in the current page to follow with the probability d, or get bored and jump randomly to any page on the Web with the probability $(1-d)$. $PR(A)$ can easily be calculated by using a simple iterative algorithm, which corresponds to the principal eigenvector of the normalized link matrix of the Web.

As discussed earlier, due to the dynamics of link structures, search engines should update their link-based ranking metrics regularly in order to provide an effective ranked result. However, these computations are too expensive to be performed frequently. To tackle this problem, Chien et al. [33] proposed an efficient algorithm that incrementally computes approximations to PageRank when links are added or removed from the Web. The basic idea of their algorithm is as follows. Given a set of link changes, a small subgraph of the Web graph is constructed. This subgraph contains a small neighborhood in the vicinity of these changes (i.e., the area of the Web graph affected by the changes) and the rest of the Web graph is modelled as a single node. Then, a version of PageRank is computed on this small subgraph and the results are suitably transferred to the original graph. In a range of experiments either on real Web data or on synthetic models, their algorithm performed well in speed and quality. They also demonstrated the efficiency of the algorithm by running experiments on various types of link modifications (e.g., small or large numbers of changes, random or correlated link additions).

Another commonly used algorithm to rank the search results is Kleinberg's HITS [48]. Useful pages on a topic are categorized into two types: "authorities" which contain information on the topic, and "hubs" which provide links to authorities. Each page is originally assigned an authority weight and a hub weight of one. The vector of authority (hub) weights is then normalized to unit length. Starting with a set of results from a keyword search, the HITS algorithm first constructs the connectivity matrix M, and then multiplies M^T (the transpose of M) by the vector of hub weights to update the authority weights. Similarly, multiplying M by the vector of authority weights updates the hub weights. The update process is repeated until the weights remain fixed. Essentially, the vectors of authority weights and hub weights are the principal eigenvectors of M^TM and MM^T respectively.

In addition to keep the rankings up-to-date, the search engines should also be able to provide stable rankings under small perturbations to the Web link structure, such as the additions or deletions of several links. Ng et al. [62, 63] used the techniques from matrix perturbation theory and Markov chain theory to analyze the stability of the ranks derived by HITS and PageRank. They found that the rankings by PageRank tended to be more stable than those by HITS. As for HITS, as long as the eigengap of M^TM, which is defined as the difference between the largest and the second largest eigenvalues, is large, HITS is insensitive to small perturbations. The condition for the perturbed PageRank scores to remain stable is that the perturbed Web pages did not have high overall PageRank scores. Based on the analysis, they further designed two stable ranking algorithms, namely randomized HITS and subspace HITS. Randomized HITS merges the notions of hubs and authorities from HITS with a stabilizing reset mechanism from PageRank. Subspace HITS

combines multiple eigenvectors from HITS to produce aggregate authority scores. Both two algorithms were proved to be stable empirically on real Web query data.

5 Dynamics of Web User Interests

Web searching is basically an interaction between the Web and the users. Thus, the *dynamics of Web user interests* plays an important role in search engine development. The dynamics of Web user interests refers to the differences in the information needs arising from different Web users. For example, with the simple query "apple", some users may be interested in Web pages about "apple" as a computer brand, while other users may want information related to "apple" as a fruit. Furthermore, even the same user may have different information needs for the same query at different times. To address the dynamics of Web user interests problem, personalized Web searching techniques have been developed to help search engines deliver adapted results to individual users according to their personal information interests. Some common search engines such as Google have launched personalized search services (c.f. the beta version of personalized facilities in [7] in 2004). We review the state-of-the-art personalized Web searching techniques in this section.

5.1 Personalized Web searching

We first identify three important features in a Web searching process on which personalization can be carried out. We then introduce a classification of personalized Web searching and review some representative techniques in each category.

- *Relevance measure.* As the primary goal of Web searching is to find relevant Web pages for a user query, the first feature is to evaluate the *relevance* between the query and the Web pages. To determine relevance, several retrieval models have been proposed, for example, the Boolean model [69], the vector space model [20], and the probabilistic model [68].

