A Collaborative Decentralized Approach to Web Search

Athanasios Papagelis and Christos Zaroliagis, *Member, IEEE*

Abstract—Most explanations of the user behavior while interacting with the web are based on a top-down approach, where the entire Web, viewed as a vast collection of pages and interconnection links, is used to predict how the users interact with it. A prominent example of this approach is the random-surfer model, the core ingredient behind Google's PageRank. This model exploits the linking structure of the Web to estimate the percentage of web surfers viewing any given page. Contrary to the top-down approach, a bottom-up approach starts from the user and incrementally builds the dynamics of the web as the result of the users' interaction with it. The second approach has not being widely investigated, although there are numerous advantages over the top-down approach regarding (at least) personalization and decentralization of the required infrastructure for web tools. In this paper, we propose a bottom-up approach to study the web dynamics based on web-related data browsed, collected, tagged, and semi-organized by end users. Our approach has been materialized into a hybrid bottom-up search engine that produces search results based solely on user provided web-related data and their sharing among users. We conduct an extensive experimental study to demonstrate the qualitative and quantitative characteristics of user generated web-related data, their strength, and weaknesses as well as to compare the search results of our bottom-up search engine with those of a traditional one. Our study shows that a bottom-up search engine starts from a core consisting of the most interesting part of the Web (according to user opinions) and incrementally (and measurably) improves its ranking, coverage, and accuracy. Finally, we discuss how our approach can be integrated with PageRank, resulting in a new page ranking algorithm that can uniquely combine link analysis with users' preferences.

Index Terms—Collaborative search engine, personalization, person-oriented system, user information space.

I. INTRODUCTION

A FAIR portion of current research regarding the web is focused on tools that can tailor the web experience to end-user needs. Web personalization, for example, can range from tools that make the presentation of a site more pleasing to complex algorithms that predict the needs of the users and provide relevant results under various contexts. More often than not, such tools have to overcome significant difficulties taking into account the size and the heterogeneity of the web, as well as the number of its users. Many current web personalization algorithms use user-specific data to bias an initial model or algorithm toward the end-user needs. In this paper, we refer to this school of thought for web personalization as the *top-down approach*.

Examples of this approach concern personalization attempts regarding web search. In [6], an alteration of the PageRank algorithm [7] is discussed that takes into account user-specific characteristics and bias the page ordering using the users' preferences. However, this approach does not scale well with the number of users. The approach is further exploited in [18], where several PageRanks are computed for a given number of distinct search topics. Their system can compute the relevance of a query to the given topics, and subsequently it can select the most appropriate PageRank computation to better order relevant pages. Another approach is used in [29], where the PageRank computation takes into account the content of the pages and the query terms the web surfer is looking for. In [22], a decomposition of PageRank to basic components is suggested that may be able to scale the different PageRank computations to a bigger number of topics or even distinct users.

The aforementioned approaches have the inherent drawback that they begin with a pre-existing model or strategy for describing the web dynamics, mainly through its topology, that does not take into account specific user needs. They try to alter or tune the initial model by merging it, a posteriori, with user-related data. In other words, user-specific data is not a cornerstone ingredient of their initial web model making personalization a daunting and expensive task.

In this paper, we depart from the above lines of research and propose a *bottom-up approach* for representing the interesting part of the web that does not rely on the exploitation of the web topology. In particular, our approach starts from the user and views the web as an aggregated collection of URLs, browsed, collected, tagged, and semi-organized by individuals. Our approach is based on the observation that users act as small crawlers seeking information on the web through various media. They often store, tag, and organize useful URLs inside their information spaces or to social bookmarking sites. These collections of web-related data can be combined, without loosing their discrete nature, to produce a view of the web from the user perspective. Understanding the underlying mechanism of this process can lead to a better understanding of the actual user needs and a new generation of more democratic and personalized web services.

As a vehicle to discuss several of the characteristics of the proposed bottom-up approach, we propose a hybrid bottom-up

search engine. This search engine has been materialized into *Searchius*, a system that produces search results by collecting and analyzing user-generated URL collections and their structures. Conceptually, Searchius can be positioned somewhere between search engines and web catalogs. A main difference between Searchius and traditional search engines is the source and the format of the collected data as well as the methods used to collect and update it. A main difference between Searchius and current peer-to-peer systems is that the latter are mainly used to exchange files between users, while Searchius facilitates the sharing of information.

A bottom-up search engine, like Searchius, has several unique characteristics compared to a traditional search engine. Specifically.

- It can be expanded and updated in an ad-hoc manner through asynchronous connections initiated by end users avoiding the costs of web crawling.
- It can potentially overcome shortcomings of algorithms based on link analysis (e.g., web islands), where information unreferenced by other sites is not being indexed.
- It is not capital intensive since it concentrates only on a small portion of the data that typical search engines collect and analyze; however, the collected data is of the highest interest for users, and thus, from a searcher perspective, the difference goes unnoticed for a broad number of searches.
- It allows for simple and efficient implementations of several personalization algorithms. To order pages by importance, Searchius uses an aggregation function based on the preference for pages by different users, thus avoiding the expensive iterative procedure of algorithms like PageRank.
- It can be used in a decentralized environment by groups of people that want an easy, informal way to share knowledge. For example, imagine a university department having its own members contribute to a local version of Searchius that can be later queried for department specific information.
- It can be used to segment the URL space to relative subspaces. This property can be exploited to provide efficient solutions to a range of additional applications, including the construction of web catalogs and finding related pages.

To further elaborate and support our approach, we also make the following contributions.

1) We demonstrate via an extensive experimental study the qualitative and quantitative characteristics of user generated URL collections and provide functions with high correlated coefficients to estimate the way people collect and organize URLs and the rates that distinct URLs and keywords grow.
2) We discuss how our approach can be integrated with PageRank, providing an alternative version of PageRank that combines two authorities: the link analysis and the users' preference. This approach can be used to improve a global page ordering or produce a page ordering that is biased toward the needs of specific groups of users.
3) We compare our bottom-up search engine with a traditional search engine using the existence of overlap in search results as an indication of effectiveness. The experiments show consistent overlaps for a broad range of topics even for our medium-sized database. We also compare our page ranking method with PageRank and pin-point similarities and differences.
4) We discuss personalization extensions for our hybrid search engine and demonstrate how the proposed bottom-up approach can provide personalized results in an efficient and straightforward manner.

The rest of the paper is organized as follows. Section II describes the different types of user generated web-related data. It also strengthens our bottom-up approach by pinpointing common cases in which people opt to collect and organize URLs and elaborates on the limitations of this process regarding the amount and quality of collected data. Section III presents the concept and the characteristics of our bottom-up search engine as well as, its scope, problems and solutions, and its structural differences with current state-of-the-art search engines. In Section IV, we describe how our analysis can be extended to use the user browsing history as well. Section V describes how the collected data can be efficiently used to provide personalized results for specific groups of users. Section VI describes additional applications of the collected data. Section VII describes an extension of the PageRank algorithm that combines link analysis and the users' preference. Section VIII extensively describes experimental results using real world data and comments on the way that users collect and organize URLs. In Section IX, we summarize our contribution and discuss open problems and future directions. Finally, in Section X, we discuss related papers and web sites that share common ideas with our work. Preliminary portions of this work appeared in [27], [28].

II. CONCEPT VALIDATION

User generated web-related data can be divided into two broad categories: explicitly and implicitly generated data.

Explicitly generated data refer to *personal bookmarks* that are stored inside the end-user information environment and *social bookmarks* that are shared with other users through sites like Delicious (www.delicius.com) or Stumbleupon (www.stumbleupon.com). Personal and social bookmarks have a lot in common. Both are generated, organized, and updated from end users, and both include temporal metadata for a URL (when it was generated, when it was updated). Personal bookmarks use a tree-like structure to describe and organize URLs, while social bookmarks use keyword tags to describe them. The folder names on the tree structure can be considered an efficient way to attach tags to multiple URLs at once; this simple observation renders the two methods almost equal regarding their semantic characteristics. An important difference between personal and social bookmarks is the underlying usage process. Personal bookmarks serve a personal need and include URLs that the user wants to revisit in a frequent fashion. On the other hand, social bookmarks are mostly about sharing an interesting link that the contributing end user may, actually, never revisit. In this respect, personal bookmarks are a more trustworthy metric of the everyday user activity.

	Data Volume	Temporal Characteristics	Attached Semantics	User Representation	Privacy Concerns
Social bookmarks	+	++	+++	+	+
Personal bookmarks	++	++	+++	++	++
Browsing history	+++	+++	++	+++	+++

Fig. 1. Types and characteristics of user generated web-related data.

Implicitly generated data refer to the URL history that is created while users browse the web. This source of information differs significantly from bookmarks and creates bigger privacy concerns. However, it is an extremely rich source of information that can be used to accurately describe the web dynamics.

The table in Fig. 1 presents the differences between personal bookmarks, social bookmarks, and browsing history over five dimensions (as it stems from our observation). Recent research works [20], [35] have tried to evaluate the effectiveness of social bookmarks as a complementary data source for search engines with mixed results. On the one hand, the discoveries pinpoint that social bookmarks have unique characteristics that can be valuable to search engines. On the other hand, a common criticism is that their volume is not yet big enough to make a difference on search results. It comes as a natural extension for these works to supplement them with an analysis of more types of user-generated data, describe the architecture and effectiveness of a user-centric search engine, and offer a unification method between the ranking algorithm of current search engines and user generated data.

To validate our concept, we have opted throughout the paper to concentrate on the qualitative and quantitative analysis of explicitly generated data collections (in particular, collections of personal bookmarks). This is not loss of generality, since in Section IV, we discuss how our analysis can be extended to include implicitly generated data collections as well. Personal bookmarks and browsing history offer much bigger data volumes than social bookmarks, but we need to overcome practical problems like the collection process and the deeper privacy concerns.

A. Do Users Collect and Organize URLs?

One might argue that people do not collect URLs or that they do not organize them in any reasonable manner. It is common for many users not to bookmark a site they can locate easily through a search engine. For example, there is little use to bookmark the home page of the *Nature* journal since it is easy to find it through a search engine. However, there is a number of everyday scenarios in which locating (and most importantly relocating) a site is not search engine oriented and bookmarking it is perhaps the best way to access it (or at least not loosing it). Some examples follow.

- Search engines are commonly used as a start-point for traversing the web. Upon following hyperlinks, it is common for someone to find a web site he likes. Often, it is really difficult to use a search engine to find this site again mainly because it can be located on the third or fourth

results' page for most correlated queries. In this case, it worths bookmarking it.
- Web 2.0 has sparkled the creation of many tools that produce URLs in a non standardized way. For example StumbleUpon (www.stumbleupon.com), a FireFox plugin, fetches URLs based on users' interests in a random manner. If one loses a URL, it is really difficult to relocate it; he needs to bookmark it. Other examples of sites that fall in this category are Digg (digg.com) and YouTube (youtube.com), which produce URLs about news or entertainment topics.
- One-click access to URLs that are integrated to current browsers, as always visible Bookmark Toolbars, is more convenient than the search-locate-click process of a search engine. Today, it is common for users to organize inside Bookmark Toolbars their everyday URLs.
- Word of mouth, books, magazines, and newspapers, TV and radio, instant messengers, IRC chat rooms, and email can also produce URLs that would be difficult to rediscover through search engines.

The URL organization does not need to be strict as well. People can tag their URLs with names they find useful. Communities like the one at Delicious have shown that this vague categorization based on nonstructured descriptions can be used effectively in practice. The signal loss in traditional categorization schemes comes from compressing things into a restricted number of categories. With open categorization schemes, when there is signal loss, it comes from people not having any commonality in talking about things. The loss is from the multiplicity of points of view, rather than from compression around a single point of view. However, in a world where enough points of view are likely to provide some commonality, the aggregate signal loss falls with scale in bottom-up systems, while it grows with scale in top-down systems. The solution to this sort of signal loss is growth. Well-managed organizational schemes get worse with scale because the costs of supporting such schemes at large volumes are prohibitive. Bottom-up approaches, by contrast, get better with scale. With a multiplicity of points of view, the question isn't "where this URL belong?", but rather "what percentage of the community believe that this URL belong to this category?" From this perspective, trying to find one single way to organize things is a mistake.

Finally, as discussed in [1] and we experimentally demonstrate in Section VIII, there are specific patterns with good log-log fits (that indicate power law distributions) that precisely describe the way people collect, organize, and express preferences for URLs. There is no concrete reason to believe that these patterns will change in the foreseeable future.

B. Can We Collect Enough Data?

The amount of data that we can collect through implicit or explicit URL collections is another potential problem. Without much data, we will not be able to produce or improve general purpose applications.

Current successful web-based user communities have several millions of users. For example, eMule (www.emule-project.net), a peer-to-peer file exchange program, reports at least 2 million connected users at any given time. Delicious reported 2 million registered users on March 2007, which was two times more than six months before. Stumbleupon reported 7.8 million users on July 2009. Facebook (www.facebook.com) reported 68 million users on January 2009. Web email service is a good example of what levels the user base of such systems can reach. Recently, Yahoo (yahoo.com) reported 163 Million registered users for its services, while Hotmail (hotmail.com) reported more than 200 Million active users.

