Personal Search Assistant: A Configurable Personal Meta Search Engine

P.R.Kaushik, Persistent Systems Private Ltd.,"Panini", 2A Senapati Bapat Road, Pune 411016, India.kaushik@pspl.co.in

Dr. K. Narayana Murthy, Department of CIS, University of Hyderabad,Hyderabad 500 046, India knmcs@uohyd.ernet.in

Abstract

This paper presents Personal search assistant (PSA) as a model for Meta searching on the World Wide Web. PSA differs from other conventional web search architectures by the fact that the collection phase works offline, updating a local repository instead of presenting results immediately to the user, constructing a concept database in the process. Typically using the PSA the user presents a query to the system and terminates the session. The PSA works in the background constructing and updating a virtual local web on the user system. This could take place immediately or gradually over a couple of days, depending upon how the system is configured. The local repository which is kept virtually independent of the remaining part of the system, can be investigated at any time to view results which have been updated or results from a previous search. The resultant setup is more alike Push Technology but the resultant information channels setup are more specific to user preferences and are not constrained by service providers in Channels. The personalize agent a part of the PSA works on constructing a user profile. The personalize agent gains from the additional information available to it from user browsing patterns. The agent makes suggestions to the user and looks for confirmation before taking decisions and to reinforce its convictions about user preferences. The Personal Search Assistant architecture provides a framework for improvements in web searching and information retrieval with respect to resource allocation, user effort and personalized searching.

Introduction

According to the results of a report published in the Institute of science the world wide web is estimated to contain 320 million pages as of 1998. Another startling fact is the web continues to grow exponentially doubling every few months.To add to the problem the web is highly unstructured and as a result locating information on the web is getting more difficult. Search services appeared on the scene in 1994 when Yahoo (http://www.yahoo.com) was conceived as the first portal site. In the year 1995 there were about 12 search services. Today there are more than 2000 search services available on the web. The necessity of search services are evident when today all of the popular sites on the web include a search service or provide access to one.

All these search engines such as Yahoo, Altavista(http://www.altavista.com) and Infoseek (http://www.infoseek.com) follow the standard technique of using a spider to update an index of web pages. Due to the extremely rapid growth of the web this technique of indexing suffers because of the need to upgrade the index to include new web pages added onto the web apart from refreshing the existing index. Inspite of extremely efficient indexing and storage techniques none of the search engines can claim of a comprehensive search index. The result, a user using a particular search service may lose out on potentially useful resources returned from another service. To avoid this a user would need to submit and resubmit his queries to multiple search engines. Web searching by itself is a time consuming process and would be more so if one needs to resubmit queries over multiple search services [7].

Meta search services have been described as the next level up the information food chain. A Meta search engine provides a common interface to query multiple search services at the same time. Meta search services do not maintain an index or a spider of their own but instead rely on the underlying search services.

The primary advantage of using a Metasearch service include

- The need to access only a single web page to present a query.
- Needs to only learn a single search interface and query format.
- Can perform search across a wider range of search engines.
- Can get integrated set of results (often with duplicates removed).

With the Internet growing at the phenomenal rate it is today, the amount of user time spent in locating relevant information will increase proportionately. In such a scenario search services would move towards becoming an integral part of Internet usage.
This paper presents the Personal Search Assistant as an Meta search architecture providing a framework to solve some of the problem with world wide web searching. We shall look at some of the problems with web searching and later show how the PSA attempts to solve some of them. We shall use the term search service to include search engines located on the web and Meta search service for its Meta search counterpart.

Current Problems with web searching

We shall first investigate some of the problems with web searching as presented by some of the search services available on the web. Although other issues are involved this work shall limit itself to providing a framework to solve some of the problems listed in this section.

1. None of the search engines are comprehensive

With the phenomenal growth of the web most of the search services have accepted the fact that it would be almost impossible to index the entire web. They instead concentrate on a specialized subset of the web[5] and use ranking techniques to determine which of the web pages to index. The user is slowly growing aware of this and the fact that a service which does not return a single relevant hit does not imply that such a resource does not exist but simply implies that this service does not have it in its index . As a result the user is forced to resubmit his queries over multiple search services to avoid missing potentially useful resources.

2. Most of the popular search services are used online

Most of the search services are themselves located on the web and submission of queries and browsing results is done online. The availability of sufficient bandwidth is an increasing problem in many parts of the world and inspite of low response times this mode of usage of search services puts additional pressure on network resources. The amount of time spent by the user in browsing through results returned is considerably more than the response time of the service. None of the services allow this phase to continue offline which could be possible if the results were downloaded. However the user is averse to this because of the large proportion of irrelevant hits.