- *Importance measure.* As the size of the Web is huge and a simple query may retrieve a large number of results, most statistical retrieval models are not selective enough to limit the number of query results to a manageable size. In order to guide the *ranking process* of search results, the second feature is to evaluate the *importance* of a Web page. Link analysis techniques are then proposed to compute the importance scores of Web pages. For example, the commonly used ones are PageRank [42] and HITS algorithms [48].

- *Ranking function.* In a Web searching process, a final scalar is used for ranking the search results. A ranking function is needed in a search engine that combines a number of relevance and importance scores and then generates a final ranking score before listing the results. Therefore, the third feature is to employ an effective ranking function that evaluates a page in a Web searching process.

Based on these three features of Web searching, personalization techniques can be developed. Accordingly, the personalized searching techniques can be categorized into *content-based personalization, link-based personalization* and *function-based personalization.*

First, content-based personalization deals with the "relevance" measure of Web pages and the user query. We call this kind of techniques as "content-based" due to the fact that the user's query is modified to adapt the *content* of search results delivered to specific users. In order to manage the users' interests, content-based personalization constructs user profiles and stores users' interests derived from users' search histories.

Second, link-based personalization carries out personalization on link analysis techniques. Traditional link analysis techniques, like the PageRank algorithm, only compute scores that reflect a "democratic" *importance* and have no preferences for any particular pages. However, in reality, a user may have a set of preferred pages that he or she considers more interesting. The link-based personalized searching techniques redefine the *importance* of Web pages according to different users' preferences or different queries. For example, a user may wish to specify his bookmarks as a set of preferred pages, so that any retrieved pages that are important with respect to his bookmarked pages would be ranked higher than other non-bookmarked pages.

Third, the ranking function of a search engine combines different relevance and importance scores into a uniform ranking score. Function-based personalization derives optimized weight values for individual users, according to the user's preferences obtained from the clickthrough logs. As the ranking function serves different users with specific adapted weights, personalized ranking results can be delivered to the users.

Table 3 summarizes the classification of the personalization to deal with dynamics of Web user interests, which we discuss in subsequent sections.

Category	Personalized Features	Techniques or System Names
Content-based	Relevance Measure	Liu et al. [57, 58], Outride [66] Sugiyama et al. [72]
Link-based	Importance Measure	Topic-sensitive PageRank [42], Personalized PageRank [45]
Function-based	Ranking Function	SpyNB with RSVM [36]

Table 3: Three Categories of Search Engine Personalization

5.2 Content-Based Personalization

The underlying idea of content-based techniques is to import personalization into the computation of relevance scores between the user query and Web pages [58, 66]. Different users may have different views about what kind of pages are relevant to a simple query. For example, the Web pages about "red apple" should be considered irrelevant for a user who wants information about "apple computers". The content-based techniques typically associate a category "computer" with the user's query for searching, in order to obtain only "computer" information and to discard "fruit" pages. In order to store the users' search interests, content-based techniques usually construct user profiles that consist of a set of terms extracted from search histories. The terms in a user profile representing the user's search intention are then used as a context for augmenting the user's queries in the search process.

One representative content-based personalization technique is reported in [58], in which Liu et al. proposed a technique to map a user query to a set of categories, which represent the user's search intention. This set of categories can serve as a context to disambiguate the user's query words. A user profile and a general profile are learned from the user's search history and an Open Directory Project (ODP) [15] hierarchy respectively. These

two profiles are then combined for associating a user query with a set of related categories in the searching process.

Another representative work is the Outride System [66], which adopts a contextual computing approach to assist in understanding the information consumption patterns of each user. The primary way for the system to personalize search results is *query augmentation*. For example, if a user is looking at a series of pages on car information and searches for "contour", the system may augment the query by adding the term "car" or "Ford" to provide the user with results about the Ford Contour car. In particular, the system first constructs user models by weighting the top 1,000 categories in ODP [15], according to the user's browsing history or a set of favorite links. Later, when a user has submitted a query, the similarity between the query term and the user model is computed. If the query is on a topic that the user has previously seen, the system reinforces the query with similar terms, or suggests results from prior searches. Otherwise, the system does not augment the query; instead, it helps the user by providing a diverse set of results that the user may be interested in.