Taking into account the radically increasing user base of community/collaborative systems, it should be no long until we see sites with several hundreds of millions of users. Specifically in Searchius, we would need around 320 million unique users to build a database of 2 billion distinct URLs (according to the function of the URLs growth presented in Table V of Section VIII). A bottom-up search engine with that amount of data would have a recall rate similar to current generic search engines. Although this amount of users is very high, it is still nonprohibitive under the context of the above discussion.

Even if we accept that the current volume of personal and social bookmarks is not enough to produce practical applications, our analysis can be extended to implicit collections of URLs (see Section IV). Implicit collections of URLs constitute a dependable source of user-specific data since people will always produce implicit streams of URLs when interacting with the web. Our analysis is expanded later to include this source of data. Note though that the user browsing history raises much greater privacy issues than bookmark collections.

The question of *incentives* to contribute to a social service and overcome the *cold start* problem is a difficult one, and its answer partly relies on social studies. In practice, however, we can find several services, like Wikipedia (www.wikipedia.com), that offer no real incentives to users to contribute and still have turn to major successes. As another example, Delicious, a conceptually close fit to Searchius, offers no real incentives to users except the usage of the service itself. Still, in Section IX, we briefly discuss *reward* schemes that can bootstrap this process.

III. SEARCHIUS: CONCEPTS AND SYSTEM DESCRIPTION

Under a bottom-up search engine setting, users act as small-scale intelligent crawlers, seeking information on the web using various media (search engines, catalogs, word-of-mouth, hyperlinks, direct URL typing, etc.). They tend to store and organize important-for-them URLs in tree-like structures, referred to as *bookmark collections*, where the folder names act as semantic tags over the collected URLs.[1] This method of organizing data helps people to recall collected pages faster, but can also be used as a kind of semantic tagging over the URLs.

Note that the path to the URL can be perceived as different ways to communicate the URL itself. This information constitutes part of the user's *personal information space*, and it is indicative of his interests. More on personal information spaces and on how people tend to collect and organize URLs can be found in [1] and in Section VIII.

Several techniques have explored the utility of personal information spaces for the construction of personal profiles and the consequent exploitation in favor of the user [3], [4], [9], [24], [33]. Searchius takes another direction, combining the personal information spaces of many users to a concrete source of information. This combined source of information can be used as a global search engine.

Fig. 2 shows the proposed view of the web versus the traditional method of representing the web as a graph. The left part of Fig. 2 models the web as a graph. Nodes represent web pages, while edges represent directed links between two pages. Each link can carry a keyword list that is used as a descriptor for the destination page. The right part of Fig. 2 models the web through a combined user perspective. Keywords are used to group pages and organize them to tree-like structures. Keywords at different levels can have a father-child relationship carrying implicit semantic meaning. Each tree structure is a vague description of the whole web, allowing one to construct a better model of it in an incremental manner. Intuitively, in order to produce a meaningful representation of the whole web, one has to combine as many users' tree structures as possible.

Searchius is a collaborative search engine; it stores, analyzes, ranks, and serves URLs that were suggested directly from end users. When a user adds a URL to his personal information space, he actually gives a positive *vote* to this URL: web users tend to store pages that they find interesting and would like to revisit. If a URL is globally important, then we can assume that it can be found on many different users' personal information spaces.

A. UsersRank Concept

Under the above context, the ranking of pages in Searchius is based on how many *different* users have voted for a specific page p. The total number of such votes is called the *UsersRank of page* p. This differs from the method followed by PageRank, which is based on the web graph to decide about the importance of a page.

In our implementation, Searchius relies entirely on users and does not need the iterative process of PageRank to produce a quality ranking.[2] At its simplest version, a simple aggregation of different users that have stored the URL is enough. Each

---

[1]As an example of the semantic tagging via the bookmarks structure, consider the path: `Sys−>Tennis−>RolandGarros` (http://www.rolandgarros.com/).

[2]There is still some controversy—based on experimental studies—whether the iterative process of Google is needed in order to find the authority score of a URL. For example, it is suggested in [2] that even simple inbound link counting can perform equally well.
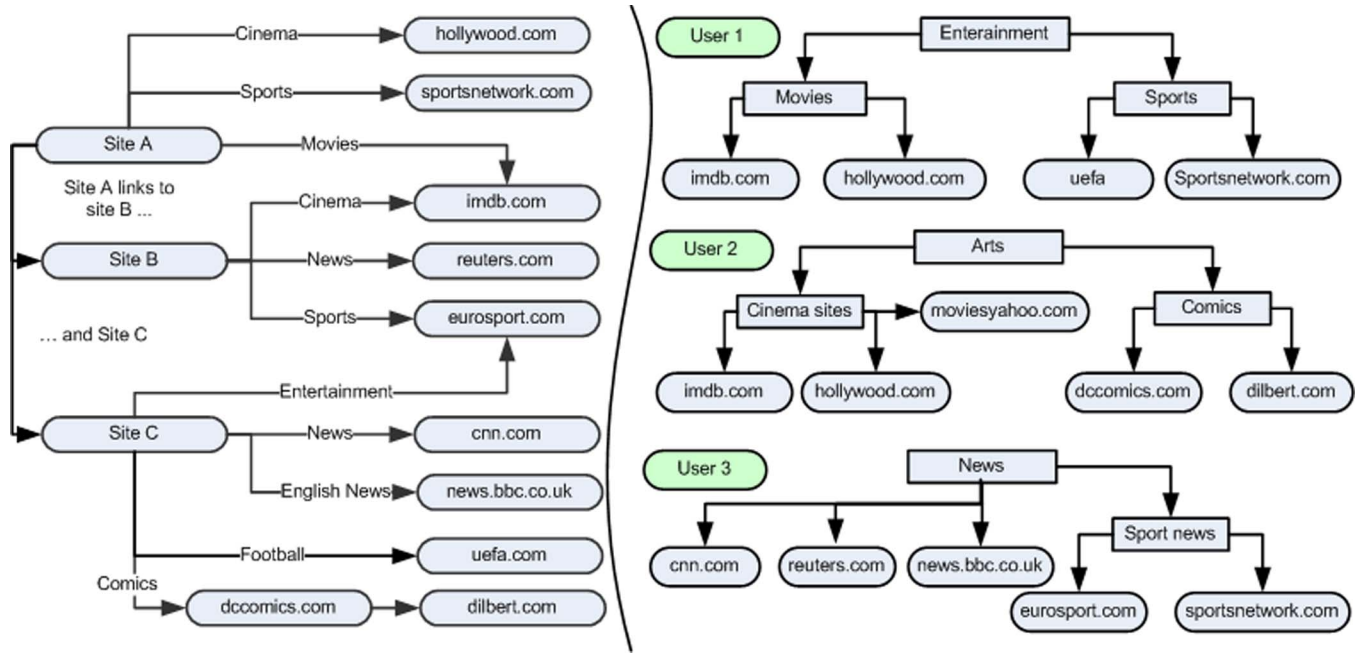
Fig. 2.   Left: the web as a graph. Right: the web as a collection of organized, user-specific URL sets.

user's URL collection is treated equally, but it is possible to change the importance of a specific user bookmark and thus the aggregated value of the corresponding URLs (e.g., we can give lower weights to older URL sets). We call this method of ranking pages *UsersRank*, since it is controlled by the collective preference of the users.

Formally, let us assume that our database consists of a set U of $x = |U|$ users, each having a URL set U_i, where $i \in \{1, 2, \ldots, x\}$. Also, let $B = \{B_1, B_2, \ldots, B_b\}$ be the set of all different URLs from all users. Then, each U_i can be represented by a vector of size b with each element $U_i(j)$ being 1 or 0 depending on whether this specific user has the URL B_j or not.

The *UsersRank of a particular page* B_j in the database is the total number of users from U that have it in their URL set. The UsersRank is represented by a b-vector R_U, whose j-th element $R_U(j)$ equals the sum of all URL values of all users at the jth position of their URL set, that is

$$R_U(j) = \sum_{k=1}^{x} U_k(j).$$

We shall see in Section VII how the UsersRank can be integrated into current search engines and help them provide more unbiased search results. In addition to factors such as keyword matches and link popularity, UsersRank integrates into the page ordering equation the users feedback, potentially improving the quality of searching results.

Fig. 3 shows the computation of UsersRank. A column represents the URLs of a user, while a row represents the UsersRank of a URL. Note that the value of a cell can initially be 0 or 1. Through time, we diminish the value of current URL sets to half producing values like 0.5, 0.25, and so on. This small extension does not alter the UsersRank computation and bias results toward fresh information.
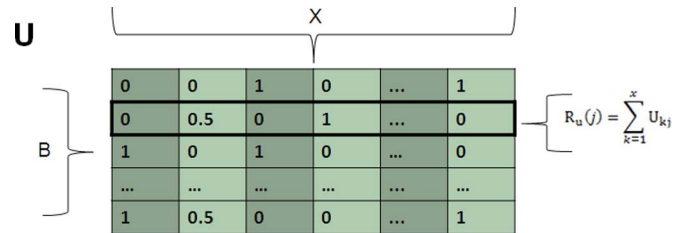

Fig. 3.   UsersRank computation.

B. System Architecture

A high level overview of Searchius architecture is presented in Fig. 4. At its current form, Searchius is a peer-to-peer system from a functional and not from an architectural point of view since it stores data on a centralized infrastructure. However, all data is produced and consumed by end users.[3]

Data is collected via the *Communicator*, a small executable that can be freely downloaded and installed to the end user's system. This program collects the URLs, their titles, and their semantic tags as given by the users by working on the background of the user's system. The user has full control on what part of his data is allowed to be sent to the systems infrastructure. Collected data together with a unique, neutral, and system specific key (*User-Key*) are transferred and stored to the *Data Repository*. The submitted data can include metadata about the user like his city, country, language, and interests. These data can be later used to personalize the search results according to individuals that share similar attributes. Whenever the system finds a batch of data that has an already existing User-Key, it replaces the old data with the new one. Each new data transfer is initially tagged as *unsettled*.

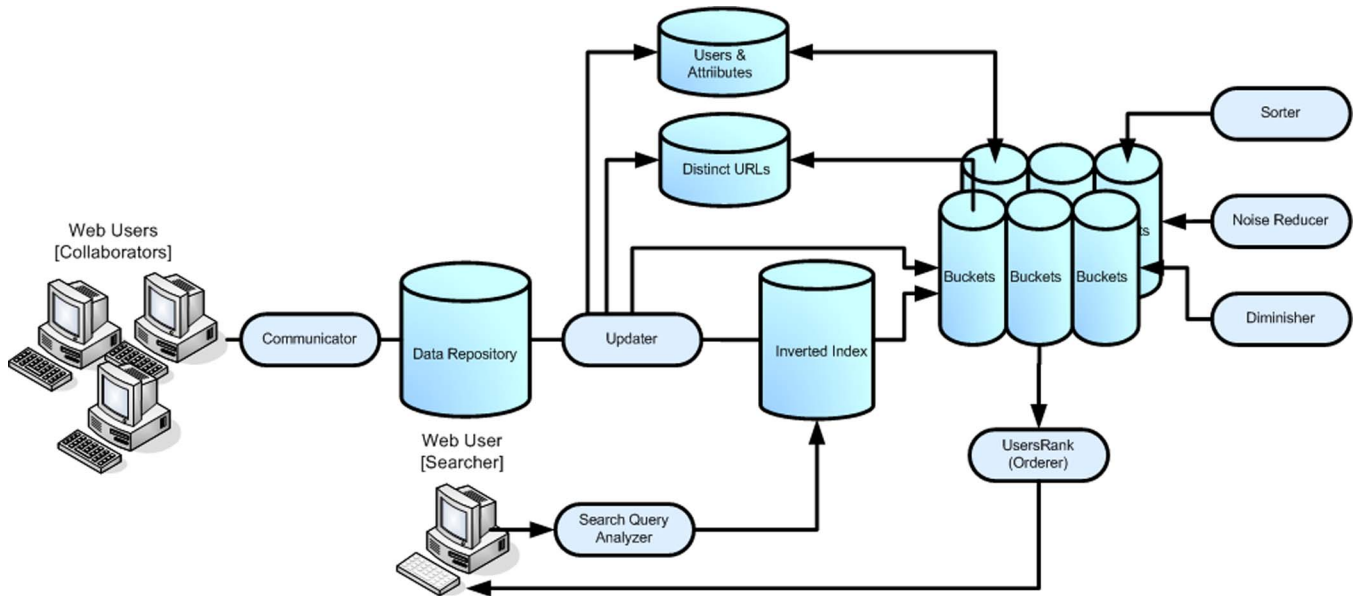[3]The interested reader can find a working prototype of Searchius at: http://Searchius.ceid.upatras.gr.

Fig. 4.   High level architecture of Searchius.

The *Updater* checks at specific time intervals for unsettled data collections. For each such collection, the Updater performs a number of database updating tasks.

1) It adds the user key and his metadata on the *Users-Attributes* table or replaces already existing records for that specific user.
2) It checks for not yet discovered URLs and adds them to the *Distinct URLs* table.
3) It parses the URLs, their titles, and semantic tags and adds possible new keywords to the *Inverted Index* table. The Inverted Index table maps each word with a unique number, a word-ID, that is used as an identifier for all URLs that are related to this word.
4) It adds all distinct connections between word-IDs, URLs, and users to the *Bucket* tables. In Buckets, we actually store a detailed decomposition of how words, URLs, and users interact with each other. For each word, URL, and user connection, a *value* is also stored that is later used from the *UsersRank* module to order pages. Currently, Searchius gives higher values to keywords that are found to URL titles (semantic tags). The system uses a number of different Buckets to keep them reasonable in size.

