SearchPad: Explicit Capture of Search Context to Support Web Search

Krishna Bharat
Compaq, Systems Research Center, Palo Alto, CA 94301
(Current Address: Google Inc., 2400 Bayshore Parkway, Mountain View, CA 94043)
krishna@google.com

Abstract

Experienced users who query search engines have a complex behavior. They explore many topics in parallel, experiment with query variations, consult multiple search engines, and gather information over many sessions. In the process they need to keep track of search context -- namely useful queries and promising result links, which can be hard. We present an extension to search engines called SearchPad that makes it possible to keep track of "search context" explicitly. We describe an efficient implementation of this idea deployed on four search engines: AltaVista, Excite, Google and Hotbot. Our design of SearchPad has several desirable properties: (i) portability across all major platforms and browsers, (ii) instant start requiring no code download or special actions on the part of the user, (iii) no server side storage, and (iv) no added client-server communication overhead. An added benefit is that it allows search services to collect valuable relevance information about the results shown to the user. In the context of each query SearchPad can log the actions taken by the user, and in particular record the links that were considered relevant by the user in the context of the query. The service was tested in a multi-platform environment with over 150 users for 4 months and found to be usable and helpful. We discovered that the ability to maintain search context explicitly seems to affect the way people search. Repeat SearchPad users looked at more search results than is typical on the web, suggesting that availability of search context may partially compensate for non relevant pages in the ranking.

Keywords: Search engines, search context, queries, bookmarking, data collection, relevance information, Javascript, cookies.

1. Introduction

As users gain expertise in searching on the WWW they begin to make use of the wide choice in search services available online. However, as they cast a wider net to locate the information they seek, they start to employ a more elaborate and complex search process. Experienced users searching on the web seem to have the following behavior:

1. They search on many unrelated topics in parallel, often with many browser windows.
2. A given search for information may extend over many sessions. They may terminate and restart the browser between sessions.
3. For each information need they use many queries, often by a process of query refinement. Power users may employ variants of queries that worked well in other contexts.
4. They may try the same query on many search services. (By a search service we mean search engines such as AltaVista [AltaVista] and Google [Google], meta search engines such as AskJeeves [AskJeeves] and Metacrawler [MetaCrawler], and resource directories such as Yahoo! [Yahoo] and Open Directory)
5. Some users may look at more than one search result page.
6. When they do find a useful result, they are often unsure whether the information they have found is the best available or they should search further.

The trouble with the above behavior is that the user needs to carry around a lot of contextual information over time and there is no convenient way to record it or make it explicit. Specifically, they need to remember URLs of potentially useful results as they look for more results, and remember useful queries over time. Both of these can be hard to memorize. Saving information to the browser's collection of bookmarks is one potential solution. However, there are several reasons why this is not convenient:

1. Most users would be reluctant to contaminate their bookmark list with tentative leads. The list of bookmarks is intended to store high quality web pages that they wish to remember for a long time, and not intermediate results.
2. To remember a query one would need to bookmark a search result page. However this provides no way to run the same query on a different search service.
3. As result pages and tentative results from many queries get bookmarked they become interleaved and hard to distinguish. One solution to avoid clutter would be create bookmark folders for each information need in advance, and bookmark each result and query into the appropriate folder. However, this takes too much effort on the part of the user.

In this paper we describe an extension to the search result page called SearchPad, which helps users search more effectively by explicitly maintaining their "search context." By search context, we mean queries recently deployed by the user, along with hyperlinks of result pages the user visited and/or liked in the context of each query. SearchPad is very similar to a bookmarks window except that it is search specific and maintains a relationship between queries and links the user would like to keep track of (which we call leads). As with bookmarks, clicking on a saved lead causes the corresponding page to be loaded in the browser. Saved queries can be replayed on other search engines.

To make SearchPad usable by a large audience we had two design goals which made the implementation of the system challenging:

- To appeal to the widest possible audience we wanted an implementation that was portable (i.e., worked on all browsers and platforms) and did not impose any overhead on users (loaded quickly and transparently). The rationale for the latter condition was the feeling that many users would be unwilling to use a search service which required them to first explicitly download a modified client or a plug-in. Indeed, on some platforms the delay of several tens of seconds in starting the Java Virtual Machine makes even Java a poor choice for implementation.
- A second design goal was to not impose any extra storage or communication overhead on the search service. This greatly simplifies the integration of SearchPad support into new search services. However, this implies that all control and storage happens at the user end.