3. None of the search services on the web store user information

None of the search services on the web store information about the user as a result each search is a fresh start at skimming through millions of pages and returning results. Search services have no way of storing similar or same search queries initiated by the same user or other users to present the results immediately .
Studies have shown that users generally restrict themselves to a particular subset of topics when they initiate a web search. This can be used to construct a user profile which should be extremely useful in search optimization [1]. Considerable research is being done on this particular topic but none of the major search services construct and use user profiles.

4. Search services provide no information on what they do not index

None of the search services provide information to the user on queries which would not return any results to the user. Instead the user needs to submit a query and later be presented with no hits.

5. None of the Meta search/search services make sufficient usage of history information.

Every time a search query is submitted and results are obtained a large amount of history information is available to the Meta search service in terms of which search services returned a greater proportion of relevant queries and in shorter time with reference to the search query. Using such information for future search and recommendation could improve relevance , response time and efficiency of the service to a large extent.
Search services could on the other hand store information relevancy of sites as evident from user perception.

Using a Meta search technique for resource information gathering on the web is an already well developed field with numerous search services available. The Metacrawler (http://www.metacrawler.com) and the Savvy search (http://www.savvysearch.com) are two well established search services The Metacrawler utilizes 9 search engines to which it submits its queries. The service removes duplicates from the results returned and presents it to the user in a click able format. The All in One Page is a Meta search service which provides results from multiple resource applications which include gopher, archie, http and others.

The above cited problems have been addressed to various extents in some of the present day search utilities. The current work attempts to provide an alternative architecture to address the above mentioned problems. The focus of the PSA project was initially to address problem of user time spent in searching for required information on the web. However, these problems are all interconnected and the PSA project has grown to address the other problems as well.

Overview of the system

We shall in this section investigate the major subsystems of PSA. We shall in the process understand the function of each subsystem with a view towards the overall architecture of PSA. The overall architecture is shown in Fig I in the following section. The current version of PSA has been implemented using CGI scripts written in Perl (5.0) which run under the Apache Web Server (1.3.0).

The various subsystems of PSA are described below.

Subsystems

The PSA consists of the following major subsystems

(a) Collector. (b) Update Subsystem. (c) Local Repository Subsystem .

(e) Personal Agent. (f) Number of Interface scripts(Query, Database).

(a) Collector

Each time the user submits a search query to PSA it begins a collection phase. The collection phase begins with simultaneous query submission to multiple search services on the web. The collector uses an efficient and upgradeble way of locating and retrieving resource information. PSA can be configured to retrieve urls,ftps and documents related to a particular query term. The collector is an equivalent of the broker in the resource discovery system [2].However since PSA relies on other search services for resource identification, it does not need to spider and instead only communicates with the search services. The collector needs to understand and query each service in its own interface and involves integrating varied search interfaces into a generic format [4].
The collector essentially works in the background and more than one collector session can be initiated and terminated. The need for this was to enable PSA to reside on a local intranet and used in a multi user environment. Each collector interacts with the users own Personal Agent before conducting a search and results get returned into the users own tablespace. The above could also cause resource problems if not handled carefully. It would be necessary for PSA to make a check of resource availability before making any use of network resources. PSA does this using a Resource allocation table which contains the current availability of network and system resources. This table is updated just before a new collector session is initiated. The collector could be configured to utilize bandwidth during low usage. A decision is also made on how many search engines to query in parallel.

(b) Update subsystem

The collector subsystem above retrieve the results from a search query. It however does not load the results into the local repository. This was a conscious policy decision made in order to ensure the user is aware of the what results are being loaded into the local repository by making it manual. The user is presented with the results of a previous search and expected to manually load the results into the local repository. The user also has a choice of not doing so but losing the results obtained.
The update subsystem is responsible for loading results into the appropriate table space and inserting links to the formation of a concept database. The result format of varied search engines are taken care of by this subsystem and the results are converted and stored in a unified format.