The *Sorter*, the *Noise Reducer*, and the *Diminisher* are triggered at specific time intervals to improve the response and quality of our database. The *Sorter* sorts Bucket entries based on their word identifier. This speed ups the process of summing up the total values of URLs for a specific keyword. The *Noise Reducer* uses heuristic filters to remove from the database low quality URLs. The *Diminisher* reduces the values of Bucket rows and is the main method for keeping the database updated since we never directly delete data from Buckets. Diminisher is triggered once each month and reduces the value of all buckets entries to half. This way Searchius results are biased toward newer data.

When a user poses a query, the *Search Query Analyzer* parses and analyzes it. For each nontrivial word, the Analyzer finds the relative bucket and word-ID from the inverted index table. The *UsersRank* module produces an ordered list of the relative URLs based on their aggregated values. When we work with multiple word queries, we internally get a result set for each word inside the query. The result sets are then merged by multiplying their original single word score and by giving a *disproportional benefit* to results that are presented in multiple set pairs. This way single word matches are not excluded from the results but better matches combined with a good UsersRank are more probable to occur at the first spots.

It is possible to augment the search query with restrictions about the participants. For example, we may ask to reduce the scope of the users to those that are based in Europe. In this case, the system finds the users that have the required country attribute from the Users-Attributes table and reduces the Users-Rank scope only to the bucket entries that belong to users from that set. This is a computationally efficient method to bias the search results toward the interests of specific groups of users.

The proposed system can be scaled in a straightforward manner. First of all, the amount of data we work with is only a small fraction of the data of mainstream search engines, since we do not collect the actual page content. Furthermore, the Communicators are distributed to end-user systems and can be programmed to communicate with different Data Repository servers. The Updater can incrementally update the system's infrastructure whenever there are idle CPU cycles. The same stands for the sorter, noise reducer, and diminisher that can be activated at ad-hoc intervals and can target only small subsets of buckets. A crucial point is to scale the number of Buckets as system grows and more keywords are discovered and also introduce a maximum level on the amount of data we store for distinct word-IDs. The later is important to keep the UsersRank computation reasonable in time, since it is actualized using a value aggregation process that is produced in real time.

C. Searchius and State-of-the-Art Search Engines

In this section, we briefly compare the mechanics of current search engines to our approach with respect to data collection, quality of search results, and updating of data.

1) Data Collection: Current search engines collect data using crawlers. Crawlers scan the web by traversing pages and return discovered data to the search engine's infrastructure. Crawling can be expensive regarding time, processing power and bandwidth.

In our approach, we use a radically different method to collect data; we provide the end user with a tool that allows him to easily communicate a part, or all, of his data to our infrastructure. This allows us to build a reasonable sized database using limited infrastructure. Also, it allows for efficient real-time search engine updating, unlike current search engines which, typically, separates the crawling, database updating, and the page rank computation. However, this also means that we rely on users in order to improve and update the database and that we generally collect only a small portion of web data.

2) Quality of Search: Current search engines store and analyze (at least) page body text, the title tag, the URL, and anchors text (the text that describes hyperlinks). In order to provide search results, they compute an *Information Retrieval* (IR) score out of page specific factors and the anchor text of inbound (incoming) links to a page. In this way, the pages related to a query are determined. To combine the IR score with the page ranking algorithm, the two values are combined through a process called rank merging [17].

Searchius uses a similar approach. The IR score of a page is calculated based on the page title, URL, and semantic tagging given by users. This semantic tagging is the exact analog to anchor links and can produce diverse descriptions of pages. Then, we employ a simple rank merging technique to produce the final ordering; we multiply the results with the UsersRank of each page in the collection to produce the final ordering. We do not store the body text at all, although an extension of this system could use a simple fetching technique to bring and store the actual pages content based on the URLs' database (such an extension is out of the scope of this paper).

Let us elaborate on the the fact that our collected URLs are only a portion of the Web's URLs. We argue that for most generic queries this has minimum effect from the user point of view. Our URL collection is anticipated to have a better average quality than any URL collection produced by a current search engine, since our method of collecting URLs (through lists organized by users) is, by default, better (with respect to quality) than any mechanized crawling technique.[4] The fact that for a given query the returned page matches are less is transparent to the user, since a user typically visits only a small portion of pages. As long as the first few pages include the *most* important results, the user is satisfied. This anticipated behavior has indeed been verified by our experimental study (see Section VIII).

However, the fact that we store less URLs and we do not store page content can produce problems with *multiterm* search queries. For such queries, there is only a small number of page matches and a bigger database does make a difference. Multiterm queries do not interfere with the quality of returned pages directly; they are mostly used to determine potential matches through well established IR techniques. To this end, any limitation that has to do with multiterm queries does not affect the UsersRank of a page.

The main issue with multiterm queries stems from the fact that Searchius does not store the actual content of pages. Thus, the multiterm matching is based on a shallower text repository and potentially can miss good matches based on page body. Note that even the Google founders emphasize the importance of web-page title data over page content on finding related pages [7], by stating that: "*For most popular subjects, a simple text matching search that is restricted to web-page titles performs admirably when PageRank prioritizes the results.*"

Summarizing, at its present form, Searchius is pretty good at answering generic search queries with a small number of search terms on any topic as long as the answer is important for many people. Otherwise, when we need to answer complex and multiterm search queries it is preferable (and more practical) to use UsersRank as a complementary ingredient of existing search engines. This limitation stems from the *usage context* (rather than the generic nature of Searchius) and is mainly a problem of *depth* rather than *breadth* of search.

3) Updating of Data: To update their database, current search engines have to constantly crawl the web. In our context, it is not possible to use the same technique because updating can be initiated only by end users. Updating is crucial, because web pages tend to become obsolete over time—for example, they may be time specific (e.g., a music event) or they may refer to a site that no longer exists. To cope with such problems, we introduce an *aging* procedure for the collected pages. At predefined time intervals, we reduce the *value* of each page in the Searchius database. Since we use a simple value aggregation procedure to find the most important pages for a search query, the effect of old URL collections to searching results diminishes over time giving more room to fresh data.

D. Searchius and Peer-to-Peer Systems

Pure peer-to-peer systems maintain a catalog based on the shared files of *only* the currently connected users. This is obligatory in their context, because a big amount of data that cannot or should not be stored in a central server have to be transferred between end users. In our context, it is not necessary to transfer any data of that size, and thus it is efficient to have a persistent database of all users who have submitted their URL collections over time. When a user submits his data, we keep a user neutral identification string for him. This string is unique for his system so we can easily map the user with subsequent submissions. This prevents, to a reasonable extend, multiple submissions from the same user. In case of a resubmission, we simply replace the data that the user has already submitted with the new data.

---

[4]We show in Section VIII-D that indeed the average PageRank of pages stored by users is "bigger" than the average PageRank of pages found on the web.

IV. UsersRank and Implicit Collections of URLs

Extending Searchius to work with implicit collections of URLs (browsing history) is straightforward. In order to maintain a fresh view on what a user prefers, we need to use an *observation window* that consists of an arbitrary number N of URLs. These are the last N pages that a user has visited. We need to slightly modify our initial UsersRank definition as discussed in Section III in order to make it compatible with browsing history. Again, we assume that our database consists of a set U of $x = |U|$ users, each having a URL multiset (the users browsing history) U_i', where $i \in \{1, 2, \dots, x\}$. Also, let $B = \{B_1, B_2, \dots, B_b\}$ be the set of all different URLs from all users. Finally, let $F_i(j)$ be the number of occurrences of page B_j in U_i'. Then, the preference $U_i'(j)$ of user i for any page B_j can be formulated as

$$U_i'(j) = F_i(j)/N.$$

Each U_i' can be represented by a vector of size b with each element $U_i'(j)$ being the preference of user i over page B_j, and the UsersRank of a particular page B_j is defined similarly as

$$R_{U'}(j) = \sum_{k=1}^{x} U_k'(j).$$

The quality of page ordering will be benefited from browsing history for three reasons. First, the pages we visit determine with high accuracy our current interests. If we can monitor the users browsing history, then we can always produce a fresh view of how the web dynamics evolve. Second, the frequency of visiting specific pages gives a much better indication of our relative preference for specific sites. Third, the order in which we visit sites can also be used and exploited in many ways. For example, frequently subsequent sites in a browsing history can be linked as belonging to the same theme.

Note, however, that implicit collections of URLs lack semantic tags that group similar sites under the same context. Also, they cannot be used directly for other applications that rely on the exploitation of the URLs structure like the related-URLs discovery application as presented in Section VI.

Collecting the user browsing history can be achieved by tools installed on a user's computer, which monitor his surfing behavior. The user may decide to switch on or off this tool for privacy issues. Current browser add-ons, like Google's or Yahoo's toolbars, can be easily extended to include this functionality (assuming that they do not include it already). This extension is out of the scope of this work.

V. Personalization

The fact that we use collections of data that represent discrete users' preferences opens a wide spectrum of possible applications regarding personalization. In this section, we present an approach to personalization on the *group level*, that is, we provide a simple way to personalize the results of the search engine based on groups of users that share one or more similar characteristics.

In our analysis, we assume that each user's URL set is accompanied with a set of attributes that are implicit or explicit. For example, when a user submits his data, we can implicitly decide (with high probability of success) about his country and city from his IP. The user can provide, at a second step, additional attributes about his interests, his language, and so on.

Assume that there is a total of z attributes, and let A_t be the value-set of attribute t. Let $A = \bigcup_{t=1}^{z} A_t$ be the set of all different attribute values that we collect, and let $a = |A|$. Also, let P_i, with $i \in \{1, 2, \dots, x\}$, be the different sets of attributes of user i. We can represent each P_i as a vector of size a, where the element $P_i(j)$ is 1 or 0 depending on whether this specific user has the specific attribute value, or not. For a particular user i, define the set $K_i = U_i \cup P_i$, which is the URL set of user i extended with his attributes as included in P_i. We represent K_i as a vector of size $b + a$.

We are now ready to present our personalization algorithm. Consider the scenario of having a user i who wants to personalize his searching results based on a few given attributes and their values. The algorithm proceeds as follows. It constructs a vector P_i^* of size a whose values are all zero except for those in the positions of the attribute values the user i wants to optimize. It then enforces the logical AND on P_i^* and the subvector consisting of the last a positions of K_i, that is, $P_i^* \wedge K_i[b+1, b+a]$. As a result, there will be vectors K_i, whose last a positions are now all zero. We consider all such users i as being "removed" by this operation, since $K_i[b+1, b+a] = [0, 0, \dots, 0]$ implies that these users do not have these attribute values. Denote by K_i^* all those K_i for which $K_i[b+1, b+a] \neq [0, 0, \dots, 0]$, and let $U_i^* = K_i^*[1, b]$ (the subvector of K_i^* consisting of the first b positions). Let also U^* be the set of all U_i^*, and let $x^* = |U_i^*|$; clearly, $x^* < x$.

The searching now concentrates only on the users i that belong to the set U^* and the page ordering is based on the pages they prefer.

The UsersRank values of all pages optimized for user attributes now become

$$R_{U^*}(j) = \sum_{k=1}^{x^*} U_k^*(j)$$

where $U_k^*(j)$ is the jth entry of U_k^*.

Note that the personalization as proposed is not based on the characteristics of the end user, but on what the people that belong to the selected groups of attribute values prefer. In particular, in the case of special interest groups or geographical limited searching[5] we can expect highly targeted results. Fig. 5 shows the process of restricting the search to users that share certain characteristics.