Our implementation of SearchPad uses cookies and a subset of Javascript that is known to work on all platforms. It loads instantly with the first result page obtained from the search service, and communicates with the server on an as needed basis. To simulate the behavior of actual services providing support for SearchPad, we implemented a proxy that provides access to 4 major search engines: AltaVista, Excite, Google and HotBot. The role of the proxy was to transform result pages streaming through to make them appear as they would if they were implementing a service such as SearchPad. Note that other pages such as result pages are not fetched through the proxy. They are fetched directly from the WWW. The role of the proxy is only to simulate how search engines would behave if they supported SearchPad.
As we shall describe, our implementation provides an additional service. It allows the search service to collect query specific result relevance and usage data. Specifically, SearchPad can log for each client:

1. Queries that were issued
2. Result pages viewed for each query
3. Result hyperlinks considered relevant for each query
4. The order in which result pages were viewed
5. The time spent viewing the result
6. Whether a result hyperlink considered relevant was actually viewed by the user

Such information is valuable to search providers. It can be used to statistically compare two ranking algorithms and find out which one is better. Similarly, it can be used to compare two search services. It can also be used to discover the most relevant pages for popular queries, which in turn can be used to improved results for those queries in the future.

Collection of usage data raises concerns about privacy and the author strongly supports the privacy of users on the web. However, the major vulnerability from the user's point of view is having search services know about their interests. Unfortunately, this is already revealed by the query. The information we collect, namely the results they viewed and found useful, reveals more about the quality of the pages returned than about the user. Thus we argue that this is not a further breach of privacy. In any case there other search companies on the web, notably Direct Hit [DirectHit], with a business model based on collecting data on the pages that user's look at. They count "click-throughs" (i.e., the number of times users click on a particular link) received by result links for popular queries and reorder results based on perceived popularity. We believe that the information we can collect is superior to Direct Hit's data, because we discover the results people actually liked -- not just the results they clicked on. Even when a query finds no useful results, users tend to click on a few results per query to understand what happened, which can contaminate the click-through log. With our scheme the data collected is purer. Also, collecting click-throughs as done previously imposes an overhead both on the server and the user. In Direct Hit's scheme click-throughs are trapped by redirecting result accesses through a web server that logs the data and then issues a redirect to the actual result page. With our scheme all logging is done at the client without an extra HTTP access.

2. Interaction with SearchPad

We present a walk-through to illustrate the user's interaction with SearchPad. In our implementation, the user accesses AltaVista, Excite, Google and Hotbot through special URLs that route communication with the engines through the SearchPad proxy.

Figure 1 shows an AltaVista page transformed by the proxy. The "SearchPad" button at the top left brings up the SearchPad agent (Figure 3), allowing the user access any previously marked queries and leads. Each result on the AltaVista page has a blue "Mark" button associated with it. Clicking on this button causes the corresponding link to be added to SearchPad, along with the corresponding query. If the query already exists the link is merged into the existing set of links. Links added to SearchPad are called "leads." Note that marking is a cheap operation, and involves only a local transfer of data from the result page to the SearchPad agent. No network communication occurs and hence no delay. This is illustrated in Figures 2 and 3.
Figure 2 shows the second *AltaVista* result for the query: genetic engineering, which has just been visited by the user. All visits to result page and time spent therein are logged by *SearchPad* as part of its data collection process. Also, on return from a result page, the blue *Mark* button for the just-visited result link turns red as in Figure 2 (hard to see in grayscale). The color change is an invitation for the user to mark the link. Also, it makes the result easier to spot, increasing the likelihood of the user marking the lead if they liked it.
Figure 3 shows the SearchPad window with a list of 3 queries bookmarked, and under each query a list of leads. SearchPad is merely a web page rendered by Javascript code, and appears within an independent browser window. Each query has an open/close triangular toggle to control the visibility of leads under it. E.g., clicking on an open toggle causes it to close. The last marked query ('genetic engineering') is at the top of the SearchPad heap and hence most visible. Queries in SearchPad are maintained in a most-recently-accessed order to keep pace with the user's varying interests. For each marked lead the title is shown, hyperlinked to the corresponding web page. To conserve space only the hostname is shown after it. SearchPad is designed to have the form factor of a small notepad -- small yet useful for recording essential information during a search.

