Searchius: A Collaborative Search Engine *

Athanasios Papagelis
Dept of Computer Engineering and Informatics
25600, Patras, Greece
papagel@ceid.upatras.gr

Christos Zaroliagis
R.A. Computer Technology Institute, and
Dept of Computer Engineering and Informatics
25600, Patras, Greece
zaro@ceid.upatras.gr

Abstract

Searchius is a collaborative search engine that produces search results based solely on user provided web-related data. We discuss the architecture of this system and how it compares to current state-of-the-art search engines. We show that the global users' preference over pages can be efficiently used as a metric of page quality, and that the inherent organization of the collected data can be used to discover related URLs. We also conduct an extensive experimental study, based on the web related data of 36483 users, to analyze the qualitative and quantitative characteristics of user collected URL collections, to investigate how well the users URL collections cover the web and discover the characteristics that affect the quality of the search results under the proposed setting.

1 Introduction

In this work we propose a new framework for representing the web that does not rely on the exploitation of the web-topology. Our approach starts from the user and views the web as an aggregated collection of URLs, collected and semi-organized by individuals inside their information spaces. URLs can be explicitly collected (e.g., bookmarks) or implicitly collected (e.g., web-browsing history). 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.

We specifically concentrate on the qualitative and quantitative analysis of collections of explicitly collected URLs (bookmarks collections) and later on we discuss how our analysis can be extended to include implicit collections of URLs. Our approach is based on the observation that the web users act as small crawlers seeking information on the web using various media, which they subsequently store and organize into tree-like structures inside their information spaces.

One might argue that people do not collect URLs or that they do not organize them in any reasonable manner. However, as discussed in [1] and we experimentally demonstrate in Section 5, there are specific patterns that describe the way people collect, organize and express preference to URLs and there is no concrete reason to believe that these patterns will change any time soon. Furthermore, our analysis can be extended to implicit collections of URLs that constitute a dependable source of user specific data since people will always produce such implicit streams of web pages when interacting with the web.

The proposed framework has been materialized into a collaborative search engine, named *Searchius*, that produces search results by strictly collecting and analyzing URL collections. Conceptually, Searchius can be positioned somewhere between search engines and web catalogs. Searchius can be easily expanded and updated in an ad-hoc manner through asynchronous connections initiated by end-users. It can overcome shortcomings of algorithms based on link analysis (e.g., web islands), where information unreferenced by other sites is not being indexed. Moreover, Searchius is not capital intensive, since it concentrates on a small portion of the data that typical search engines collect and analyze. However, this portion of data is of the highest interest for the users. To order pages by importance, Searchius uses an aggregation function based on the preference to pages by different users, thus avoiding the expensive iterative procedure of PageRank. Finally, the way people organize their bookmarks can be used to segment the URL space to relative sub-spaces. This property can be exploited to provide efficient solutions to other applications, including the construction of web catalogs and finding related URLs.

*This work was partially supported by the Future and Emerging Technologies Unit of EC (IST priority – 6th FP), under contract no. IST-2002-001907 (integrated project DELIS) and the Action PYTHAGORAS of the Operational Programme for Educational & Vocational Training II, with matching funds from the European Social Fund and the Greek Ministry of Education.

In addition to the above, this paper makes the following contributions:

1. We demonstrate via an extensive experimental study the qualitative and quantitative characteristics of user specific 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 compare our hybrid search engine with a traditional search-engine using the overlap in search results as an indication of quality. Although the overlap is not a trusty indication – the commercial search engines produce a small overlap on results – the fact that there is *some* consistent overlap is a good indication that the proposed system can find and order sites adequately.

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 [5, 6, 15]. An initial experimental study on the characteristics of user bookmarks collection can be found in [1]. Another work under the same context can be found in [7] where the architecture of a 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, web history, last search queries, last page clicks, time spent on a web page) for personalization and improved searching accuracy are discussed in [3, 8, 11, 14]. Known web sites that allow people to upload, share and search public-bookmarks are IKeepBookmarks (ikeepbookmarks.com), FURL (furl.ney), Simpy (http://www.simpy.com) and BackFlip (backflip.com). FURL allows people to store online all content of pages they visit and uses a similar to our approach technique to order pages using the notion of page popularity. GiveALink (givealink.com) is a public site where people can donate their bookmarks to the Web community for research reasons.

The paper is organized as follows. Section 2 presents an architectural overview of our system. In Section 3 we describe how our analysis can be extended to implicit collections of URLs. Section 4 describes an additional application of the collected data for finding related URLs. In Section 5 we discuss experimental results using real world data and comment on the way that users collect and organize URLs.