---

[5]In this setting, a geographical limited searching refers to the results that are produced by using information only from people that are located at a specific geographical location.
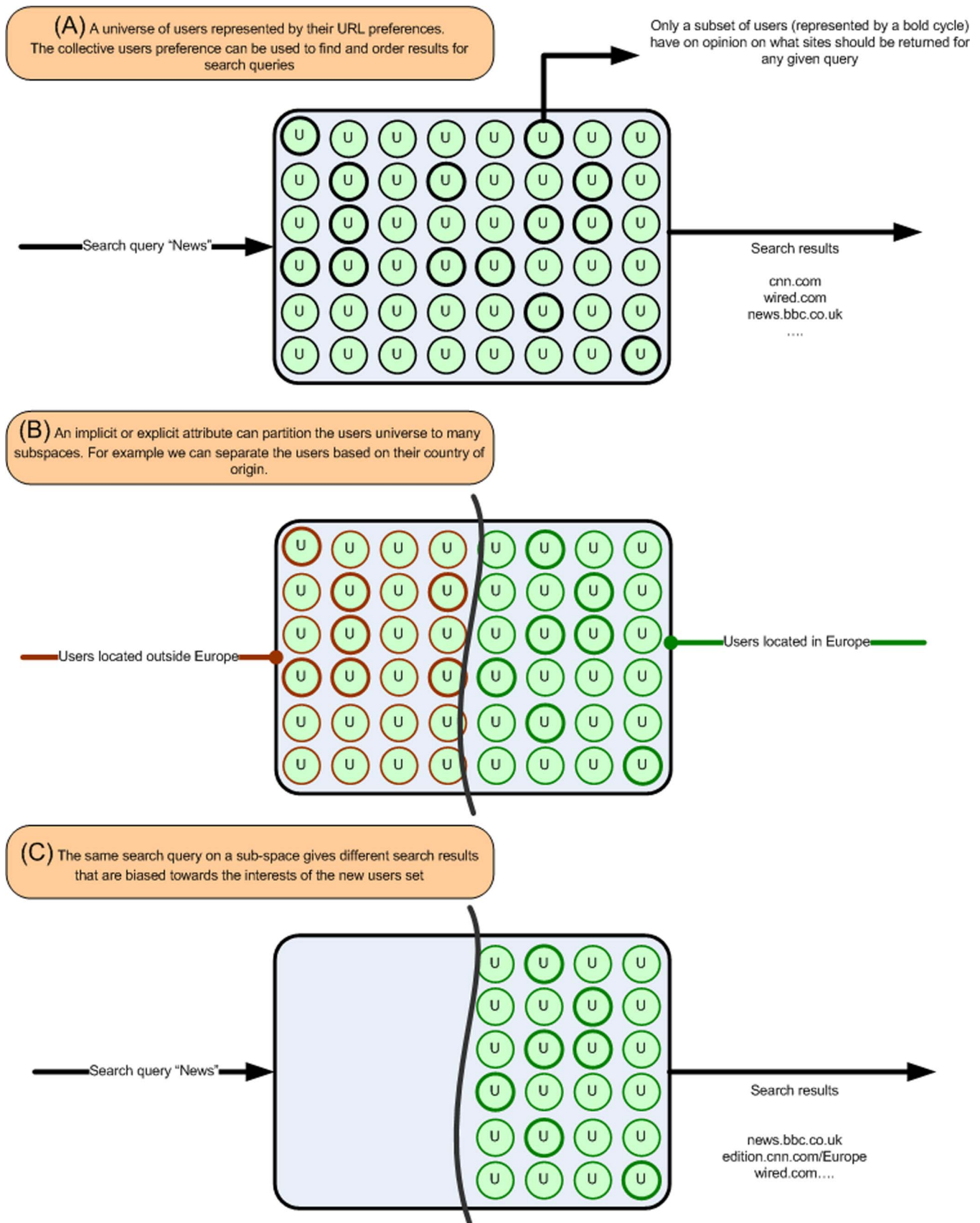
Fig. 5. Attributes can separate the users universe to related sub-spaces and effectively bias the searching results.

The proposed personalization has already been implemented in the Searchius prototype using standard database indexing techniques. Every user can tag his URL set with a set of charac-teristics. When someone poses a query to the database, he can restrain the characteristics of the users that will participate in the results.
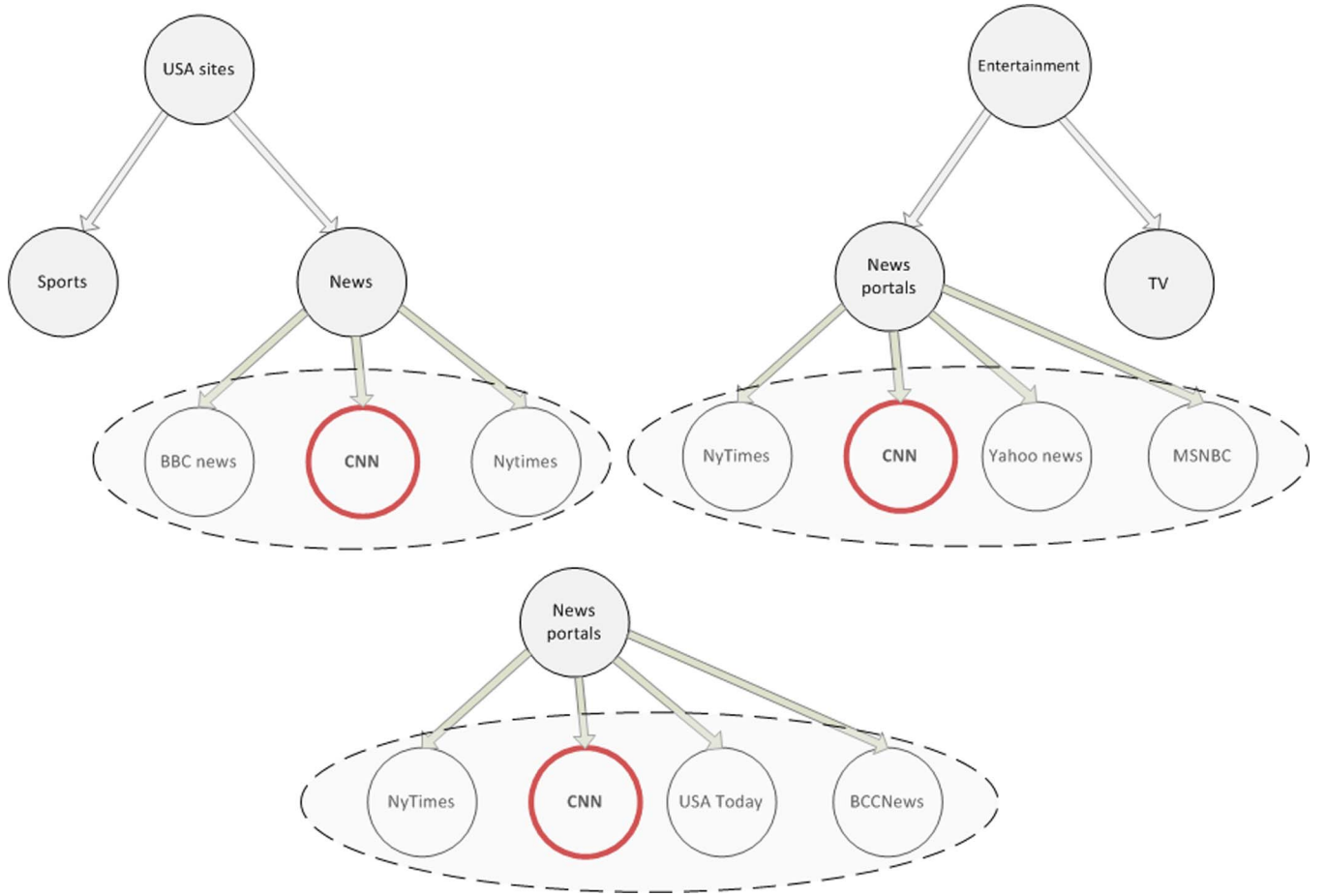
Fig. 6.   Exploiting the structure of URLs to find related pages.

VI. ADDITIONAL APPLICATIONS OF COLLECTED DATA

The structure of the collected data produces a wide range of possible additional applications as well as intuitive new methods to tackle known web-related issues from a different perspective.

In this section, we elaborate on one such application. More specifically, we present a simple algorithm to discover related pages. In Section VIII, we will see how the results of the new algorithm overlap with the results of a current search engine.

Current methods for finding related pages use link analysis as proposed in [14]. To achieve the same goal in Searchius, we exploit characteristics of the URL structure. When a user adds several pages under a folder, he actually groups them by some sort of similarity. This infers that folders partition the URL-space to conceptually related sub-spaces.

The algorithm we propose starts from an initializing URL and finds the different users and folders that have as child that URL. It then concentrates on the URLs under those folders counting the number of occurrences of each URL. It returns the URLs that occur more often on that subspace. In other words, it is an alternative version of UsersRank, where the scope of the algorithm is reduced to the related-to-the-URL users and their related-to-the-URL folders. We ignore all users that do not store the initial URL under some folder. This algorithm can control the accuracy over speed by reducing the maximum number of users that should contribute to the final ordering. Fig. 6 shows

how we exploit the URL structure in order to find related URLs. In this example, we use the domain of CNN as an initializing URL and we count the appearance of different URLs in the subfolders that contain the initial URL. For this particular case, the domain of NYTimes is the one most related to CNN.

Assume that B_k is an initializing URL for which we want to find related pages. Let the *parent concept* of a URL be the tag of its parent folder in the tree structure of its URL set. Let Z_{ij} be the parent concept of URL B_j on U_i. Note that Z_{ij} is zero, if $B_j \notin U_i$ or if B_j is not stored under a folder. Also, let Y be the set of users that store the page B_k under some folder. That is, $Y = \{\text{user } i : U_i(k) = 1 \land Z_{ik} \neq 0\}$. Finally, let $relevanceWithB_k$ be an integer b-vector, which uses URLs as indices. For a URL w, $relevanceWithB_k(w)$ stores an integer value indicating how relevant is w to B_k (the larger the value, the higher the relevance). The algorithm that finds pages related to a given page B_k is as follows.

Algorithm SimilarPages(B_k)

    for all $i \in Y$ **do**
        compute $P = \{B_p \in U_i : Z_{ik} = Z_{ip}\}$;
        for all $w \in P$ **do** $relevanceWithB_k(w)$++;
    sort $relevanceWithB_k$;
    return the top URLs of $relevanceWithB_k$.

VII. INTEGRATING USERSRANK AND PAGERANK

In this section, we describe how our approach can be integrated with an authority-based search engine and potentially improve the quality of searching results or bias the results toward certain group of user needs.

The UsersRank method of ordering pages can be viewed as an expert that has a very specific, global opinion on the exact value of each page in its database. Google's PageRank can be viewed as another expert that has another possible different opinion from the first expert opinion on the exact value of each page. This section provides a modification of Google's PageRank that takes into consideration not only the linkage structure of the Web, but also the feedback from users regarding the pages' value as returned by UsersRank. We start by briefly describing Google's PageRank and its intuitive justification.

The original PageRank algorithm [6] is given by

$$R(A) = \frac{(1-d)}{N} + d \sum_{i=1}^{I(A)} \frac{R(T_i)}{O(T_i)}$$

where $R(A)$ is the PageRank of page A, $R(T_i)$ is the PageRank of page T_i that links to page A, $O(T_i)$ is the number of outgoing links of T_i, N is the total number of web pages, $I(A)$ is the number of pages that link to page A, and d is a damping factor that can be set between 0 and 1.

In practice, the PageRank of pages is computed using an iterative procedure where at the first step all web pages are given an initial value. The iterations continue until the ranking of each page stabilizes within some threshold. The initial values of pages do not have any effect on the final rankings of pages; however, they can affect the number of iterations before rankings stabilize. The damping factor has been introduced to guarantee the convergence to a unique rank vector, but it also limits the effect of rank sinks [6].

An intuitive justification of PageRank is that of the *random surfer model* [6]. Imagine billions of surfers that initially are randomly distributed on the web graph. If at a certain time step a surfer is looking at page p, then at the next time step, he looks at a random outgoing neighbor of p. As time goes on, the expected number of surfers at each page p converges to a limit $R(p)$, which is independent of the initial distribution of starting points. This limit is the PageRank of p and reflects the number of people expected to be looking at p at any time. The probability for the random surfer of not stopping to click on links is given by the damping factor d, which is set between 0 and 1. The higher the d, the more likely the random surfer keeps clicking links. Since the surfer jumps to another page at random after he stopped clicking links, this probability is incorporated as the factor $(1-d)/N$ into the algorithm.

In [6], [18], an alteration of the initial algorithm that can produce biased PageRanks has been proposed. Imagine that after stopping following links, the random surfer does not jump at a totally random page, but it selects a page based on a probability vector. Then, the initial algorithm has to be modified as

$$R(A) = (1-d)\Pr(A) + d \sum_{i=1}^{I(A)} \frac{R(T_i)}{O(T_i)} \qquad (1)$$

where $\Pr(A)$ is the probability for the random surfer going to page A after he has stopped following links.

UsersRank can be used to produce such a global probability distribution $\Pr(\cdot)$ over web pages. Intuitively, when a user gets bored following links, he will jump to a page in his URL set. The total occurrences of a page inside the total users' URL set determine the probability of following this page given the user community as a whole. This gives

$$\Pr(A) = \frac{R_U(A)}{\sum_{p=1}^{L} R_U(p)}$$

where $R_U(A)$ is the UsersRank of page A and L is the total number of pages for which we have a UsersRank.

Note that in our approach, there might be pages for which $R_U(A)$ may be zero due to the fact that page A may not be in the UsersRank authority database. In this occasion, the damping factor effect in (1) is eliminated, since $\Pr(A) = 0$, and the total PageRank for that page may be zero. In an extreme case, this may also lead to the total elimination of the effect of damping factor which may prevent convergence. On the contrary, the original PageRank assigns a minimum value to every page, given by $(1-d)/N$. In order to address this issue, we produce an alternative version of PageRank, which ensures that each page has a minimum PageRank. To summarize, we propose the following alternative version of PageRank that incorporates UsersRank and is given by

$$R(A) = e\frac{(1-d)}{N} + (1-e)\frac{(1-d)R_U(A)}{\sum_{p=1}^{L} R_U(p)} + d \sum_{i=1}^{I(A)} \frac{R(T_i)}{O(T_i)}$$

where e represents a second damping factor that can be set between 0 and 1. Intuitively, when a user gets bored following links he has two options: either to follow a random link as suggested by the pure PageRank, or to follow a link that lies inside the URL sets of all users. The second damping factor represents the probability of selecting between these two options.

The aforementioned methodology can be used to bias the PageRanking algorithm toward the needs of distinct groups of users. The new setting is similar to what we already discussed in this section; only this time, we assume that the UsersRank is based on a subset of users. We can define subsets of users that share similar characteristics with techniques like the one described in Section V. Under such a setting, the resulted ordering of pages would be biased from the needs and interests drawn from the initial users subset.