![SearchPad](http://www9.org/w9cdrom/173/173.html)

**Figure 3: The SearchPad Helper with a new query: "genetic engineering" and a new lead**

Each query has a circular selector in front of it to support query selection. To send a query to a search engine the user would first select the query and click on the search engine. If they selected the most recent query, 'genetic engineering', and clicked on Google, they would get the result set shown in figure 4.
Figure 4: Google results for the replayed query: 'genetic engineering'

In Figure 5 the user has subsequently marked the lead labeled 'MelissaVirus.com: The very latest Melissa Virus information' for the (repeat) query "melissa virus". This moves "melissa virus" to the top of the list of SearchPad queries and adds the new lead to the end of the list of leads for the query.
SearchPad also has an Edit Mode (see Figure 6) to support changes to the stored data. This is because, although the browser may shutdown and the machine get rebooted, the information stored in SearchPad is permanent. Hence, the user may periodically want to delete some leads or queries to free up space. Also they might want to merge the leads classified under various related queries into a single meaningful query. In Edit Mode, SearchPad is still fully functional, except that it provides extra buttons to edit its state. The cross ("X") marks are buttons to delete the query or lead they are associated with. Queries can be renamed by clicking on Rename, which brings up a dialog to enter the new query. If the new query matches another existing query the leads in the two queries are merged. The old query is discarded.
3. Implementation

In this section we describe the implementation of SearchPad.

Since a design goal was to not require any extra storage at the server or impose any communication overhead for marking, all storage and computation is moved to the client. Consequently, we implemented SearchPad as an HTML document containing embedded code in Javascript [Javascript] (VB Script [VBScript] could have been used as well). Result pages are extended by embedding code in Javascript as well. When the user marks a link the associated scripting code communicates the link’s URL and associated information to a corresponding piece of code in SearchPad. The code within SearchPad then updates its display showing the new link.

This approach is faced with the following problem. Embedded scripts are constrained by the browser both in terms of access (i.e., limited access to other windows) and storage (no access to the filesystem) in the normal mode of operation. In some web browsers, the embedded scripts can request the user for more access to the web browser’s state. Nonetheless this is not useful because many users will refuse such a request, since it might represent a security
risk. Thus, embedded scripts face many restrictions. We describe next how these may be overcome.

We use cookies (a mechanism for host-specific persistent client-side storage) both for communication between the result page and SearchPad, and for persistent storage. (See RFC 2109 [RFC2109] and Netscape’s documentation on cookies [NetscapeCookies]) A client such as Netscape’s Navigator which implements RFC 2109, supports a limited amount of client-side storage in the form of cookies. Each cookie holds 4 Kb of text data, and each host a user visits will be allowed at least 20 cookies. Such cookies are persistent and save their state on the user’s hard disk. This allows SearchPad to remember marked leads across web browser sessions.

Javascript was ideal for our implementation because it allowed cookies to be read and written from within the browser. A restricted form of cookies sharing is possible between Javascript instances. This allows code on the result page to pass messages to code in SearchPad. Javascript is single threaded across the entire browser. This gives us mutual exclusion and simplifies the design. Also, Javascript has support for timer driven callbacks which is needed to implement polling behavior within SearchPad.

The search service (or a proxy server through which the search service is accessed) embeds a button (or equivalent device) within each search result to allow the user to “mark” the result as a lead. The button links to embedded code in JavaScript. The code is invoked when the button is clicked, and causes relevant information about the link and query to be written to a log maintained in a set of cookies, associated with the web site (known as the access log).

For example, the following are logged for each “mark” action:

- The query
- The Title, URL and rank of the result being marked
- The time at which the event occurred

Similarly, when a result’s hyperlink is clicked to view the result page, we log the same type of information in association with the “view” event. When the user returns to the page containing search results after viewing a result page, the “return” event is logged as well, with a timestamp. When a “return” event follows a “view” event, the time difference provides an estimate of the time spent viewing the result page.

All the information collected above resides in a set of cookies associated with the originating web site, and is available to scripts executing within other pages downloaded from the same site. In particular it is visible to SearchPad, which is an HTML document whose contents are dynamically generated by embedded Javascript. SearchPad polls the access logs every few seconds in order to respond to events.