(c) Local Repository subsystem

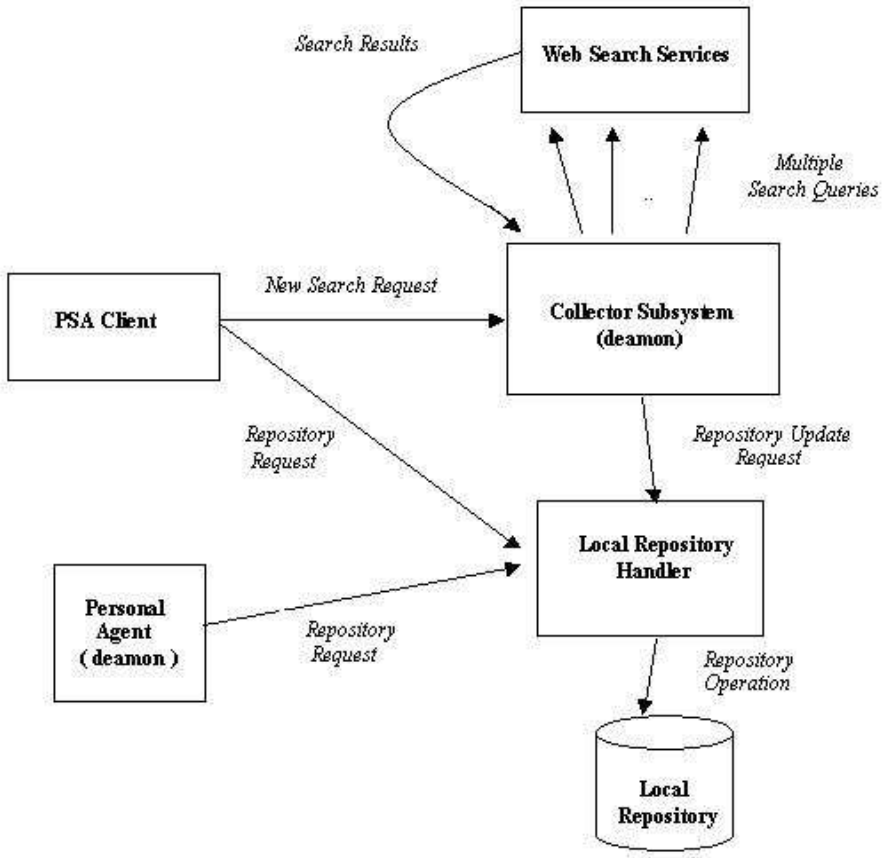
The local repository browse session allows the user to view the contents of the local repository. The local repository has been kept totally independent of any collector phase in progress, which means it can be viewed at any time. The repository subsystem is responsible for handling all requests for viewing, searching and modifying the repository. It would also handle requests from the different subsystems in their interaction with the local repository. This includes the update subsystem and personal agent. The Update subsystems requests the Local repository subsystem which updates the repository.

(e) The Personal Agent

The agent is responsible for personalizing the system and the construction of user profiles.The agent continues to work as a daemon process irrespective of whether a search has been initiated or not. The agent observes various user actions which may be useful for constructing a user profile. There are two major user sessions when the agent collects user information. Initially when the user configures a fresh search, the agent collects the following information from the repository. The search terms, the preferred search engines and their priority, the number of levels through which a collection phase takes place and whether documents need to be downloaded.
Additionally when the local repository is browsed by the user the user browsing patterns can be used to construct a user profile and a search plan for the user.

An example would explain how the Personal agent works for a test search session. The user initiates a search for "Statistical Pattern Recognition" and requests for the usage of Altavista, Infoseek and Metacrawler to return results.Additionaly the user request the service to return documents related to the search in addition to urls. The user also provides an upper bound on hits returned. The priority allocated to the engines are in the order altavista, metacrawler and infoseek. Once the collector session is initiated the user exits. The collector uses the search engine priority to determine which engines *not to* query in the case of a resource constraint. At the termination of the collection phase the local repository is updated with the results of the search , performance related issues which include success ratio of the search engine which is a factor of the returned hits to the average speed.
The user may later initiate a session with PSA upon which is immediately presented with the results of the previous search. The results include the speed and success of the engines. The hits returned and the documents retrieved. The user may immediately browse the local repository or do it later. In the browse session a trace of the user activities is saved and used by the agent. In the above example the user browses through the results of "statistical pattern recognition" and investigates technical papers dealing with "density estimates". The conclusions are that the user prefers technical papers against other documents and is specifically interested in density estimates. At the end of the browse an estimate of the most preferred search engine is made from the browse. Such information is used to reinforce a confidence level in the profile by looking for user confirmation. It should be kept in mind that PSA simply makes recommendations to the user but does not take decisions and looks for confirmations.