A. Effect of Damping Factors

The main focus of the paper is on the dynamics of user-generated data and the applicability of the UsersRank approach for building a search engine. Under this point of view, it is out of the scope of this paper to actually integrate UsersRank with a traditional search engine. However, for the shake of completeness, we briefly demonstrate the effect of different damping factors on the modified PageRank computation with the help of sample graphs.
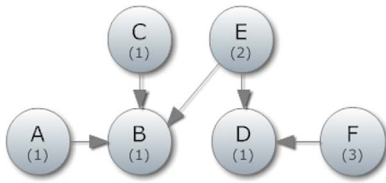
Fig. 7. Sample graph with embedded UsersRank values to every node.

TABLE I
COMPUTATION OF THE MODIFIED PageRank FOR DIFFERENT COMBINATIONS OF DAMPING FACTORS

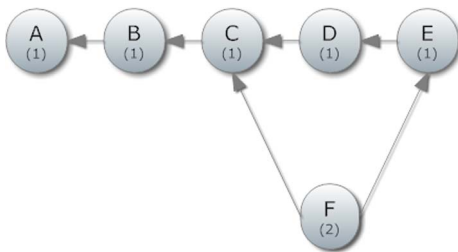|	A	B	C	D	E	F
d=0.25, e=0.5	0.58	0.99	0.58	1.00	0.99	**1.19**
d=0.25, e=0.0	0.41	0.77	0.41	0.97	1.23	**1.64**
d=0.50, e=0.5	0.39	**0.94**	0.39	**0.95**	0.66	0.80
d=0.50, e=0.0	0.27	0.75	0.27	1.02	0.82	**1.09**
d=0.75, e=0.5	0.19	**0.61**	0.19	**0.62**	0.33	0.40
d=0.75, e=0.0	0.14	0.49	0.14	**0.70**	0.41	0.55
d=0.25, e=1.0	0.25	**0.72**	0.25	0.53	0.25	0.25

Fig. 8. Second sample graph with embedded UsersRank values to every node.

When e is set to 1, then our modified algorithm is transformed to the traditional PageRank algorithm. When e is set to 0, then we do not teleport users to random sites. Instead, we consider that one can be transfered only to a bookmarked site. When we set d to 0.5 and e to 0, then we have *two equally balanced experts*. Intuitively, the higher the d, the more important the influence of link-structure. In the same fashion, the lower the e, the more important the combined users' opinion on page value.

For our demonstration, we use combinations of the following dumping values: $d = \{0.25, 0.5\}$, $e = \{0, 0.5\}$. We also include a combination with $e = 1$ and $d = 0.75$, which renders to the traditional PageRank.

Consider the graph in Fig. 7. Each node includes a *number* that represents the different people that have bookmark the web page it represents; this number can be used to compute the UsersRank value of this node. Table I presents the PageRank values of nodes for available combinations of damping factors. Each row has in bold the number that corresponds to the node with the higher PageRank value.

When d is low then pages with higher UsersRank get an important boost on their value (evident on the first two rows). When d and e are equal, then we see a good balance between the two metrics. For example in the third row, node B and D get almost identical PageRank values. When d and e are high, then the algorithm is closer to the traditional PageRank computation. For example, compare the fourth row with the traditional PageRank computation on the sixth row.

Fig. 8 and Table II present another graph and the corresponding PageRank computations. Again, when d is low then nodes

TABLE II
COMPUTATION OF THE MODIFIED PageRank FOR DIFFERENT COMBINATIONS OF DAMPING FACTORS (SECOND GRAPH)

	A	B	C	D	E	F
d=0.25, e=0.5	0.94	0.96	**1.05**	0.90	0.82	1.02
d=0.25, e=0.0	0.87	0.90	1.01	0.84	0.80	**1.29**
d=0.50, e=0.5	0.95	0.98	**1.02**	0.78	0.63	0.68
d=0.50, e=0.0	0.90	0.94	**1.02**	0.75	0.64	0.86
d=0.75, e=0.5	**0.82**	0.78	0.74	0.50	0.36	0.34
d=0.75, e=0.0	**0.79**	0.77	0.75	0.50	0.38	0.43
d=0.75, e=1.0	**0.85**	0.79	0.72	0.51	0.34	0.25

with higher UsersRank get a boost on their value. For example, node F has a higher value than node A on both the first and second rows. However, as the value of d increases this effect diminishes. When d and e are equal then we see a balanced behavior with node C getting the higher PageRank. Finally, as d increases the computation gets closer to the traditional PageRank.

VIII. EXPERIMENTS

We have conducted an extensive experimental study based on URL collections from 36 483 users that produced a database of 1 436 926 URLs. This amount of data was enough to produce highly fit functions that describe the underline phenomena and predict how accuracy, coverage, and recall will develop in larger scales.

The data was produced partly from user submitted data at Searchius' web site and partly from crawled collections of public URL sets from [38]. We conducted experiments toward four major goals aiming to the following.

1) Identify the internal characteristics of URL collections and discover patterns on the way people collect and organize URLs. These experiments are expected to reveal several key characteristics of URL collections and give qualitative answers on whether people still collect URLs and in what manner, what sites they prefer to collect, and whether they tend to organize these data and to what extend.

2) Identify the growth rates that URLs and search keywords are accumulated and give functions to estimate their growth based on the number of users. In this way, we can project the size of our database to larger scales and estimate its web coverage characteristics.

3) Measure the effectiveness of Searchius as a search engine. Our effectiveness measure is the amount of overlap between the results obtained by Google and our approach. Assuming that our approach is reasonable, we should expect *some* consistent overlap between our results and those of other search engines. The same method was used to find overlaps between related pages.

4) Justify the utility of humans judgment over URLs quality. This experiment aims to clarify whether the people collectively decide about the quality of web pages somewhat different than automatic metrics like PageRank. More specifically, we compare the PageRank and UsersRank of several thousand URLs trying to identify differences and similarities.

TABLE III
DATABASE CHARACTERISTICS

Distinct Search Terms	312,572
Distinct URLs	724,116
Total URLs	1,436,926
Distinct Folders	69,287
Total URLs at Level 1	1,140,193
Total URLs at Level 2	310,114
Total URLs at Level 3	57,978
Distinct Users	36,483
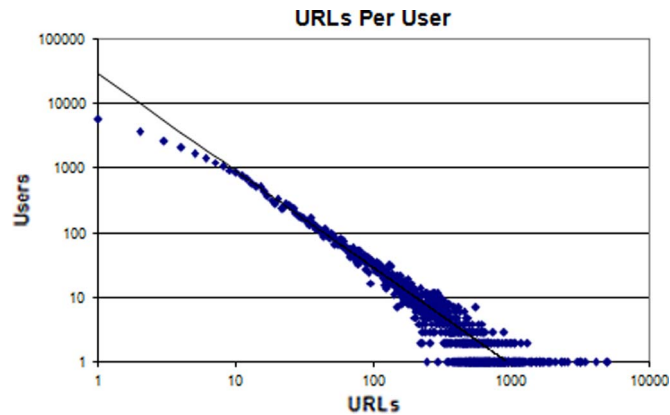
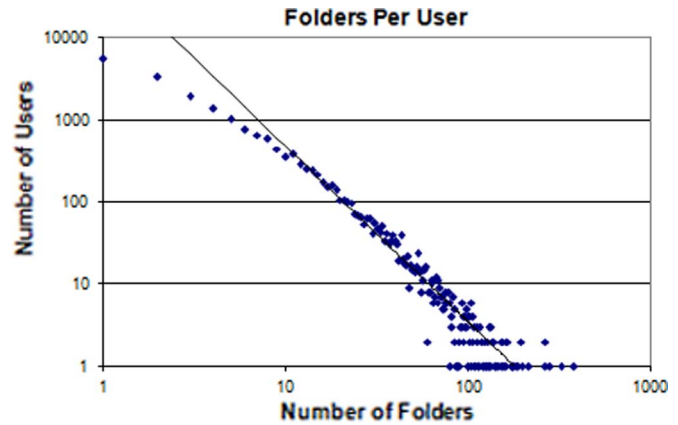Fig. 9.   Log-log distribution of URLs per user.

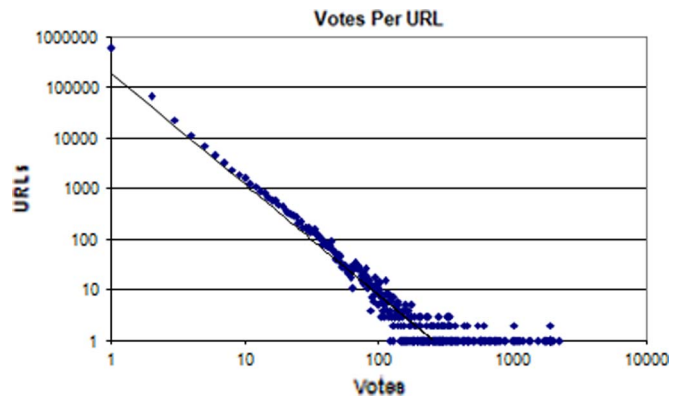Fig. 10.   Log-log distribution of folders per user.

Fig. 11.   Log-log distribution of votes per URL (number of different users that have stored a URL).

TABLE IV
POWER LAW COEFFICIENTS FOR SEVERAL DATABASE METRICS

Description	Function	R-Square
URLs/User	$y = 28112x^{-1.496}$	0.9997
Folders/User	$y = 64970x^{-2.135}$	0.9994
Votes/URL	$y = 187200x^{-2.193}$	0.9999

A. Identification of Internal Characteristics of URL Collections

Table III presents the basic characteristics of the collected URL sets. *Distinct Search Terms* refers to the unique words inside our database. The difference between *Distinct URLs* and *Total URLs* is the number of URLs that reside on *more than one user's information space*. Without this overlap of preference-for-certain-sites, we would not be able to produce a UsersRank ranking of pages. *Distinct Folders* refers to the set of descriptors people use to organize their URLs. Recall that we use these descriptors during the search process. A folder name can be used to extend the IR score of many pages for a multitude of search queries. Folder names are treated as semantic tags over pages and are the equivalent of anchor links. In other words, folder names help associating pages with many queries even when the actual page title or URL may not include the search words of the query. *Total URLs at Level 1* represent the number of URLs that reside under at least one folder name. Note that people can just add a URL to their bookmarks list without placing it under a folder. *Total URLs at Level 2* and *Total URLs at Level 3* represent the number of pages that have at least two and three folder descriptions, respectively. The folder names at every level extend the possible query matches for that page. As we can see from Table III, the vast majority of URLs are at least under one folder, while more than 25% of URLs are under at least two folders.

To further analyze the way users organize their URLs, we constructed several graphics based on different user-URL characteristics. Fig. 9 plots the URLs per user, Fig. 10 reports the distribution of folders per user and Fig. 11 shows the distribution of user votes (number of different users that have stored a URL) over pages. All three graphics have a good linear

fit in a log-log scale which indicates a power law distribution. To find the coefficients of the produced power law distributions, we used the Trust-Region nonlinear least-square algorithm to fit the data [5]. We also measured the R-square (the coefficient of multiple determination) of their overlap to find how successful the fit is in explaining the variation of the data. For all graphics, the correlation coefficient was higher than 99% which indicates a very accurate fit. These results are in accordance with recent reports regarding power law distributions on web topology [15] and user's page access behavior [21]. Table IV presents these results and the produced coefficients.

We can use these functions to estimate generic characteristics of URL collections like the number of users with a specific number of URLs or the percent of URLs that have at least a specific number of votes. Furthermore, any unbiased collection of user constructed bookmark sets should follow similar power laws and thus, we can evaluate the quality of user provided URL collections and possibly detect excessive fraudulent submissions.
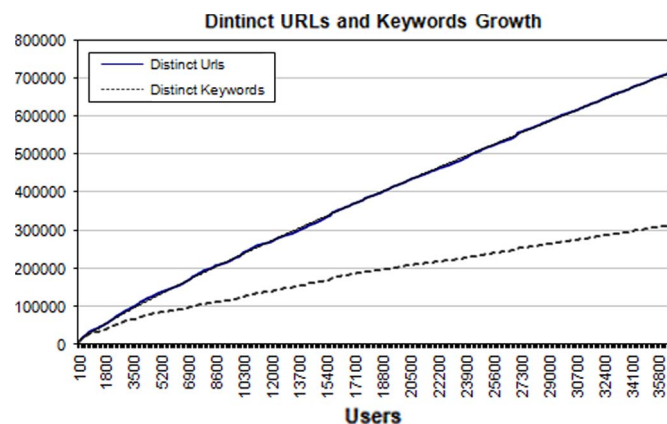
Fig. 12. Heaps' Law distributions for the number of distinct URLs and Keywords while constructing the database.

TABLE V
HEAPS' LAW COEFFICIENTS FOR URLS AND KEYWORDS GROWTH

Description	Function	R-Square
URLs Growth	$y = 76.04x^{0.8709}$	0.9998
Keywords Growth	$y = 191.1x^{0.7027}$	0.9988

B. Identification of the Growth Rates of Data

This set of experiments is based on the analysis of the database characteristics during the database construction phase. Before initiating this experiment, we shuffled the URL sets and then we reconstructed the search database, while at given user intervals, we measured several database metrics. Two of the most interesting measures are those of the different keywords and different URLs in the database. These metrics are particularly interesting because they were found to be immune to the user ordering, while at the same time, they can help to predict the amount of URL sets we need to obtain a satisfactory coverage of the web compared to other search engines. Fig. 12 shows the distinct URLs and keywords growth, while Table V presents the coefficients of their functions found using the Trust-Region nonlinear least-square algorithm [5]. Both growths follow the Heaps' Law [19];[6] As more URLs and keywords are gathered, there are diminishing returns in terms of discovery of the full URLs and keywords from which the distinct terms are drawn.