2 System Description

Main Concepts. The main principle of our approach is that users act as small, 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. 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.

There are several techniques that explore the utility of personal information spaces for the construction of personal profiles and the consequent exploitation in favor of the user. 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.

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.

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. At its simplest version, a simple aggregation of different users that have stored the URL is enough. Each 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 URL. We call this method of ranking pages *UsersRank*, since it is controlled by the collective preference of the users.

System Architecture. Figure 1 presents a high level overview of Searchius architecture. The interested reader can find a working prototype of Searchius at: `http://searchius.ceid.upatras.gr`.

Data is collected via the *Communicator*, a small executable that can be freely installed to the end-user's system. Communicator 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*) is transferred and stored to the *Data Repository*. The submitted data can include metadata about the user like his city, country, language and

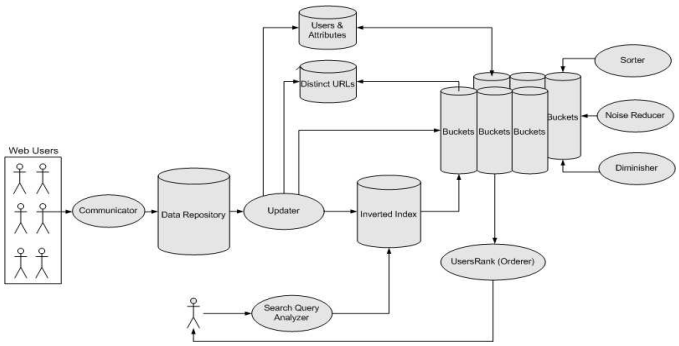

Figure 1. High Level Architecture of Searchius

interests. These data can be later used to personalize the search results. Whenever the system finds a batch of data that has an already existing User-Key it replaces the old data with the new one.

The *Updater* checks at specific time intervals for unsettled data collections. For each such collection it performs a number of database updating tasks: (a) It adds the user key and his metadata on the *Users-Attributes* table or replaces already existing records for that specific user (b) It checks for not yet discovered URLs and adds them to the *Distinct URLs* table (c) It parses the URLs, their titles and semantic tags and adds 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 (d) It adds all distinct connections between word-IDs, URLs and users to the*Bucket* tables. For each connection it also stores a *value* that is later used from the *UsersRank* module to order pages. In Buckets we store a detailed decomposition of how words, URLs and users interact with each other. 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 delete data from Buckets. When a user poses a query, the *Search Query Analyzer* parses and analyzes it. For each non trivial 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. If the query includes more than one search word, it creates different lists of ordered URLs and gets their intersection to provide the final relative page ordering. 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 UsersRank scope to the buckets that belong to users from that set.

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.

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 very 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 to improve and update the database and that we generally collect only a small portion of web data.

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 most often multiplied.

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 multiply the results with the UsersRank of each page in the collection to produce the final ordering. We do not store the body text, 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).

Updating of Data. To update their database, current search engines have to constantly crawl the web. This technique is not applicable n our context since updating can be initiated only by end users. Updating is crucial, because web pages tend to become obsolete through 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 through time allowing more room for fresh data.

3 Explicit vs Implicit collections of URLs

Explicit collections of URLs refer to bookmark collections while implicit collections of URLs refer to users browsing history. Both collections share similar characteristics but at the same time they have important structural differences.

The main common characteristic is that both of them can be used as preference votes from users towards URLs. Implicit collections of URLs carry also the relative preference for URLs as is represented by the frequency of visiting specific pages. Explicit collections of URLs can carry important semantic and structural information through the way that people organize their URLs.

The quality of page ordering will be benefited by implicit collection of URLs 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. On the contrary, bookmark collections may be somewhat outdated. Second, the frequency of visiting specific pages gives a much better indication of our relative preference for specific sites than the bookmark collections. 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 can not be used directly for other applications that rely on the exploitation of the URLs structure like the related-URLs discovery application as presented at section 4.

We now formally describe the *UsersRank of a particular page* using explicit collections of URLs. Let us assume that our database consists of a set U of $x = |U|$ users, each having a URLs set U_i, where $i \in \{1, 2, ..., x\}$. 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_{ij} being 1 or 0 depending on whether this specific user has the URL B_j or not. The *UsersRank of a particular page* 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 j-th position of their URL set; i.e., $R_U(j) = \sum_{k=1}^{x} U_{kj}$.