The Operation



Overall System Architecture of the PSA (Fig I)

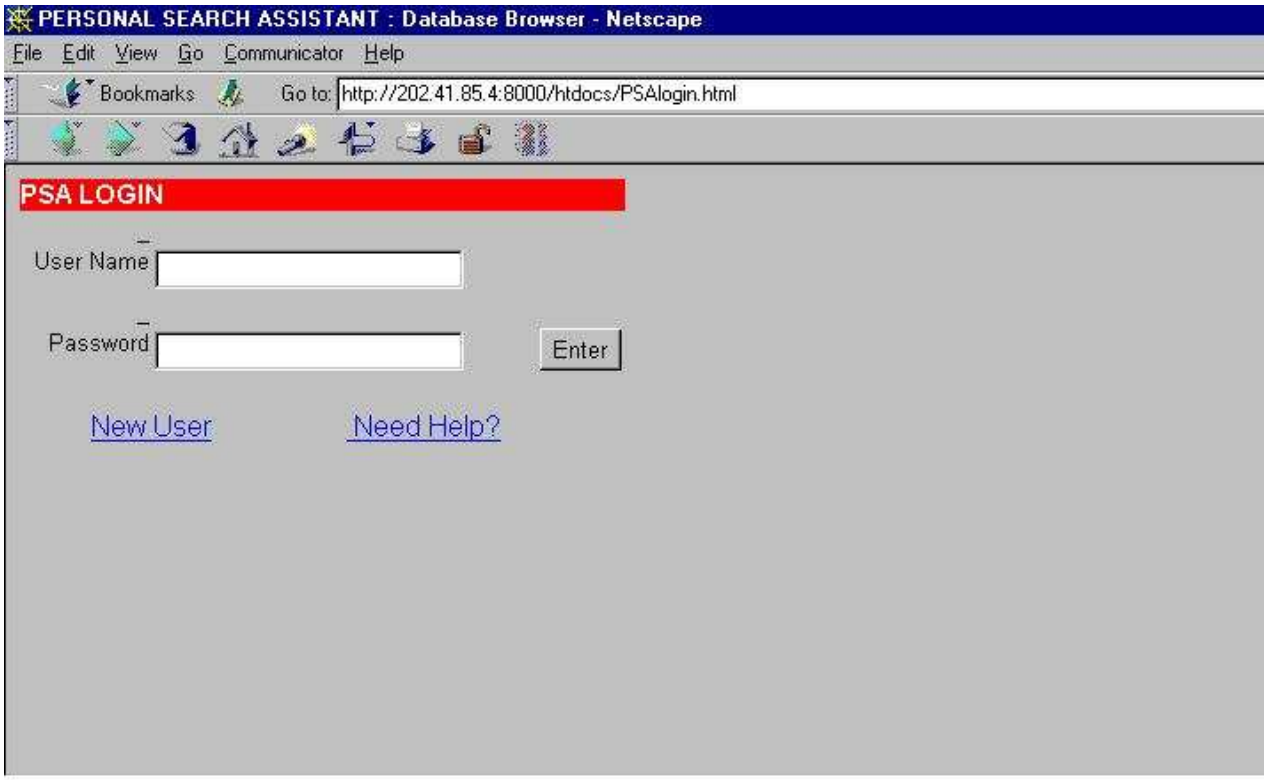
Two of the important subsystems of PSA are the Collector and the Local Repository subsystem while the personal agent works in the background.
We shall present how a typical search session is initiated using PSA and how the major subsystems interact as a result. The PSA server components could reside either on the local web server serving people on a Intranet or on a public domain web server although the former would be more suitable to its architecture.
The client component of the PSA resides on the local user system, shown as PSA Client.
Each user logs onto the search service using his unique userid and password [Screen I]. The user is initially presented with the results of his previous search [Screen II]. There would be no results for a new user.
The user may immediately view the results from Screen II. He may otherwise choose to view the results at a later stage.

The user can choose to query a new search term through the search interface as shown in Screen III. The search is processed in the background and as a result more than one search can be issued simultaneously. The user is provided an interface for configuring the system. Ideally the user starts one or more collector sessions and exits the system.