These functions are valuable in predicting the *coverage* of the web that we can expect as a function of the number of users. Fig. 13 shows a large-scale projection of the expected distinct URLs and keywords based on the number of users. This figure tells us that if we would like to build a database that has the quantity of information provided by Google (> 8 billion pages), then we would need > 1.6 billion users. However, as our next experiment shows, important sites are discovered *earlier* during the database construction phase, and thus, they will be included on search results even with relatively small databases.

The next set of experiments sheds light on the manner in which the UsersRank of a page is distributed among sites.
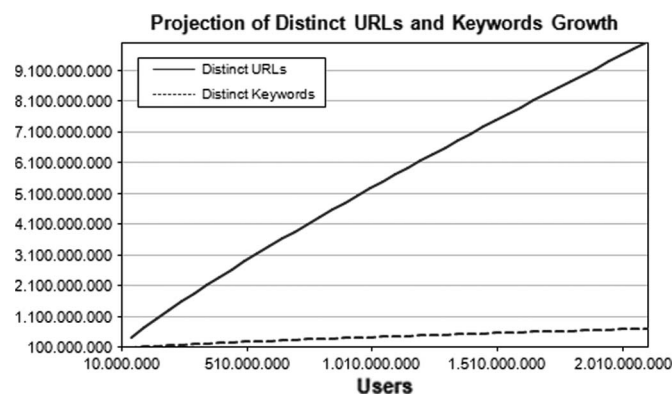

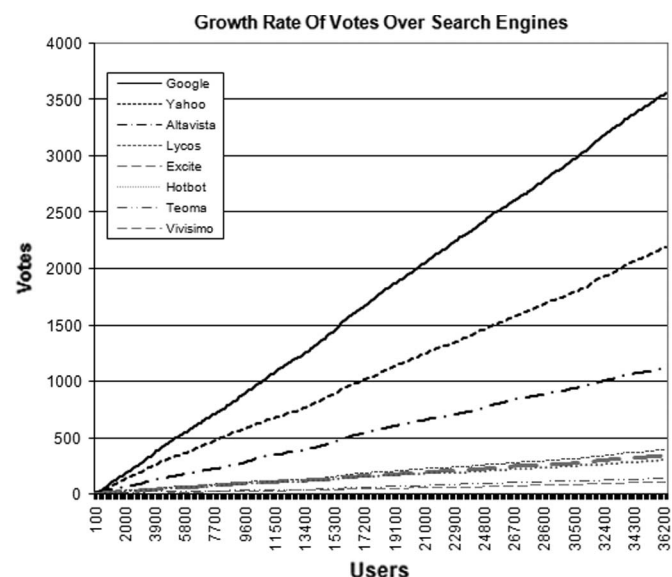Fig. 13. Projection of distinct URLs and keywords growth.

Fig. 14. Growth of UsersRank for several search engines.

Again, while reconstructing the database, we measured at given user intervals the UsersRank that several sites have accumulated. Figs. 14 and 15 illustrate the Bookmarks (Votes) accumulation process for several search engines and e-commerce sites accordingly. These graphics demonstrate a linear relationship between the number of users and the UsersRank over sites. Depending on the site, this linear relationship has a different slope that characterizes the site. Table VI summarizes the discovered approximation functions for the given sites.

An important consequence of these graphics is that the *most valuable-for-users sites will appear even in a relatively small users database*. This is indeed the case, since important sites will have steeper slopes, and thus they will appear *earlier* during the database construction phase. Note that the slope for a given URL can be used as an indication of its relative value and that it should stabilize as soon as we have enough data to describe it.

C. Comparison With Traditional Search Engines

Our third set of experiments aims at investigating the effectiveness of the proposed search process. As a metric of effectiveness we use the results overlap between our approach
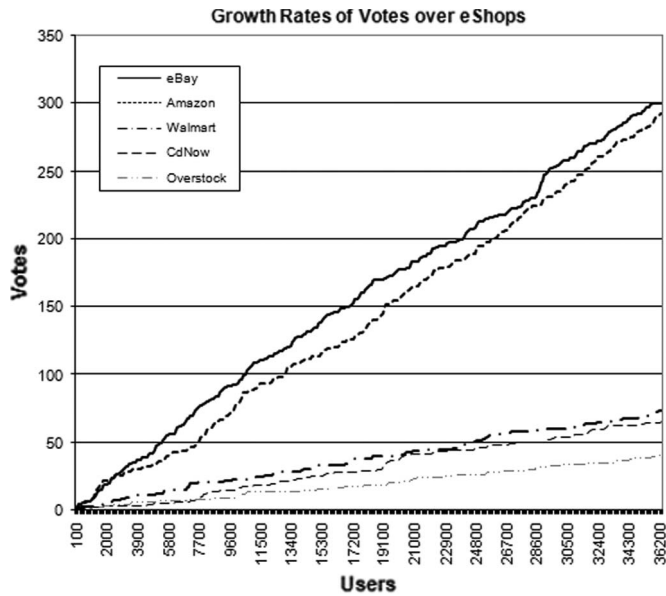
Fig. 15.   Growth of UsersRank for several eShops.

TABLE VI
UsersRank Growth Coefficients for Several Sites

Site	Function	R-Square
Google	$y = 0.09894x - 38.35$	0.9997
Yahoo	$y = 0.0589x + 16.41$	0.9996
Altavista	$y = 0.03122x - 9.382$	0.9991
Lycos	$y = 0.0101x + 5.814$	0.9968
Excite	$y = 0.00886x + 5.323$	0.9982
Hotbot	$y = 0.007941x + 7.439$	0.9972
Teoma	$y = 0.003698x - 4.037$	0.9893
Vivisimo	$y = 0.002999x - 6.62$	0.993
EBay	$y = 0.008115x + 8.911$	0.9964
Amazon	$y = 0.008007x - 4.631$	0.9975
CDNow	$y = 0.00192x - 3.616$	0.9912
Walmart	$y = 0.001921x - 2.157$	0.9913
Overstock	$y = 0.001058x - 0.05651$	0.9916

and those produced by Google. Our intention is not to directly compare Searchius with Google—something like that would be inappropriate taking into account the limited size of our database—but rather to use the results obtained from Google as an indication of quality. The idea is that if our approach is effective, then it should produce *some consistent* overlaps with the results of other established search engines. At the same time, it should not mimic their behavior but produce, to some extent, its own unique and interesting results. Recent studies [31] have shown that the first page results among different search engines produce a small but substantial overlap. For example, Google produces an overlap between 20% and 25% on the first page results with the following search engines: Yahoo, Ask Jeeves, MSN, Dogpile.

We use the existence of consistent overlaps as an indication that Searchius can produce *meaningful results*. Meaningful results in our context are results that provide end users with important and related URLs for search queries. It is truly difficult to compute how satisfied are the users from the provided URLs, and probably it is better to rely on user surveys to get some insight. Still, it is undeniable that a widely adopted search engine like Google would get a good score on this metric.

From an end-user point-of-view, an important page is one that is shown on the first one or two pages of search results. People do not have the patience or time to traverse through many pages of results. To this end, a metric based on common first results between Google and Searchius (as detailed below) is easy to understand and reasonable enough given its proposed scope. Again, these tests do not try to prove *superiority* but rather *applicability* of the proposed UsersRank metric.

For this experiment, we concentrated on a broad collection of one word generic queries from diverse topics. It is likely to expect bad results on long or very specific queries, since we have a medium-sized URLs database and we do not collect the actual page content. One word queries give a better understanding of the potential of both ordering and important page discovering as described in this paper.

For the experiments, we used two sets of one-word queries and one set of URLs. The first set (set-A) is a small hand-picked collection of 88 words spanning many topics from *cities* to *emotions*. The second set (set-B) is a collection of 41 234 generic English words taken from [39]. We use set-A to demonstrate several specific characteristics of the proposed search engine and set-B to describe general trends. We also used a collection of 190 first level domains (set-C)—produced using a certain methodology that we describe later on—to compare the results of our related-URLs algorithm with the results of Google.[7]

To perform the experiments, we implemented an automatic comparison tool. This tool is fed with a set of queries that is performed on both search engines. Afterwards, it checks for common returns on the first 20 (top-20) results. We opted to use this method, because from a user perspective, the first 20 results are typically what one actually sees and uses. The comparison tool also collects and reports the total number of relative URLs for each given query both for Searchius and Google. A similar tool has been implemented for testing the overlap between related-URLs results, only this time the tool is fed with sets of URLs for which we want to find related URLs. Again, this tool checks for common URLs on the first 20 results.

A general overview of the search results is as follows. Our experiments showed that there is indeed consistent overlap of the results between Searchius and Google. For our experiments, the overlap was between 16% and 32% depending on the type of experiment.

Let us analyze Searchius performance through a Precision/Recall point-of-view [10]. *Precision* is the fraction of retrieved instances that are relevant, while *recall* is the fraction of relevant instances that are retrieved. Both precision and recall are therefore based on an understanding and measure of relevance. A high recall means you haven't missed anything but you may have a lot of useless results to sift through (which would imply low precision). High precision means that everything returned was a relevant result, but you might not have found all the relevant items (which would imply low recall). For most topics Searchius has returned an order of magnitude less results than Google, which implies low recall (Fig. 17). However, it

[7]Note that since search engines are dynamic systems the actual search results may be slightly different over time.

Percent of common URLs at first 20 search results between Searchius and Google for 88 one-word queries

Search Query	Percent
Average	23%
Blue	25%
Green	20%
Purple	5%
Red	15%
Education	45%
Careers	15%
Jobs	20%
Egypt	25%
Spain	15%
Italy	20%
Japan	25%
Germany	15%
Mexico	15%
USA	10%
Canada	20%
Bible	45%
Religion	20%
God	15%
Kids	45%
Love	10%
People	20%
Animals	20%
London	20%
New York	35%
Chicago	30%
Athens	15%
Rome	20%
Alexander	15%
Aristotle	25%
Bush	15%
Newton	5%
Einstein	40%
Shopping	10%
Currency	45%
Finance	15%
Magazines	0%
Newspapers	15%
News	30%
Maths	35%
Chemistry	15%
Biology	15%
Science	10%
Astronomy	35%
Encyclopedia	60%
Space	35%
Geography	35%
Travel	30%
Tickets	30%
Maps	35%
Weather	40%
Wine	5%
Food	25%
Health	40%
Genetics	10%
DNA	10%
Nokia	0%
Sun	10%
Intel	0%
Apple	15%
Microsoft	20%
IBM	10%
Hockey	25%
Tennis	5%
Basketball	20%
Football	30%
Hardware	15%
Java	20%
Hackers	5%
Antivirus	35%
Virus	45%
Search Engines	55%
Downloads	20%
Shareware	55%
Windows	20%
Linux	55%
Cartoon	25%
Comics	25%
Quotes	25%
Puzzles	25%
Chess	25%
Museums	10%
Theatre	20%
Movies	30%
Poetry	40%
Books	30%
Entertainment	20%
TV	10%
Music	30%

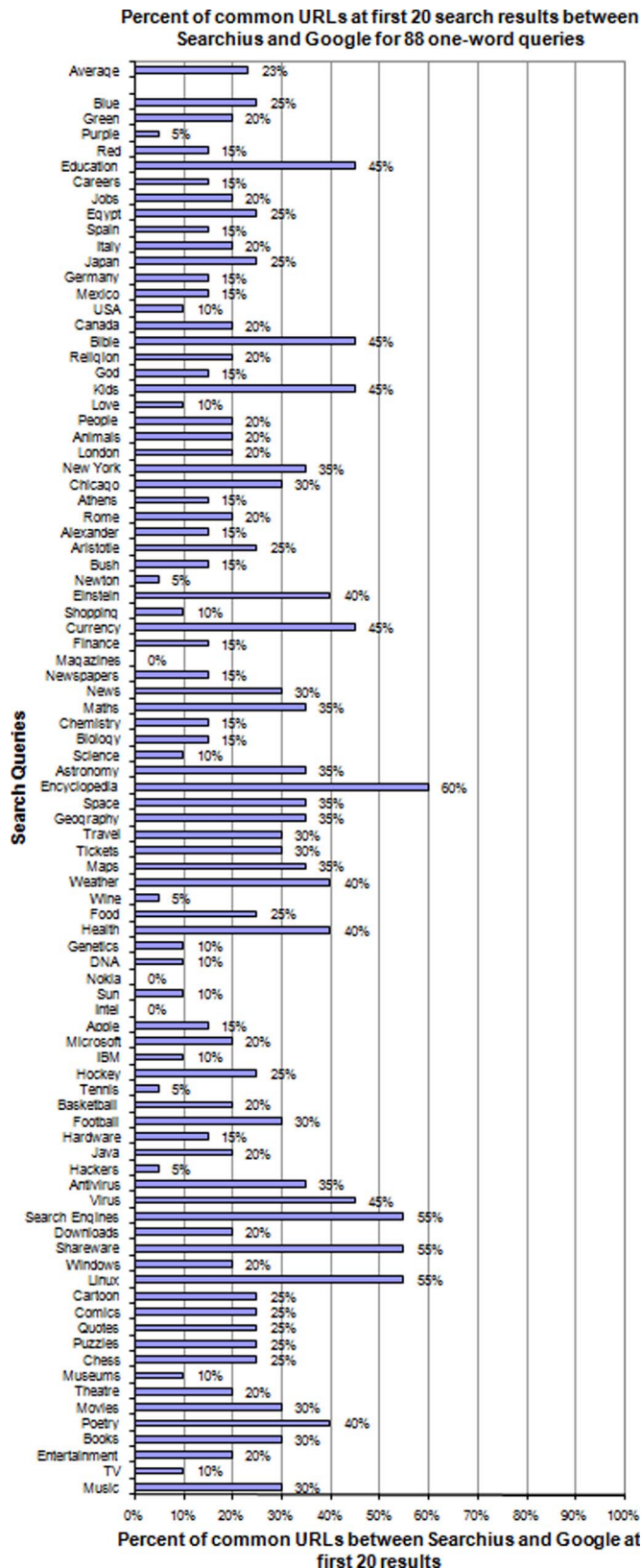Percent of common URLs between Searchius and Google at first 20 results