All the data needed by SearchPad to display marked queries and leads to the user is maintained in a cookie access log. When the cookie access log is updated due to a new event which requires a change in SearchPad’s display, the SearchPad code initiates a "soft" reload. The soft reload operation fetches the cached web page corresponding to SearchPad from the browser cache and executes the code again. At this point the Javascript reads the cookies and redraws itself to reflect the new state. To initiate a reload, either the code in the result page

can signal SearchPad to notify that the state has changed, or SearchPad can periodically examine the cookie log to see if new leads have been added (as in our implementation). Changes to SearchPad's display due to interaction with SearchPad (e.g., open/close operations and mode change operations) are handled similarly. The Javascript event handler updates the visual state of SearchPad represented in the cookies, and initiates a soft reload.

Eventually, as events accumulate and leads are added, the storage available in the cookie access log will be exhausted. At this point either the user can be prevented from marking any more leads (unless some are deleted), or SearchPad can compress the data. We support a clever form of data compression to free up more space.

To compress data in the cookie access log, SearchPad does a "hard" reload of itself. This causes fresh copy of the SearchPad web page is fetched from the server ignoring the cache. The cookies comprising the cookie access log are configured so that they are transmitted to the web server every time SearchPad is reloaded over the net. Also, a fresh set of cookies are transmitted back from the server and overwrite the previous cookies. This is part of the standard RFC 2109 cookie exchange protocol. We use this to transfer activity log information to the server and also to reduce the data stored in SearchPad. Specifically:

1. All data in the event log that the server needs to keep for its data collection is logged at the server. The remaining logged data is cleared in the cookies.
2. The verbose information for each newly bookmarked lead is removed from the cookies. This is because the same information is already present at the server. Each lead is replaced by an identifier representing the URL (known as the URLID), based on the internal handle to the URL at the server.
3. However, the URLID is not intelligible to the user and unsuitable for presentation. Hence, the server dynamically generates a new version of the SearchPad web page in which the Javascript code is augmented with a lookup table mapping URLIDs to title and URL information, for all marked leads. This allows the same presentation to be given to the user as before compression. However, the bulk of the data is moved from the cookies to the SearchPad Javascript code. Since the mapping from URLIDs (server internal ids) to title and URL information is assumed to be available at the server, no extra storage is needed at the server to support the user base.

To ensure timely data collection at the server, SearchPad is configured to periodically hard reload itself, thus logging the user's activity periodically. Further, to avoid transmitting the cookies to the server during other communications, the cookies are configured so that they will be transmitted only when SearchPad is reloaded and not when result pages are fetched. We do this by associating SearchPad with a path that extends the path of result pages, as explained in RFC 2109. This has the effect of allowing SearchPad to read cookies set by result pages but not vice versa.

4. Experience

We conducted a trial of the SearchPad service at our research laboratory - Compaq, Systems Research Center, from May 6 - Sep 3, 1999. The service was available on the company intranet, but most of the usage was by the research staff of the Systems Research Center (about 50 people), and to a smaller extent by Compaq Research as a whole (about 150 people). Logs were collected in partially shrouded format so that queries themselves were unrecognizable, but hostnames and other details were preserved. Our logs show that accesses outside the research community did not contribute significantly to usage.

Table 1 summarizes the usage statistics for the 4 month period. This does not include...
accesses by the author for testing. The aim of the study was to understand if people would find our service useful. Although users were invited to use the system through internal advertising, no incentive was given to make them use it. Also, assurances were given that we would protect their privacy. Hence, we did not attempt to keep track of the results that were bookmarked. since within a small community such information might reveal more than it would on the Internet at large. Also, we would need a large user base to collect a statistically significant sample of usage information to make any relevance judgements.

### Table 1: Usage Statistics from a 4 Month Trial

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of Result Pages Viewed</td>
<td>2281</td>
</tr>
<tr>
<td>AltaVista</td>
<td>1352</td>
</tr>
<tr>
<td>Excite</td>
<td>148</td>
</tr>
<tr>
<td>Google</td>
<td>724</td>
</tr>
<tr>
<td>HotBot</td>
<td>57</td>
</tr>
<tr>
<td>Number of Distinct Accessing Hosts</td>
<td>178</td>
</tr>
<tr>
<td>Number of Distinct Queries</td>
<td>1133</td>
</tr>
<tr>
<td>Average # of Result Pages/Query</td>
<td>2.01</td>
</tr>
<tr>
<td>Average # of Result Pages/host</td>
<td>12.8</td>
</tr>
<tr>
<td>Percentage Accesses w SearchPad &quot;Docked&quot;</td>
<td>8%</td>
</tr>
</tbody>
</table>