Some other studies in this category focus on designing personalized search systems based on *user profiling*. User profiling is the process of collecting information about the characteristics, preferences, and activities of a Web site's visitors. This can be accomplished by explicit collection of user profile data through the use of online registration forms, questionnaires, and the like. The methods that are applied for implicit collection of user profile data vary from the use of cookies or similar technologies to the analysis of the users' navigational behavior that can be performed using Web usage mining techniques. In particular, Sugiyama et al. [72] proposed a new system to adapt Web searches based on the user's profile construction without any user effort. Their system first records all the navigation history of each user, then assigns different weights to each term based on the individual user. The weight is tuned with the time the user accessed. Then, the system applies KNN (k-nearest neighbor) classification to find the similar users based on their preference for each term to do collaborative filtering.

	User Profile	General Profile	Based on ODP	User Efforts
Liu et al. [58, 57]	√	√	√	Not required
Outride [66]	√		√	Need some
Sugiyama et al. [72]	√			Not required

Table 4: Content-Based Personalization

Table 4 summaries the work on content-based personalization as we discussed so far in this section. We compare the methods on the following four aspects: building user profiles, building general profiles, presence of the keywords in the profile extracted from ODP, and collecting user interests requiring users' efforts. Liu et al.'s work is the most comprehensive one, since it constructs both user profiles and general profiles to model a user's interests, and the construction of profiles is based on the ODP hierarchy. Moreover, the way they capture the user's interests does not require any user effort. The Outride system builds user profiles based on the ODP hierarchy as well, but they do not build general profiles. Regarding the efforts from users, they may require the users to import a set of favorite links to build the initial profile. Comparatively, Sugiyama et al.'s work constructs user profiles without relying on efforts from users. However, there is no general profile and the keywords contained in the user profile are not as standard and informative as ODP keywords.

5.3 Link-Based Personalization

Link-based personalization adapts the importance measure of Web pages. In the original PageRank algorithm [65], a PageRank vector is computed using the link structure of the entire Web as already discussed in Section 4.2. These generic PageRank scores are absolute and independent of any queries or users. However, the *importance* about Web pages may be variable with respect to different queries or users.

For example, suppose a user searches for sports news via a search engine and the search engine finds that the homepages of CNN[1] and ESPN[2] both contain sports news. The search engine then decides which one should rank higher. As CNN is a general news center but ESPN is a news portal specializing in sports, ESPN is superior to CNN regarding sports news and thus the search engine is expected to rank ESPN's homepage higher than CNN's. However, the generic PageRank computation may give CNN's homepage a higher PageRank score, since CNN is more general and thus may have more important in-links than those in ESPN. Thus, a personalization strategy is to adapt the importance scores to some specific topics. The essence of the link-based personalization technique is to enable the search engine to evaluate Web pages, which we call *biased importance*.

One approach of such adaptation is reported in [42], in which Haveliwala proposed a *topic-sensitive PageRank* algorithm for extending original PageRank computation to have biases for particular topics. By making PageRank topic-sensitive, heavily linked pages are not ranked highly for queries on which they have no particular authority. The topic-sensitive PageRank algorithm can be summarized in the following two steps:

1. During off-line processing, 16 topic-sensitive PageRank values are pre-computed for each Web page, and each value is biased using URLs from a top-level category from the Open Directory Project (ODP) [15]. A topic-sensitive PageRank vector of a Web page is given by, $R = (r_1, \ldots, r_{16})$, where each r_i represents the importance of this Web page with respect to one of the 16 topics.

2. At query time, a similarity vector, $S = (s_1, \ldots, s_{16})$, of the query is computed. Then, a composite PageRank score is calculated by the vector dot product $R \cdot S$. The composite PageRank score reflects the importance of a Web page with respect to the query's topic.

Another approach is reported in [45], in which a *personalized PageRank* algorithm was proposed to compute "personalized" PageRank scores, with biases to a user-specified set of initially-interesting pages, for example via the user's bookmarks. As a result of the personalized PageRank algorithm, any query results that are important with respect to the user's bookmarked pages would be ranked higher.

Personalized PageRank extends the original random surfer model by introducing a set of user preferred pages, P, such as one's bookmarks. We are then able to account for the preferred pages in P using the random surfer model as follows: at each step, a surfer jumps back to a random page in P, with a teleportation probability c, and with probability $(1 - c)$ continues forth along a hyperlink. Intuitively, the limit distribution of surfers in this model would favor pages in P, pages linked to some pages in P, and so on. The distribution that a random surfer visits any page is represented as a *Personalized PageRank Vector* (PPV), which has biases towards P.