Fig. 16. Search results overlap between Searchius and Google for the set-A of queries.

provided a highly overlapped top-20 results for a broad number of topics which implies a high precision (Fig. 16). In other words Searchius is a *low recall, high precision* search engine.
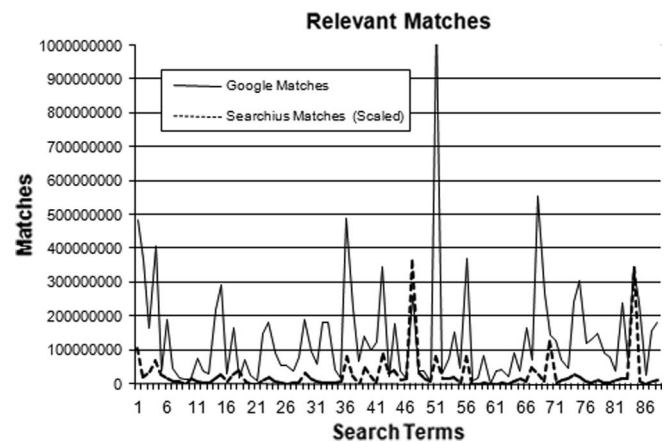

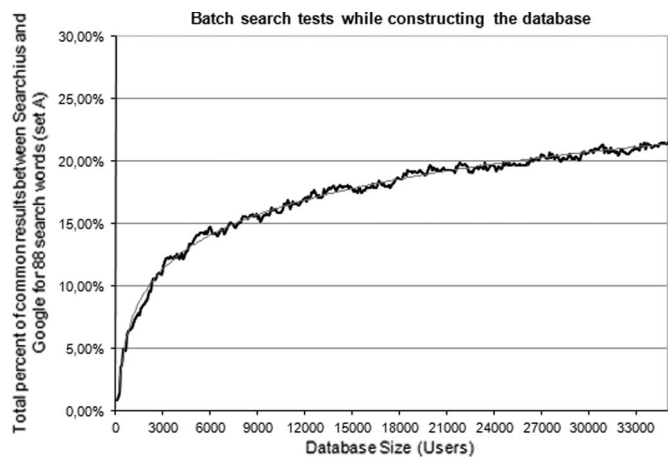Fig. 17. Relevant matches under Google and Searchius.

Fig. 18. Batch search tests for set-A queries, while constructing the database.

Experiments With Set-A: Fig. 16 summarizes the results overlap between Searchius and Google for set-A. Although the number of the relevant results for the proposed search engine are several orders of magnitude lower than Google's, we (surprisingly) found that there are consistent overlaps on the top-20 results. For set-A, the average overlap on the first 20 results was 23%.

Fig. 17 shows the total number of relevant results we got from both search engines for set-A. The results of Searchius are scaled using the ratio of URLs of Google over Searchius. Since we do not collect the actual page content it was expected that for the most part even the scaling would produce less matches for Searchius. However, what is interesting is that for most of the queries, the number of matches follow similar patterns for both search engines. In other words, if for a specific query we have a high density of relative documents at Google, we can expect a relatively high level of density under Searchius as well.

Fig. 18 shows the average results overlap we got for set-A during a simulated database construction phase. For this test, we incrementally created the database by adding new users to it, and on predetermined points, we performed the accuracy test for set-A. This experiment produced an increased level of average overlap as more users were added to the database and the system could order URLs based on more feedback.

TABLE VII
FITTING FUNCTION FOR THE SET-A BATCH TESTS,
WHILE CONSTRUCTING THE DATABASE

Description	Function	R-Square
Set-A overlap	$y = 1055x^{0.04526} - 1314$	0.9934
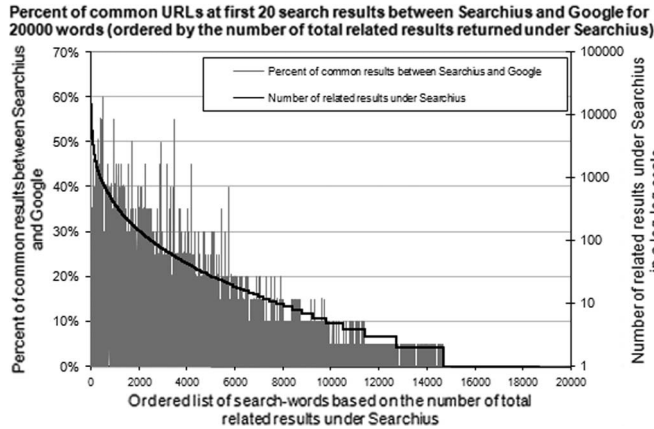
Fig. 19. Search results' overlap between Searchius and Google for 20 000 set-A queries.

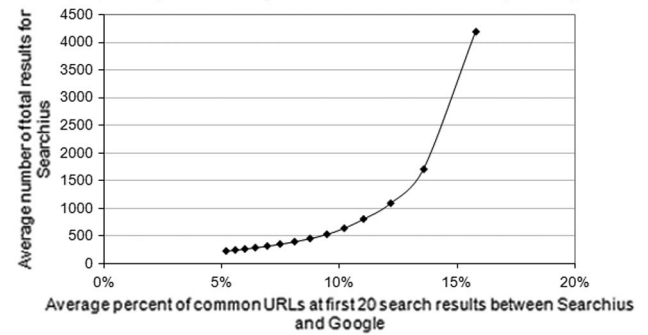Fig. 20. Prediction of search results' overlap between Searchius and Google depending on the total number of Searchius results.

Table VII presents the coefficients of the fit function for the produced graphic and the metrics of goodness of fit as described earlier.

This function suggests that we can incrementally achieve better overlap with Google's result. Assuming a database of 10 million users, then the overlap would be around 50%. However, this function also suggests that no matter how many users we add in our database, there are practical limitations on the maximum overlap we can expect between the results of the two search engines. Let us assume for a moment that the aforementioned function can be used to predict outcomes at much broader scales. Then, we would need 2 billion users to achieve an overlap of around 80%. More complex queries would make the results' overlap even more difficult.

Experiments With Set-B: Set-B is a broad-scope, single-words set (41 234 words). Through its use, we aim to get strong general indications of what to expect from Searchius regarding results' quantity and ordering quality.

An interesting relationship between quality of results and total number of related results is shown in Fig. 19. This figure suggests that the overlap on the top-20 results between Searchius and Google is bigger when the overall number of Searchius results for a query is bigger. Fig. 20 shows that in a better way by predicting the overlap we can expect depending on the total Searchius results. For example, for a set of single-word queries that returned 4000 results on average, the overlap between Searchius and Google top results would be around 16%.

Experiments With Set-C: Our final experiment compares the related URL results obtained from Searchius and Google. In order to create set-C, we adopted the following methodology. Initially, we excluded all URLs that did not have themselves under some folder. Afterwards, we ordered the remaining URLs based on their popularity among users. Finally, we removed all URLs that were not first level domains. From the remaining set, we picked the first 190 most popular URLs. This methodology

produced a set of URLs for which we have a satisfactory level of feedback from users and we can predict relative-URLs with confidence. Again, we used an automatic tool to discover the overlap between URLs on the top-20 results. Fig. 21 presents these results. For this set of URLs, we got an average overlap of 32%.

D. Comparing PageRank With UsersRank

PageRank is the main quality metric used by Google to decide about the quality of pages. Although the exact method used to compute PageRank is not publicly available, the PageRank for any given URL can be retrieved directly. PageRank is expressed on a logarithmic scale and normalized on a range from 0 to 10.

For this experiment, we used 20 000 URLs for which we have enough votes to determine a UsersRank (URLs with more than 2 votes inside our database). To be able to directly compare the UsersRank with PageRank, we computed a logarithmic UsersRank for each page similarly to the way Google does it. Our logarithmic version of UsersRank is expressed as simply being the rounded log(UsersRank) for any give URL. For our current database, the aforementioned procedure provided UsersRanks on a scale from 0 to 12.

PageRank is distributed in such a way that only a small portion of the billions web URLs can have a PageRank above 5. By means of the iterative calculation, the sum of all pages' PageRanks converges to the total number of web pages. Hence, the average PageRank of a web page is 1. On the contrary, the URLs in our database have an average PageRank bigger than 5; more precisely for all URLs that had at least 3 votes from users (UsersRank > 2) the average PageRank is 5,509. This clarifies that humans indeed filter the collected URLs providing an initial high quality URLs database.

Fig. 22 directly compares the UsersRank and PageRank of URLs inside our database. For each UsersRank (from 0 to 12), we report the average PageRank and the range (high-low) of PageRanks discovered. For example, for all URLs with UsersRank equal to 6 the average PageRank was 6.3, the highest reported PageRank was 9, and the lowest was 4. The most important finding is that a page with a high UsersRank will most certainly have a high PageRank as well; however, pages with low UsersRank often have a high PageRank. In
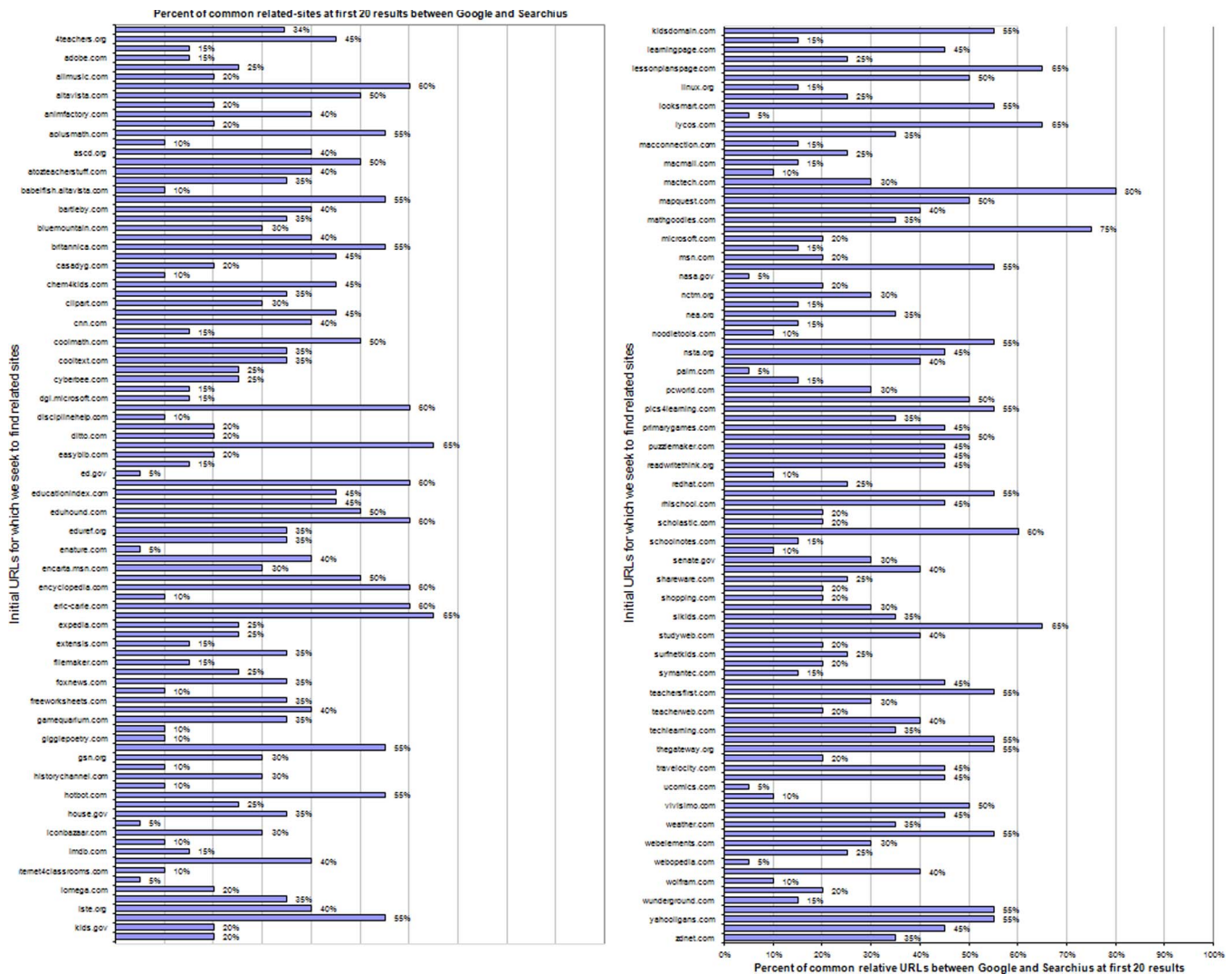
Fig. 21. Related URLs' overlap between Searchius and Google (percent of common matches at the first 20 results) for set-C.
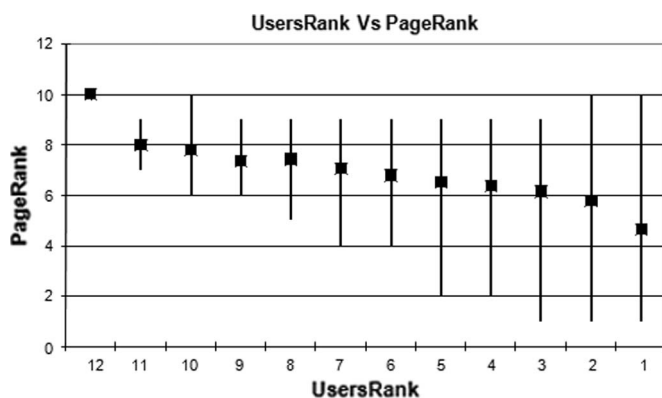
Fig. 22. UsersRank versus PageRank for 20 000 URLs.