The high usage of *AltaVista* may be biased by the fact that *AltaVista* was created by *Compaq Research*. The usage of the other engines can be taken to represent perceived value by our user base. In most cases each host in the log corresponds a distinct user. We were curious to see if usage patterns would change with the *SearchPad* model of searching. For example, we were curious if users look at more pages, since they now had the option of keeping track of temporary leads? The average number of result pages per query was 2.01, which is higher than previously reported (e.g., 1.39 was reported in a previous study by [Silverstein et al, 98]). Our number is somewhat diluted by the presence of casual users who used *SearchPad* marginally, possibly for test queries. Considering more seasoned users (users who used *SearchPad* to view more than 50 result pages) the number of result pages viewed per query is slightly higher = 2.15. We noticed a large number of single page views in the logs, even for seasoned users. A single result page view is often evidence of the fact that the user found the result they were looking for immediately (i.e., the ranking was good), or that they were disappointed with the query and formulated a better query. If we consider only cases in which users looked at more than one result page we find that the average page views per query is higher = 3.98. This suggests that having a tool to record search context may encourage users to explore result sets more deeply, and compensate for some non-relevant pages in the ranking.

The only interface design choice we tried to evaluate was the option of attaching *SearchPad* to the left of the results window, as an extra frame. This was done by clicking on the "SearchPad" button at the top left of the result page (see Figure 1). We call this "docking." Each result window could have a docked version of *SearchPad* potentially. Docking was hard to implement since it meant keeping several versions of *SearchPad* synchronized. However, the user study shows that only 8% of the users liked the docking option. This actually reduced to 5% for users with more than 50 result page views, suggesting that embedding *SearchPad* in
a frame is not convenient.

*SearchPad* was tested on Netscape versions 3 and higher on Unix, MacOS, and Windows 95/NT, and on Internet Explorer versions 4 and higher on Windows 95/NT, and found to work reliably.

5. Conclusions

In this paper we describe an extension to search engines to explicitly maintain user search context as they look for information, on many topics, using many search engines, and over many sessions. By search context we mean queries that were previously deployed and considered useful, and promising result links associated with each query. *SearchPad* is an agent that works collaboratively with result pages, and allows users to remember queries and associated leads in a convenient helper window. Unlike bookmarks, which correspond to the user's long-term memory of information, the leads in *SearchPad* constitute the user's short-term memory and represent work in progress. They tend be less valuable than bookmarks and are maintained only as long as the user's information need is current. Hence we perceive *SearchPad* as a complement to the browser's bookmarks facility.

*SearchPad* is implemented as a Javascript extension to the search results page. We demonstrated the generality of our design with an implementation that works on four major search engines. Our implementation is highly portable, requires no download or start-up delay, needs no storage at the browser and does not increase the communication overhead with the server. An added benefit is that *SearchPad* can record user actions on the search result page, and also discover which results are most valuable to users in the context of specific queries. This imposes less overhead and is qualitatively more useful than the information collected using the click-through tracking strategy of search engines such as *Direct Hit*.

The service was tested in a multi-platform environment with over 150 users for 4 months and found to be usable and helpful. It is possible that the ability to maintain search context explicitly affects the way people search. Repeat *SearchPad* users looked at more search results than reported previously. This suggests that explicit availability of search context might partially compensate for non relevant pages in the ranking.

References

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

[DirectHit]  
The Direct Hit Technology - A White Paper, *Direct Hit Inc.*,  
http://system.directhit.com/whitepaper.html

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

[AskJeeves]  
http://www.ask.com/

[Javascript]  
Javascript Reference, *Netscape*.  
http://developer.netscape.com/docs/manuals/communicator/jsref/contents.htm
Vitae

Krishna Bharat is a member of the research staff at Google Inc. in Mountain View, California. Formerly he was at Compaq Computer Corporation's Systems Research Center, which is where the research described here was done. His research interests include Web content discovery and retrieval, user interface issues in Web search and task automation, and relevance assessments on the Web. He received his Ph.D. in Computer Science from Georgia Institute of Technology in 1996, where he worked on tool and infrastructure support for building distributed user interface applications.