[1] http://www.cnn.com
[2] http://www.espn.com

We present a brief comparison of the work on original and personalized PageRank algorithms in Table 5.

PageRank	Biased Importance
Original PageRank [65]	No bias to queries and users
Topic-sensitive PageRank [42]	Biased to the topics of the query
Personalized PageRank [45]	Biased to a given set of user preferred pages

Table 5: Link-Based Personalization

For HITS algorithm, it begins by using the user's query to build a neighborhood graph over which linkage based computations of authority and hub weights are made. Therefore, unlike original PageRank, HITS has some bias to the topics of user's query. In this sense, its personalized degree is similar to the topic-sensitive PageRank. Various modifications of HITS algorithms which combine content and link analysis [24] can also be used for personalized Web search.

5.4 Function-Based Personalization

Unlike the link-based and content-based approaches, function-based personalization does not intervene in the computation of generic relevance and importance scores of Web pages. Instead, the function-based approach introduces personalization to the ranking functions, which are used to combine all relevance and importance scores into a single scalar.

The formation of a ranking function depends on the notion of the feature vector of Web pages. In general, a search engine can represent a page by using n relevance features given by the vector $(R_1, \ldots, R_n)$. For example, R_1 can be the cosine relevance of the query and the body text, R_2 can be the cosine relevance of the query and the title, and so on. Similarly, there can be m importance features in a page given by $(I_1, \ldots, I_m)$. For example, I_1 can be the PageRank score [42], I_2 can be the HITS score [48], and so on. Combining these features together, a Web page is characterized by a *feature vector*, $F = (R_1, \ldots, R_n, I_1, \ldots, I_m)$. Then, we define a weight vector, $W = (w_1, \ldots, w_{n+m})$ for tuning F. The final ranking score, S, is computed as a vector dot product of F and W. The weight vector, W, adapts its components, w_k (for $1 \leq k \leq n+m$) towards a user, U. The weight vector, W, which is regarded as the personalized ranking function, can be tuned from a user's search history based on the techniques that will be described later on.

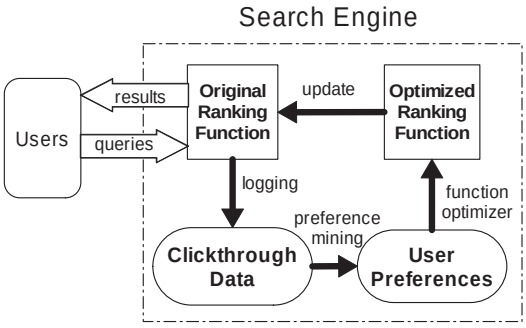

Figure 3: Function-Based Personalization Via Search Engine Adaptation

Figure 3 illustrates the workflow of a function-based personalized search engine. In

the figure, we show that clickthrough data are a sort of log record, in which, for each query, the search results presented to the user and the links clicked by the user are kept. The advantage of using clickthrough data is that the collection of such data requires no user interruption, since it is an implicit feedback for the search results. Function-based personalization methods generally comprise two main processes. The first one is *preference mining*, which discovers the user's preferences about search results from clickthrough data. The second process is *ranking function optimization*, which learns an optimized ranking function by using the preferences identified by the first step as the input data.

Now, we elaborate on preference mining algorithm and ranking function optimization, which are two main processes in function-based personalization.

5.4.1 The Preference Mining Algorithm

The goal of the preference mining algorithm is to discover users' preferences from clickthrough data such that the ranking function optimizer is able to decide which search results are what the user wants. The notion of preferences [47] can be expressed as follows.

Given two retrieved links, l_i and l_j, for a given query, q, the pairwise preference, $l_i <_q l_j$, means that the user prefers l_j to l_i for the query q.

Using the notion of preferences, Joachims first proposed "Joachims' algorithm" [46] for mining preferences from clickthrough data. The underlying idea of Joachims' algorithm is as follows: a user does not click on a link, l_i, but clicks on a lower link, l_j ($j > i$). Assuming that a user scans search results from top to bottom, the user must have observed l_i and decided to skip it, before he or she clicked on l_j. Thus, the preferences can be identified as: $l_i <_q l_j$.