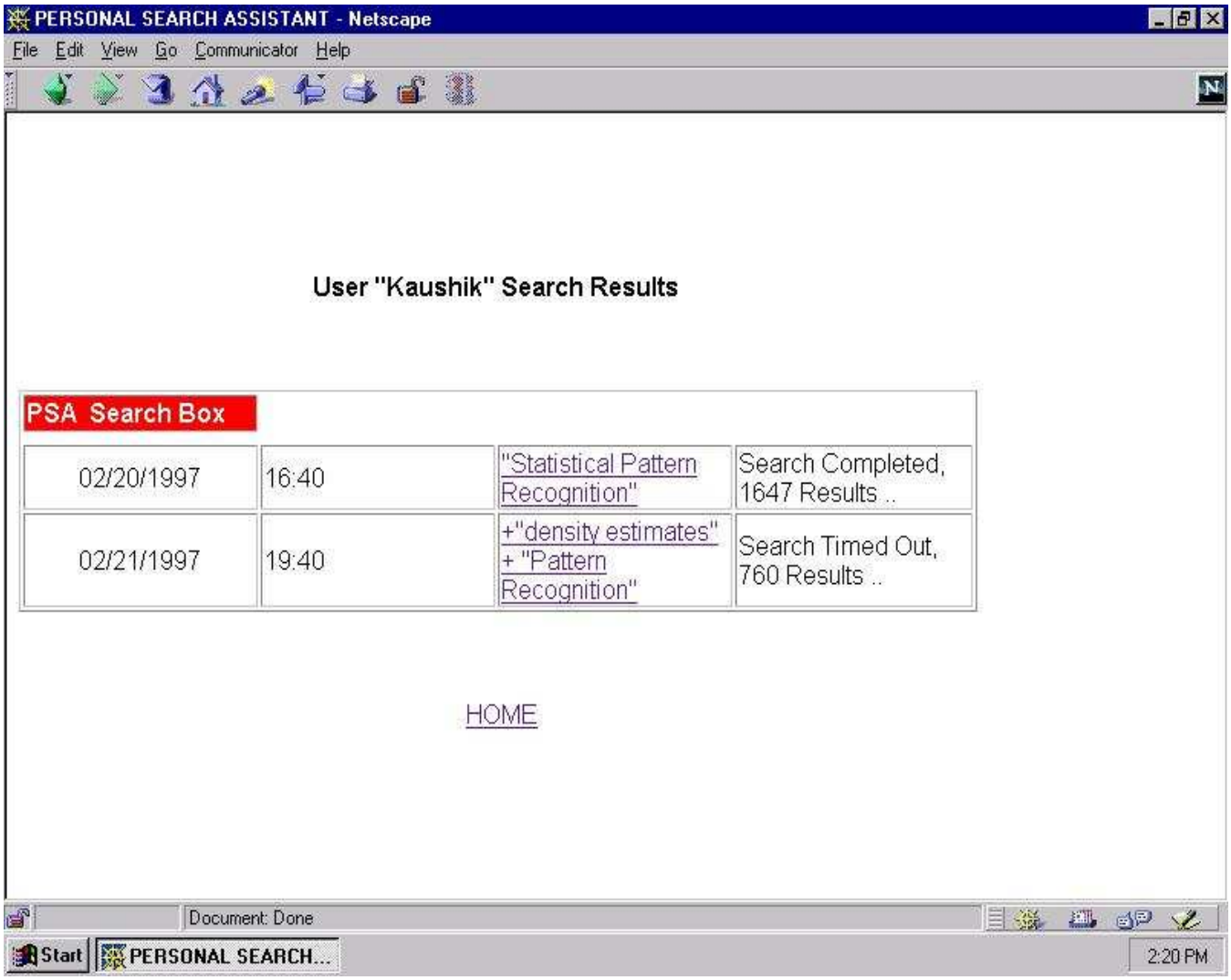
The user may also login to investigate the local repository. This can be done at any time independent of the collection phase. The repository is subdivided into concept databases based upon an initial search configuration into which results of a search are returned. For example a user could initiate a search for "statistical pattern recognition" and configure the results to be stored into a concept database called "Pattern Recognition". The user on investigating the repository is presented with the set of concept databases. The default cases go into a default database.
The different search queries for "statistical pattern recognition" and the number and success ratio of hits from different search engines are stored in the concept database "Pattern Recognition". Browsing through the concept database would allow the user to make an estimate of which search terms are more relevant to his requirement with a view of the query and their results.
The other major information stored in the concept database goes towards the construction of a user profile. The user preferences in browsing through the local repository is used to construct a user profile. The system makes suggestions and looks for confirmation to reinforce its convictions. The Personal agent stores a profile of the user by gathering information from the local repository.
PSA makes valuable savings in resource utilization in two ways. PSA first looks into the local repository if a similar or same search has been issued in the past. If there is such a case , PSA uses success ratio of different search engines to make suggestions to the user. The user is however free to choose the engine and its priority. In the case the user does accept , PSA avoids querying Search engines having a lower success ratio, this would major saving in resource utilization.
In either case the system uses a resource table which is used to decide how many search engines should be queried in parallel and in which order based on an overall ranking (Overall Engine Rank). The resource table is a measure of available network resources. Since the system can easily overload the network with multiple collector sessions each involving multiple Search Engine's this is very necessary. A ping routine is used to update the table.