Extending Searchius to work with implicit collections of URLs 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 N number of URLs. These are the last N pages that the user has visited. We need to slightly modify our initial UsersRank definition 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 URLs set (the users browsing history) U_i, where $i \in \{1, 2, ..., x\}$. Also, let F_{ij} be the number of occurrences of page j in U_i. Then, the preference of user i for any page j can be formulated as: $U_{ij} = F_{ij}/N$. Again, each U_i can be represented by a vector of size b with each element U_{ij} being the preference of user i over page j and the UsersRank of a particular page j remains $R_U(j) = \sum_{k=1}^{x} U_{kj}$.

4 Discovering Related Pages

In this section we elaborate on a simple algorithm to discover related pages based on the structure of the collected data. In section 5, we will see how the new algorithm compares with the results of other state-of-the-art search engines.

Current methods for finding related pages use link analysis as proposed in [9]. To achieve the same goal in Searchius, we exploit characteristics of the bookmarks 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 and returns the URLs that occur more often on that subspace. In other words, it is a modified 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.

Assume that B_k is an initializing URL for which we want to find related pages. Let the *father concept* of a URL be the tag of its parent folder in the tree structure of its bookmark set. Let Z_{ij} be the father concept of bookmark 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, $Y = \{\text{user } i : U_{ik} = 1 \wedge Z_{ik} \neq 0\}$. Finally, let $relevanceWithB_k$ be an integer

b-vector, which uses URLs as indices. For a URL (bookmark) 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 $relevanceWithB_k$;

5 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 produced partly from user submitted data at Searchius' web site and partly from collections of public URL sets taken from [17]. We conducted experiments towards three major goals aiming to:

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. Compare the overlap on search results obtained from Searchius with the results of a current state-of-the-art search engine. Assuming that our approach is reasonable we expect some overlap between our results and those of other search engines.

5.1 Characteristics of URL collections

Table 1 presents the basic characteristics of the collected URL sets. *Distinct Search Terms* refers to the different search 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 different descriptors people use to organize their URLs. A single folder name can be used to extend the IR score of many pages regarding many relative-to-the-folder-name queries. Folder names are treated as semantic tags over pages and are the equivalent of anchor links. 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 1, the vast majority of URLs are at least under one folder, while about 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. Figure 2 plots the URLs per user, the distribution of folders per user and 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 [2]. 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 [10] and user's page access behavior [13]. Table 2 presents these results and the produced coefficients.

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

Table 1. Database characteristics

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

Table 2. Power Law coefficients for several database metrics

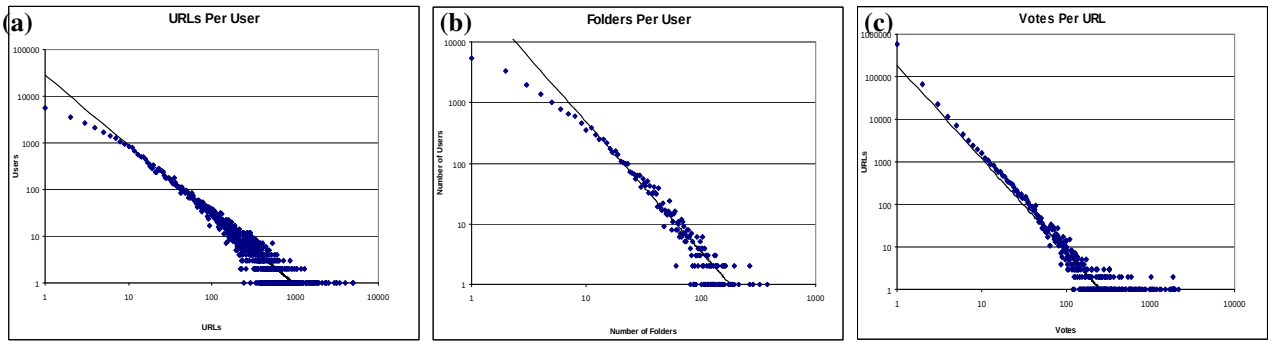

Figure 2. (a)Log-log distribution of URLs per user (b)Log-log distribution of folders per user (c)Log-log distribution of votes per URL (number of different users that have stored a URL

5.2 Identifying the growth rates of data

This set of experiments is based on the analysis of the database characteristics during the database construction phase. Prior initiating this experiment we shuffled the URL sets and then we re-constructed 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. Figure 3(a) illustrates the distinct URLs and keywords growth, while Table 3 presents the coefficients of their functions found using the Trust-Region non-linear least-square algorithm [2]. Both growths follow the Heaps' Law [12][1]; 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.