Links	The list of search results
l_1	**Apple** ***http://www.apple.com/***
l_2	Apple - QuickTime - Download *http://www.apple.com/quicktime/download/*
l_3	Apple - Fruit *http://www.hort.purdue.edu/ext/senior/fruits/apple1.htm*
l_4	**Apple .Mac Welcome** ***http://www.mac.com/***
l_5	www.apple-history.com *http://www.apple-history.com/*
l_6	MacCentral: Apple Macintosh News *http://www.macworld.com/news/*
l_7	Adams County Nursery, apple trees *http://www.acnursery.com/apples.htm*
l_8	**Apple - Support** ***http://www.info.apple.com/***
l_9	AppleInsider *http://www.appleinsider.com/*
l_{10}	ROSE APPLE Fruit Facts *http://www.crfg.org/pubs/ff/roseapple.html*

Figure 4: The Clickthrough Data for the Query "apple". (Clicked Links are bold.)

Preferences containing l_1	Preferences containing l_4	Preferences containing l_8
Empty Set	$l_2 <_q l_4$	$l_2 <_q l_8$
	$l_3 <_q l_4$	$l_3 <_q l_8$
		$l_5 <_q l_8$
		$l_6 <_q l_8$
		$l_7 <_q l_8$

Table 6: Pairwise Preferences (Joachims' algorithm) From the Clickthrough in Figure 4

Figure 4 illustrates an example of clickthrough data for the submitted query "apple". In this example, three links, l_1, l_4 and l_8, are in bold, which means they are clicked by the user. The preferences identified by Joachims' algorithm are shown in Table 6.

One drawback of Joachims' algorithm is that it may over-penalize the highly-ranked links. A modified version of Joachims' algorithm, termed the "mJoachims' algorithm", was recently proposed in [36], which adds to the standard Joachims' algorithm the following criterion: Suppose l_k is an unclicked link between two consecutive clicked links, l_i and l_j ($i < k < j$). In addition to the preferences of the standard Joachims' algorithm, the extra preference, $l_k <_q l_i$, should be added to remedy the over-penalizing effect, since the user must have observed link l_k ($k < j$) before clicking on l_j and decided not to click on l_k.

There are two major issues for preference mining, one is the lack of appropriate negative examples for a given topic/user, the other is the sparseness and incompleteness of positive examples. For common clickthroughs, we could only expect the user to click on a few positive links. As such, the unlabelled data may still consist of a significant number of positive results. While recent research on document classification has begun to address this ambiguity [41], work has not been done in the context of search engine adaptation. SpyNB (Spy Naïve Bayes) is a machine learning technique based on a more advanced preference mining algorithm, which was proposed in [36]. Unlike Joachims' algorithm and mJoachims' algorithm, SpyNB algorithm only assumes that the user's preferences of search results are actually the preferences on topics. The essential idea of SpyNB algorithm is to first classify clicked links as *positive examples*, P, and those not clicked as *unlabeled data*, U. By analyzing the texts of search results, the *Actual Negative set*, $AN \subset U$, can then be identified via a variant of the Naïve Bayes learning process. The preferences can be identified as follows: $l_j <_q l_i, \forall l_i \in P, l_j \in AN$. In order to obtain more accurate *actual negatives*, they introduced a voting procedure to make full use of all potential *spies* that are some selected positive examples mixing in U.

5.4.2 Ranking Function Optimization

After the preferences are obtained from a preference mining algorithm, a ranking function optimizer can take as input the preference data and adapt the ranking function to hold as many input preferences as possible.

Joachims [46] first proposed an effective *ranking SVM* algorithm which essentially applies an SVM approach, which is a classical machine learning technique, to optimize the ranking function. Recently, Tan et al. [73] proposed a *RSCF algorithm* to extend the ranking SVM to a co-training framework [25] for tackling the lack of clickthrough data problem.

The Ranking SVM in the Co-training Framework (RSCF) algorithm is an enhancement

of the RSVM [46]. RSCF incorporates into RSVM a co-training framework [25] to make the learning process efficient, even the amount of training data is relatively small and sparse. RSCF analyzes the clickthrough data extracted from the log files and then categorizes the data as the labelled data set, which contains the search items that have been scanned by users, and the unlabelled data set, which contains the data items that have not yet been scanned. It then augments the labelled data with the unlabelled data to obtain a larger data set for training the rankers as shown in Figure 5.