The Collection phase is started by the user logging on with his unique user id and submitting a search query. Having submitted the query the collection of related web resource links takes place, normally in the background. At the end of the collection phase the Update subsystem updates the repository. The time taken for the collection phase depends upon the network load. PSA shifts between using the entire network bandwidth when the load is low to remaining idle for some time when the network is heavily loaded.

The Local Repository can be viewed by the user any time using the database interface scripts. This implies that the user is not exposed to the delays due to network load and to him the results available from the previous searches are presented instantaneously. The user may, however, update a partial set of results from the present collection phase to be viewed from the Local Repository. The user may also use any of the search engines online by disabling the background mode.



Screen I (Login Page)



Screen II (Search Results)

System Comparison

Personal Search Assistant is a Meta search tool which is based on an architecture that deviates from other search utilities available on the web due to the advantages presented by using such an architecture. The tool is ideally expected to reside on the local intranet or the user system The fundamental task ahead of the personal search assistant was to reduce the amount of user interaction and user time spent in scanning through often irrelevant information returned from the search engine. The system in the process of addressing problem 1-6 given above *(Refer section 2)* has managed to substantially

decrease the amount of user time spent for a single search.

The personal search assistant is a meta search tool in the sense that it does not perform the following tasks performed by the conventional search engines:

1) spidering the web for new and updated pages, 2) storing the web pages with an index.

As stated in section 2, none of the search engines are comprehensive - they provide access only to a part of the web. This problem could only grow as Internet keeps growing and the information on the web gets more and more unstructured. This does not undermine the need for search engine but search engines are unsuitable for a naive user. We believe that meta search engines will be tool most suitable for user interaction. New search engines could be added as a service provider to meta search engines with very little effort. Changing an existing search engine to provide a variety of services is comparatively much more difficult. PSA works like a conventional meta search engine submitting queries to multiple search engines and the result are collected and returned to the system. The architecture of the PSA also allows the system to add and remove the underlying search engines with very little changes to the collector subsystem, and the Local Repository subsystem while the other subsystems are left unmodified.

As compared to any other search/meta search engine PSA submits all the queries in the background mode and instead of collecting the results and returning them to the user as a click able set of references, it loads a Local Repository. The user does not spend any time on the global search phase because it is a non interactive phase. The user interacts with the local repository at any time during or after the search is over. PSA allows the user to specify the need for a second or multiple retrieval levels. All such documents would be stored on to the Local Repository. Since the Local Repository is located on the user system link failures do not affect retrieval from the Local Repository.This feature is a deviation from traditional search architectures and allows the user to store previous search results.
This mitigates problem 2 since most of the user interaction is with the Local Repository, while the global search phase is an automated process which does not need any user interaction.

The problem 3 is directly addressed by PSA by storing along with the results the query which caused the search. PSA prepares every fresh search by comparing with the local repository. If the same query or a similar query has been processed by the system the search is terminated and the user presented with the results. None of the web search services provide such a feature since it would lead to storing an enormous database of results and would only be possible on search service running form the local system. Studies have shown that each user spends most of his time in a restricted subset of the web. This implies that the expansion of the local database slows down and saturates with time. In any case, the time saved for the user may more than compensate for this space needed. The user would also be completely free to edit/remove entries from the database at his will.

The problem 4 is another important problem with many search engines which have hardly any pages related to particular topics. For example, the term "entertainment corpus" returns few hits from Infoseek while Altavista would return a greater number of hits. PSA, in the process of searching particular terms, would keep track of search engines with no hits returned and will not submit the query to those search engines again. The user would also be saved of any time spent in submitting the query personally to a search engines which indexes no pages on the topic.

Problem 5 is linked to Problem 4 above where previous queries, the response, and the response times of the search engine are kept track of. This could be efficiently used along with a relevancy rank to decide which search engines to use or omit in a particular search.