As the function at Table 3 suggests, in order to build a URL database that has the quantity of information provided by Google (about 8 billion pages), we would need around 1.6 billion users. However, as our next experiment shows, important sites are discovered *early* during the database construction phase and thus, they are included in even relatively small databases. The same stands for keywords as well; common keywords are discovered earlier during the database construction phase. The next set of experiments sheds light to the manner that the users preference is distributed among sites. Again, while reconstructing the database we measured at given user intervals the UsersRank that several sites have accumulated. Figure 4 illustrates the

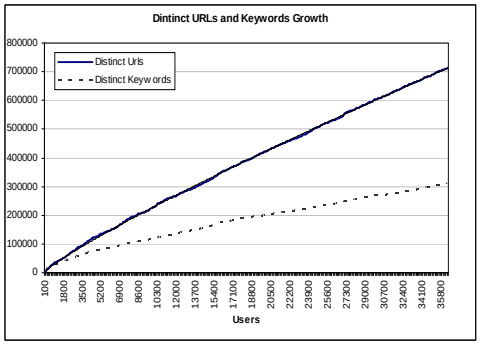

Figure 3. Heaps' Law distributions for the number of distinct URLs and Keywords while adding new users to the database

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

Table 3. Heaps' Law coefficients for URLs and keywords growth

UsersRank 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 of sites. Depending on the site, this linear relationship has a different slope that characterize the site. Table 4 summarizes the discovered approximation functions for the given sites. We witnessed similar linear relationships for the majority of URLs that were included in our database.

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

[1] In linguistics, Heaps' Law is an empirical law which describes the portion of a vocabulary which is represented by an instance document (or set of instance documents) consisting of words from the vocabulary. This can be formulated as: $V_r(n) = Kn^b$ where V_r is the subset of the vocabulary V represented by the instance text of size n. K and b are free parameters determined empirically.

its relative value and that it stabilizes as soon as we have enough data to describe it.

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

Table 4. UsersRank grow coefficients for several sites

5.3 Comparison with search engines

Our third set of experiments aim at investigating the quality of actual search results for our prototype search engine. As a metric of quality we use the results overlap between our approach and Google for a wide spectrum of search queries. Our intension is not to directly compare Searchius with Google - something like that would be unappropriate taken into account the size of our database - but rather we use the results overlap as a metric of quality. The idea is that if our approach is worthy it should produce *consistent* overlaps with the results of other established search engines. At the same time it should not mimic the behavior of other systems but rather produce, at some extend, its own unique and interesting results.

We decided to concentrate on broad collections 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 a collection of 41234 generic English words (set-A) taken from [18]. We also used a collection of 190 first level domains (set-B) - produced using a certain methodology that we describe later on- to compare the results of our related-URLs algorithm with the results of Google.

To perform the experiments we implemented an automatic comparison tool. This tool is feeded with set of queries that it performs on both search engines. Afterwards, it checks for common returns on the first 20 results. We opted to use this method because from a user perspective the first 20 results are typically what one actually sees and use. 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 feeded with sets of URLs for which we want to find related URLs. Again, this tool checks for common URLs on the first 20 results.

Experiments with set-A. Set-A is a broad-scope, single-words set (41234 words) taken from [18]. Through its use we aim to get strong general indications of what to expect from Searchius regarding results quantity and ordering quality.

An interesting depiction between quality of results and total number of relative results is presented at Figure 5(a). This figure suggests that when we have adequate relative data for a specific query on Searchius then we can expect some consistent overlap with the top results of Google. Figure 5(b) illustrates 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 return 4000 results on average, the overlap between Searchius and Google top results was around 16%. This result is an indication that with sufficient data we can expect some consistent overlap between our search engine and Google for any generic query. Some overlap is a good indication since it supports our view that the proposed methodology does work. On the other hand, excessive overlap is a bad indication since it would mean that our system mimics the behavior of other systems and that the data acquisition process is search engine oriented. Recent work [16] on the amount of overlap between commercial search engines supports the argument that there is a *small* but *existing* percentage of overlap among their results.

Experiments with set-B. Our final experiment compares the related URL results obtained from Searchius and Google. In order to create set-B we adopted the following methodology. Initially, we excluded all URLs that did not have their URLs 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 results. For this set of URLs we got an average overlap of 32%.

References

[1] D.Abrams, R.Baecker, and M.Chignell. Information Archiving with Bookmarks: Personal Web Space Construction and Organization. In Proceedings of CHI'98, 41-48, 1998.