IX. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a new, bottom-up approach to study the web dynamics based on users feedback. Also, we have described the algorithms and mechanics of a peer-to-peer, bottom-up search engine that can provide search results by combining the users preference regarding web pages. We have also discussed extensions to our approach that can provide personalized results. Finally, we have described how our approach can be integrated with PageRank, providing an alternative version of PageRank that combines two authorities: the link analysis and the users' preference.

This work was accompanied by an extensive experimental analysis that demonstrated the qualitative and quantitative characteristics of user-specific URL collections and compared our bottom-up search engine with a traditional search-engine using the results' overlap as an indication of effectiveness. Furthermore, we experimentally showed that the average user-collected URL is of high quality and that there are fundamental differences between UsersRank and PageRank.

Although we tried to answer as many questions during our research, there are always open issues for future work. One

other words, there can be a fundamental difference between the understanding of high quality within the users community, being often less interested in certain URLs, and what Google reports as high quality. Furthermore, this difference indicates that UsersRank has unique attributes not found on PageRank and implies that combining both metrics can be beneficial.

issue is to find methods to automatically analyze the quality of submitted URL sets and assign them a score in order to avoid spamming and low quality submissions.

A second issue is to investigate how and if we should include into Searchius data from users' browsing history. Although such data is a rich source of information, collecting them poses much more privacy concerns than the users' bookmark sets, mainly because they include, except for users preference, their surfing behavior. However, like in our system, before sending data to the central server, we may ask what part of the data should be transmitted.

Third, we would like to extend our personalization approach. We have discussed a personalization extension of Searchius at the group level. However, there are straightforward extensions that could produce personalized results at the user level. For example, we could devise a distance metric to compare the URLs of a user with the URLs of all other users. Then, when a specific user poses a query we could limit the users that participate on the results based on that metric.

Fourth, we plan to test the effectiveness of *reward* schemes. In the context of collaborative services a reward scheme offers incentives for end users to contribute. As an example, *eMule* uses a reward scheme by enforcing an upload to download ratio; end users can download more when they upload more. In a similar notion, a collaborative search engine could offer premium access to contributing members or restrict the number or type of searches for noncontributing users.

Finally, in this work we provided a generic merging process of UsersRank and PageRank. A reasonable future research direction is to evaluate its effect in the search results of a current search engine.

X. RELATED WORK

There are a few papers and web sites that share common ideas with our work. Early work on understanding how people interact and collect information from the web can be found in [8], [11], [32]. An initial experimental study on the characteristics of personal bookmarks collections can be found in [1]. Another work under the same context can be found in [12] where the architecture of a pure peer-to-peer system is presented that offers a distributed, cooperative, and adaptive environment for URL sharing. Potential benefits of integrating new types of searching data (bookmarks and web history data) for personalization and improved searching accuracy are discussed in [13], [18]. The implication of social bookmarks as a way to enhance search in the web was considered in [20], [35] with mixed results. Most previous studies regarding social bookmarks focused on their categorization ability (folksonomy) [16], [25], [36].

GiveALink (http://www.givealink.org) is a public site where people can donate their bookmarks to the Web community for research reasons. A method for online sites bookmarking is used at Google (http://bookmarks.google.com) and Yahoo (http://bookmarks.yahoo.com). These sites allow users to store URLs directly in an online account and not in their browser. People can access their URLs from any PC with internet access.

REFERENCES

[1] D. Abrams, R. Baecker, and M. Chignell, "Information archiving with bookmarks: Personal web space construction and organization," in *Proc. CHI*, 1998, pp. 41–48.

[2] B. Amento, L. Terveen, and W. Hill, "Does "Authority" means quality? Predicting expert quality ratings of web documents," in *Proc. 23rd Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2000, pp. 296–303.

[3] R. Amstrong, D. Freitag, T. Joachims, and T. Mitchell, "WebWatcher: A learning apprentice for the world wide web," in *Proc. AAAI Spring Symp. Inf. Gathering Heterogeneous, Distributed Environ.*, 1995, pp. 6–12.

[4] R. Barrett, P. Maglio, and D. Kellem, "How to personalize the web," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 1997, pp. 75–82.

[5] M. Branch, T. Coleman, and Y. Li, "A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems," *SIAM J. Sci. Comput.*, vol. 21, no. 1, pp. 1–23, Aug./Sep. 1999.

[6] S. Brin, R. Motwani, L. Page, and T. Winograd, "What can you do with a web in your pocket," *Bull. IEEE Comput. Soc. Tech. Committee Data Eng.*, vol. 21, no. 2, pp. 37–47, Jun. 1998.

[7] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *Proc. World Wide Web Conf.*, 1998, pp. 107–117.

[8] L. Catledge and J. Pitkow, "Characterizing browsing strategies in the world-wide web," *Comp. Netw. ISDN Syst.*, vol. 27, no. 6, pp. 1065–1073, Apr. 1995.

[9] L. Chen and K. Sycara, "Webmate: A personal agent for browsing and searching," in *Proc. 2nd Int. Conf. Auton. Agents*, K. P. Sycara and M. Wooldridge, Eds., May 1998, pp. 132–139.

[10] S. Clarke and P. Willett, "Estimating the recall performance of search engines," *ASLIB Proc.*, vol. 49, no. 7, pp. 184–189, 1997.

[11] A. Cockburn and B. McKenzie, "What do users do? An empirical analysis of web use," *Int. J. Human Comput. Studies*, vol. 54, no. 6, pp. 903–922, Jun. 2001.

[12] G. Cordasco, V. Scanaro, and C. Vitolo, "Architecture of a P2P distributes adaptive directory," in *Proc. WWW*, 2004, pp. 282–283.

[13] R. Cuha, R. McCool, and E. Miller, "Semantic search," in *Proc. World Wide Web Conf.*, 2003, vol. 154, no. 3, p. 477.

[14] J. Dean and M. Henzinger, "Finding related pages in the world wide web," in *Proc. World Wide Web Conf.*, 1999, pp. 1467–1479.

[15] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power law relationships of the internet topology," in *Proc. ACM SIGCOMM*, 1999, pp. 251–262.

[16] S. Golder and B. Huberman, "The structure of collaborative tagging systems," *J. Inf. Sci.*, vol. 32, no. 2, pp. 198–208, 2006.

[17] L. Gravano and H. Garcia-Molina, "Merging ranks from heterogeneous internet sources," in *Proc. 23rd VLDB Conf.*, 1997, pp. 196–205.

[18] T. Haveliwala, "Topic sensitive page rank," in *Proc. World Wide Web Conf.*, 2002, pp. 517–526.

[19] H. Heaps, *Information Retrieval—Computational and Theoretical Aspects*. New York: Academic, 1978.

[20] P. Heymann, G. Koutrika, and H. Garcia-Molina, "Can social bookmarking improve web search?" in *Proc. Int. Conf. Web Search Web Data Mining*, 2008, pp. 195–206.

[21] B. Huberman, P. Pirolli, J. Pitkow, and R. Lukose, "Strong regularities in world wide web surfing," *Science*, vol. 280, no. 5360, pp. 95–97, Apr. 1998.

[22] G. Jeh and J. Widom, "Scaling personalized web search," in *Proc. World Wide Web Conf.*, 2003, pp. 271–279.

[23] J. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604–632, Sep. 1999.

[24] H. Lieberman, "Letizia: An agent that assists web browsing," in *Proc. 14th Int. Joint Conf. Artif. Intell.*, Montreal, QC, Canada, 1995, pp. 924–929.

[25] A. Mathes, "Folksonomies—Cooperative classification and communication through shared metadata," Dec. 2004, [Online]. Available: http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html

[26] M. Marchiori, "The quest for correct information on the web: Hyper search engines," in *Proc. 6th Int. WWW Conf.*, Santa Clara, CA, Apr. 7–11, 1997, pp. 1225–1235.

[27] A. Papagelis and C. Zaroliagis, "Searching the web through user information spaces," in *Proc. WISE*, vol. 3806, *Lecture Notes in Computer Science*, 2005, pp. 611–612, Springer-Verlag: Berlin, Germany.

[28] A. Papagelis and C. Zaroliagis, "Searchius: A collaborative search engine," in *Proc. 8th Int. Conf. Current Trends in Computer Science—ENC*, 2007, pp. 88–98.

[29] M. Richardons and P. Domingos, *The Intelligent Surfer: Probabilistic Combination of Link and Content Information in Page Rank*, vol. 14. Cambridge, MA: MIT Press, 2002.

[30] E. Spertus, "ParaSite: Mining structural information on the web," in *Proc. 6th Int. WWW Conf.*, Santa Clara, CA, Apr. 7–11, 1997, pp. 1205–1215.

[31] A. Spink, B. Jansen, C. Blakely, and S. Koshman, "A study of results overlap and uniqueness among major web search engines," *Inf. Process. Manage.*, vol. 42, no. 5, pp. 1379–1391, Sep. 2006.

[32] L. Tauscher and S. Greenberg, "Revisitation patterns in world wide web navigation," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 1997, pp. 399–406.

[33] C. Thomas and G. Fischer, "Using agents to personalize the web," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 1997, pp. 53–60.

[34] R. Weiss, B. Velez, M. Sheldon, C. Manprempre, P. Szilagyi, A. Duda, and D. Gifford, "HyPursuit: A hierarchical network search engine that exploits content-link hypertext clustering," in *Proc. 7th ACM Conf. Hypertext*, New York, 1996, pp. 180–193.

[35] Y. Yanbe, A. Jatowt, S. Nakamura, and K. Tanaka, "Can social bookmarking enhance search in the web?," in *Proc. 7th ACM/IEEE-CS Joint Conf. Digit. Libraries*, 2007, pp. 107–116.

[36] L. Zhang, X. Li, and B. Liu, "Emergent semantics from folksonomies: A quantitative study," *J. Data Semantics*, vol. VI, pp. 168–186, 2006.

[37] [Online]. Available: http://www.dmoz.com

[38] [Online]. Available: http://www.ikeepbookmarks.com

[39] [Online]. Available: http://wordlist.sourceforge.net

Athanasios Papagelis received the Ph.D. degree from the Department of Computer Engineering and Informatics in the University of Patras, Patras, Greece.

He has been a Researcher at Computer Technology Institute (CTI—http://www.cti.gr/en/), a Senior Software Architect for Ahead Relationship Mediators, and the Technical Director and Co-founder of Epignosis, the company behind the open-source e-learning system eFront. His research interest include machine learning, information retrieval, text analysis, collaborative search, social navigation, and e-learning. His extended profile can be reached at: http://www.linkedin.com/in/papagelis.

Christos Zaroliagis (M'83) received the Ph.D. degree in computer science from the University of Patras, Patras, Greece, in 1991.

He is currently a Full Professor in the Department of Computer Engineering and Informatics, University of Patras, Patras, and a Senior Researcher of CTI, Patras, Greece. He held previous positions at the Max-Planck-Institute for Computer Science, Saarbruecken, Germany, and in the Department of Computer Science, King's College, University of London, London, U.K. He is the head of the ALGO-CUR Research Group specializing in the areas of efficient algorithms and data structures, algorithm engineering, parallel and distributed computing, combinatorial and large-scale optimization, multi-objective optimization, robust and online optimization, cryptography and information security, and web searching and applications. He has extensively published in major international journals and conferences, and has received numerous invitations from international conferences and leading academic institutions to give talks about his work. He is Editor of the Journal of Discrete Algorithms (Elsevier), and he has served in a few dozens of Program/Organization Committees of leading international computer science conferences. He has participated as a Coordinator and/or Key Researcher in numerous research projects funded by EC or by the Greek State. He has also served as a scientific consultant in the public and private sectors. Since 2000, he is the Director of the Laboratory for Computing (Computer Centre) at the Department of Computer Engineering and Informatics, which provides ICT infrastructure and services to more than 3500 users and which hosts one of the Grid Nodes of the Hellas Grid Network.

Dr. Zaroliagis is a member of ACM, EATCS, and IEEE.