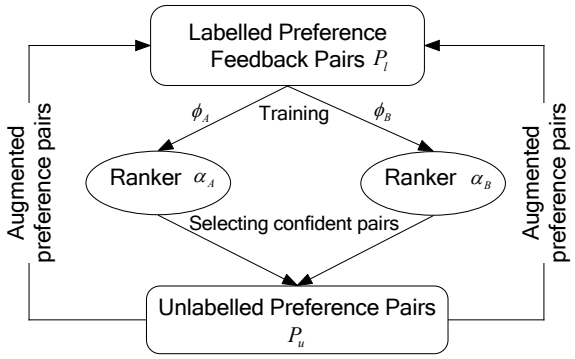

Figure 5: The Underlying Idea of the RSCF Algorithm

In the figure, two rankers, α_A and α_B, are incrementally built over these two feature subvectors, ϕ_A and ϕ_B. Both rankers use the RSVM algorithm to learn weight vectors. Each ranker is initialized with a few labelled preference pairs extracted from the click-through data. In each iteration of co-training, each ranker chooses several preference pairs from the unlabelled set, P_u, and adds them to the labelled set, P_l. The chosen pairs are those with the highest ranking confidence as given by the underlying rankers. Then, each ranker is rebuilt from the augmented labelled set. In the next iteration, the new rankers are used to rank the unlabelled preference pairs again. The ranking preference pairs process and the building rankers process repeat until all unlabelled preference pairs are labelled or a terminating criterion is satisfied.

6 Conclusions

In this survey, we review and study four dimensions of Web dynamics, namely, size, pages, link structures and user interests. We discuss their impacts on the design and development of search engines. On one hand, the dynamic nature of the Web has posed serious challenges to search engines, which aim to cover as large a portion of the Web as possible, provide the latest version of Web pages, rank search results better, and give personalized search results to users. On the other hand, to cope with the highly dynamic Web, search engines should have scalable architectures, intelligent scheduling strategies, efficient update algorithms for ranking metrics, personalized Web searching technologies, and so on.

We find that there are lots of interesting and challenging issues that need further study. The first one is the follow-up study on some "fact-exploring" research. For example, the existing work concerning Web size dynamics is not updated enough to uncover the evolution of the current state of the growing Web. Although a number of studies on link structures have been conducted, little attention has been paid to the dynamics of the link

structures.

A new challenging problem concerns the dynamics of data representation on the Web. As XML has been evolving rapidly as the de-facto standard for describing data on the Web, the ability for a search engine to understand the structure and semantics of XML data is a key feature for next-generation engines. The growing deep Web is becoming another challenging problem, since more local search engines are provided in more and more Web sites. The techniques to make the deep Web searching as effective and efficient as the current indexable Web searching is a big challenge.

Regarding various personalization categories for tackling the dynamics of user interests, existing personalized search techniques mostly assume that user's interests are persistent. It is thereby feasible to learn users' current interests from their search histories. However, user interests often change. The problem of how to model a user's changing interests over time opens the door to a new set of challenges and opportunities. As for function-based personalization, the problem of how to make more sophisticated ranking functions personalizable is an interesting area deserving future study.

Acknowledgement. We would like to express our sincere thanks to the editors and the reviewers, who gave very insightful and encouraging comments.

References

[1] AltaVista. *http://www.altavista.com.*

[2] Dogpile. *http://www.dogpile.com.*

[3] EuroSeek. *http://www.euroseek.com.*

[4] Excite. *http://www.excite.com.*

[5] Google. *http://www.google.com.*

[6] Google Directory. *http://dir.google.com.*

[7] Google Personalized Web Search. *http://labs.google.com/personalized.*

[8] HotBot. *http://www.hotbot.com.*

[9] Infoseek. *http://www.infoseek.com.*

[10] Lycos. *http://www.lycos.com.*

[11] MSN Search. *http://search.msn.com.*

[12] Netcraft Web Server Survey. *http://news.netcraft.com/archives/web_server_survey.html.*

[13] Northern Light. *http://www.nlsearch.com.*

[14] Snap. *http://www.snap.com.*

[15] The Open Directory Project: Web directory for over 2.5 million URLs. *http://www.dmoz.org.*

[16] Yahoo Directory. *http://dir.yahoo.com.*

[17] L.A. Adamic. The small world Web. In *Proc. of Third European Conference of Research and Advanced Technology for Digital Libraries*, 1999.