A great deal of research is going on relating to Problem 6. The benefit of using PSA from a personal computer and not from the web is personalization of the search. Search engines on the web do not provide any facilities for registering a user and keeping track of his previous searches since this would mean browsing through another enormous database. Storing user searches and the links followed could be used to construct a user profile to make suggestions on a variety of issues

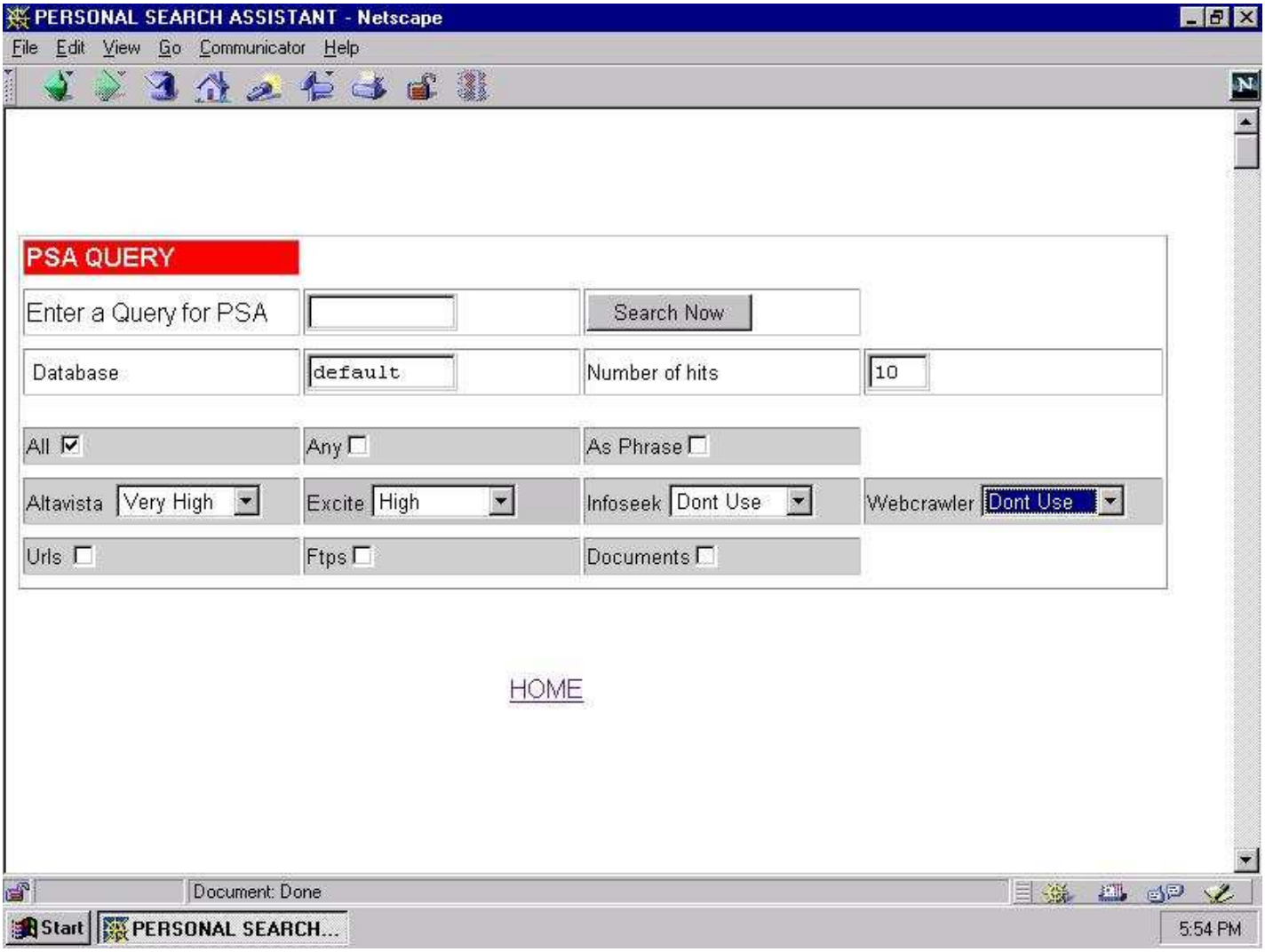
including the query format,query level, which search engines to use and so on.

User Interface

The user interface of PSA provides a minimal query format very similar to the web/metacrawler (http://www.metacrawler.com) [6].