[2] M.Branch, T.Coleman, and Y.Li. A Subspace, Interior, and Conjugate Gradient Method for Large-Scale Bound-

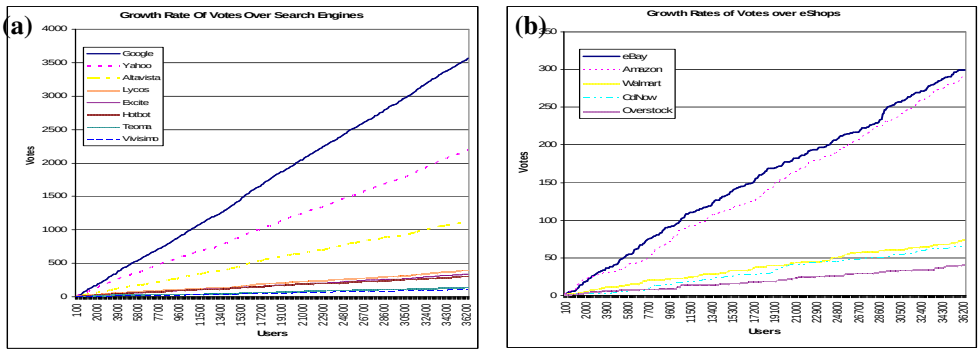

Figure 4. (a)Growth of UsersRank for several search engines (b)Growth of UsersRank for several eShops

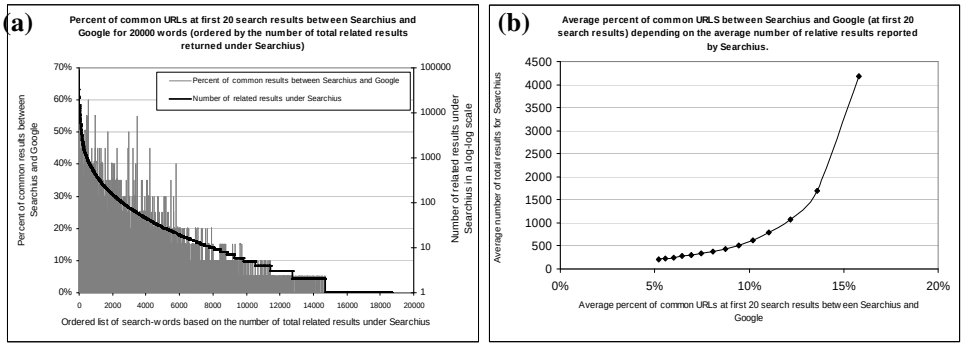

Figure 5. (a)Search results overlap between Searchius and Google for 20000 one-word queries (b)Prediction of search results overlap between Searchius and Google depending on the total number of Searchius results

Constrained Minimization Problems. SIAM Journal on Scientific Computing, Vol. 21, Number 1, pp. 1-23, 1999.

[3] S.Brin, R.Motwani, L.Page, and T.Winograd. What can you do with a web in your pocket. In Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 1998.

[4] S.Brin and L.Page. The anatomy of a large-scale hypertextual web search engine. In Proc. *World Wide Web Conference*, 1998.

[5] L.Catledge, and J.Pitkow. Characterizing browsing strategies in the World-Wide Web. Comp. Networks ISDN Systm. 27, 1065-1073,1995.

[6] A.Cockburn, and B.McKenzie. What do users do? An empirical analysis of Web use, Int. J. Human Computer Studies 54, 903-922,2002.

[7] G.Cordasco, V.Scanaro, and C.Vitolo. Architecture of a P2P Distributes Adaptive Directory. In Proceedings of WWW2004, 282-283, 2004.

[8] R.Cuha, R.McCool, and E.Miller. Semantic Search. In Proc. *World Wide Web Conference*, 2003.

[9] J.Dean and M.Henzinger. Finding related pages in the world wide web. In Proc. *World Wide Web Conference*, 1999.

[10] M.Faloutsos, P. Faloutsos, and C. Faloutsos. On power law relationships of the internet topology. In ACM SIGCOMM, 1999.

[11] T.Haveliwala. Topic Sensitive Page Rank. In Proc. *World Wide Web Conference*, 2002.

[12] H.Heaps. Information Retrieval - Computational and Theoretical Aspects. Academic Press, 1978.

[13] B.Huberman, P.Pirolli, J.Pitkow, and R.Lukose. Strong regularities in World Wide Web Surfing. Science, 280:95-97, 1998.

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

[15] L.Tauscher, and S.Greenberg. Revisitation patterns in World Wide Web navigation. In Proc. Conf. on Human Factors in Computing Systems CHI'97, 97-137, 1997.

[16] `http://comparesearchengines.dogpile.com/OverlapAnalysis.pdf`

[17] `http://www.ikeepbookmarks.com`

[18] `http://wordlist.sourceforge.net`