[18] R. Albert, H. Jeong, and A. Barabási. Diameter of the World Wide Web. *Nature*, 401(6749):130–131, September 1999.

[19] R. Baeza-Yates, C. Castillo, and F. Saint-Jean. Web dynamics, structure, and page quality. In M. Levene and A. Poulovassilis, editors, *Web Dynamics: Adapting to Change in Content, Size, Topology and Use*, chapter 5. 2004.

[20] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison-wesley-Longman, Harlow, UK, 1999.

[21] L.A. Barroso, J. Dean, and U. Holzle. Web search for a planet: The google cluster architecture. *IEEE Micro*, pages 22–28, March–April 2003.

[22] M.K. Bergman. The deep Web: Surfacing hidden value. *White paper, Bright Planet*, July 2000.

[23] K. Bharat and A. Broder. A technique for measuring the relative size and overlap of public Web search engines. In *Proc. of the 7th International WWW Conference*, pages 379–388, April 1998.

[24] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proc. of SIGIR*, pages 104–111, 1998.

[25] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. of COLT*, pages 92–100, 1998.

[26] B.E. Brewington and G. Cybenko. How dynamic is the Web. In *Proc. of the 9th International WWW Conference*, 2000.

[27] B.E. Brewington and G. Cybenko. Keeping up with the changing Web. *IEEE Computer*, 33(5):52–58, 2000.

[28] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[29] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the Web: Experiments and models. In *Proc. of the 9th International WWW Conference*, May 2000.

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

[31] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource list compilation by analyzing hyperlink structure and associated text. In *Proc. of the 7th International WWW Conference*, 1998.

[32] S. Chakrabarti, A. Frieze, and J. Vera. The influence of search engines on preferential attachment. In *Proc. of SODA*, pages 293–300, 2005.

[33] S. Chien, C. Dwork, R. Kumar, and D. Sivakumar. Towards exploiting link evolution. In *Workshop on Algorithms and Models for the Web Graph*, November 2002.

[34] J. Cho and H. Garcia-Molina. The evolution of the Web and implications for an incremental crawler. In *Proc. of VLDB*, 2000.

[35] J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. In *Proc. of SIGMOD*, pages 117–128, 2000.

[36] L. Deng, X. Chai, Q. Tan, W. Ng, and D.L. Lee. Spying out real user preferences for metasearch engine adaptation. In *Proc. of WebKDD*, pages 71–82, 2004.

[37] M. Diligenti, F. Coetzee, S. Lawrence, C.L. Giles, and M. Gori. Focused crawling using context graphs. In *Proc. of VLDB*, pages 527–534, September 2000.

[38] J. Edwards, K. S. McCurley, and J. A. Tomlin. An adaptive model for optimizing performance of an incremental Web crawler. In *Proc. of the 10th International WWW Conference*, pages 106–113, 2001.

[39] M. Ehrig and A. Maedche. Ontology-focused crawling of Web documents. In *Proc. of the 2003 ACM Symposium on Applied Computing*, 2003.

[40] D. Fetterly, M. Manasse, M. Najork, and J. Wiener. A large-scale study of the evolution of Web pages. In *Proc. of the 12th International WWW Conference*, 2003.

[41] G. Fung, J. Yu, H. Lu, and P.S. Yu. Text classification without labeled negative documents. In *Proc. of ICDE*, pages 594–605, 2005.

[42] T.H. Haveliwala. Topic-Sensitive PageRank. In *Proc. of the 11th International WWW Conference*, pages 517–526, 2002.

[43] B.A. Huberman and L.A. Adamic. Evolutionary dynamics of the World Wide Web. Technical report, Xerox Palo Alto Research Center, February 1999.

[44] P.G. Ipeirotis and L. Gravano. When one sample is not enough: Improving text database selection using shrinkage. In *Proc. of SIGMOD*, 2004.

[45] G. Jeh and J. Widom. Scaling personalized Web search. In *Proc. of the 12th International WWW Conference*, pages 271–279, 2003.

[46] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. of KDD*, pages 133–142, 2002.

[47] W. Kieβling. Foundations of preferences in database systems. In *Proc. of VLDB*, pages 311–322, 2002.

[48] J.M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. of SODA*, pages 668–677, 1998.

[49] W. Koehler. An analysis of Web page and Web site constancy and permanence. *Journal of the American Society for Information Science*, 50(2):162–180, 1999.

[50] W. Koehler. Digital libraries and World Wide Web sites and page persistence. *Information Research*, 4(4), 1999.

[51] W. Koehler. Web page change and persistence – a four-year longitudinal study. *Journal of the American Society for Information Science and Technology*, 53(2):162–171, 2002.

[52] W. Koehler. A longitudinal study of Web pages continued: a consideration of document persistence. *Information Research*, 9(2), January 2004.

[53] R. Kumar, P. Ragoavan, S. Rajagopalan, and D. Sivakumar. Stochastic models for the Web graph. *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2000.

[54] S. Lawrence and C.L. Giles. Searching the World Wide Web. *Science*, 280(5360):98–100, 1998.

[55] S. Lawrence and C.L. Giles. Accessibility of information on the Web. *Nature*, 400(6740):107–109, 1999.

[56] L. Lim, M. Wang, S. Padmanabhan, J. S. Vitter, and R. Agarwal. Characterizing Web document change. In *Proc. of the 2nd International Conference on Advances in Web-Age Information Management*, pages 133–144, 2001.

[57] F. Liu, C. Yu, and W. Meng. Personalize Web search by mapping user queries to categories. In *Proc. of CIKM*, pages 558–565, 2002.

[58] F. Liu, C. Yu, and W. Meng. Personalized Web search for improving retrieval effectiveness. *TKDE*, 16:28–40, 2004.

[59] F. Menczer, G. Pant, P. Srinivasan, and M.E. Ruiz. Evaluating topic-driven Web crawlers. In *Proc. of SIGIR*, pages 241–249, 2001.

[60] J.F. Mendes. Theory of random networks and their role in communications networks. In M. Levene and A. Poulovassilis, editors, *Web Dynamics: Adapting to Change in Content, Size, Topology and Use*, chapter 4. 2004.

[61] W. Meng, C. Yu, and K. Liu. Building efficient and effective metasearch engines. *ACM Computing Surveys*, 34(1), 2002.

[62] A. Y. Ng, A. X. Zheng, and M. I. Jordan. Link analysis, eigenvectors and stability. In *Proc. of IJCAI*, pages 903–910, 2001.

[63] A. Y. Ng, A. X. Zheng, and M. I. Jordan. Stable algorithms for link analysis. In *Proc. of SIGIR*, 2001.

[64] A. Ntoulas, J. Cho, and C. Olston. What's new on the Web? The evolution of the Web from a search engine perspective. In *Proc. of the 13th International WWW Conference*, pages 1–12, May 2004.

[65] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the Web. Technical report, Computer Science Department, Stanford University, 1999.

[66] J. Pitkow, H. Schinze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel. Personalized search. *Commun. ACM*, 45(9):50–55, 2002.

[67] K.M. Risvik and R. Michelsen. Search engines and Web dynamics. *Computer Networks*, 39:289–302, June 2002.

[68] S.E. Robertson and K.S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Sciences*, 27(3), 1976.

[69] G. Salton, E.A. Fox, and H. Wu. Extended boolean information retrieval. *Commun. ACM*, 26(11):1022–1036, 1983.

[70] S. Schikc. A glimpse into google's hardware solutions. *Internet Marketing News*, 2003.

[71] E.W. Selberg. *Towards Comprehensive Web Search*. PhD thesis, University of Washington, June 1999.

[72] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive Web search based on user profile constructed without any effort from users. In *Proc. of the 13th International WWW Conference*, pages 675–684, 2004.

[73] Q. Tan, X. Chai, W. Ng, and D.L. Lee. Applying co-training to clickthrough data for search engine adaptation. In *Proc. of DASFAA*, pages 519–532, 2004.