An important aspect of search interface includes a choice of which search engines to include in the search. The test version of PSA presently provides a choice of 4 search engines. The search engines are Altavista, Infoseek, Excite and Webcrawler. More search engines can be easily added. The number of levels to which the retrieval proceeds can also be specified by the user. The other important options provided by PSA is the kind of information to be obtained from a query submitted. For example, PSA could be configured to return for a single search url's, emails, ftp sites, News or a combination of all of these. A time-out limit can also be specified by the user to terminate the search. Instead a limit on the number of hits returned may also be specified.

The Search page of PSA is shown below.



Screen III (Search Interface)

Results from the PSA

We make a comparison of PSA with some of the other commonly used meta/search engines on the World Wide Web. PSA being a meta search engine should be strictly compared only with other meta search engines but alongside, a comparison with some of the popular search engines have also been made to show how meta search engines in general and PSA in particular compares with search engines. Comparison of the search engines are made on some of the commonly used parameters: [3]

(1) Coverage (2) Precision (3) Response Time (4) User Effort (5) Form of Output .

Coverage

PSA, as other meta search engines, is not restricted to subsets of the web since it can be configured to include any of the public domain search services available on the web, although this may imply some amount of overlap. PSA like any other meta search engine could be configured to cover a major portion of the web based on the span of the underlying search engines.

Precision

The Precision of search engines are based upon the ranking of the pages by the search engine. Meta search engines construct an overall ranking of pages returned from different sources. PSA attempts to rank the pages based on user profiles. This implies the data collected from the local repository is used to decide which of the pages were followed in previous exactly same or similar searches. Links followed by the same user is given higher weight while those followed by a different user would be given a lower weight. These weights are accumulated and stored as a part of browse trace within a concept database. In the case where a search query completely or partially matches those from a concept database, this browse trace is used. PSA provides a suitable architecture for constructing and using User Profiles for search optimization as against most of the public domain meta search engines.

Response Time

The response time of a search engine is an extremely important parameter based on which the search engine is compared. The Response time of PSA however is not an important factor since PSA does all its collection in the background mode hence the response time of PSA does not involve the waiting time of the user. We list below the response time of some of the major search engines [8] which would give an idea of how much time the user is expected to wait once a query is submitted to a search engine. It must be noted that the actual wait times are highly variable and depend greatly on the available bandwidth and network load. Wait times can be and often are worse.

Url	Search Engine	Avg. Response Time
http: //www.altavista.com	Altavista	0.9
http://www.hotbot.com	Hotbot	2.6
http://www.excite.com	Excite	5.2
http://www.lycos.com	Lycos	2.8
Experimental	Inquirus	1.3

User Effort

The amount of user effort spent in initiating and utilizing a search is an important parameter in deciding the efficiency of a search engine. PSA goes a long way in decreasing the amount of user effort spent in constructing a search and later using it. This is in fact the long term goal of the PSA project. Personalizing web searching to suit user likes and dislikes also leads to minimizing the effort the user spends in a search. PSA works in the background mode and all the results returned by the various search engines are stored into a local database without any user interaction. The minimal user interaction required for initiating a search is to configure a search according to the user's preferences.

The user interaction with the local database to browse the results of a search, ensures minimal waiting time as against other search engines where in spite of first results being returned by each search engine within a few seconds, the amount of time taken to browse through the results returned is much greater.

Thus,

- $T_t = T_f + T_r$

  Where,

  T_t = Total time spent for a term search using a Meta/Search Engine

  T_f = Time for the first Set of results

  T_r = Time to locate relevant results within results returned.

PSA decreases the first value since working in the background mode can return results faster than on an online system. The first factor loses its importance in PSA since the user is not exposed to this waiting. The second factor is in fact the greater of the two terms and this is what PSA attempts to reduce and does so quite successfully. In spite of relevancy ranking and other major ranking techniques used by search engines to return the most relevant results first, the user needs to browse through the returned results to identify ones most suitable to his/her requirement. PSA also supports this view and instead of making the user to go back and forth online through the results allows him to do the same once the results have been stored onto the local database. PSA does not take the other extreme step of downloading all the pages which would put an immense load on the network bandwidth. Instead PSA downloads pages only if configured to do so and making optimum use of available bandwidth.

PSA also continuously checks for changes in web pages and updates the corresponding pages onto the local database. Pages not on the local database are automatically downloaded if the popularity of the page exceeds a specified value. This again

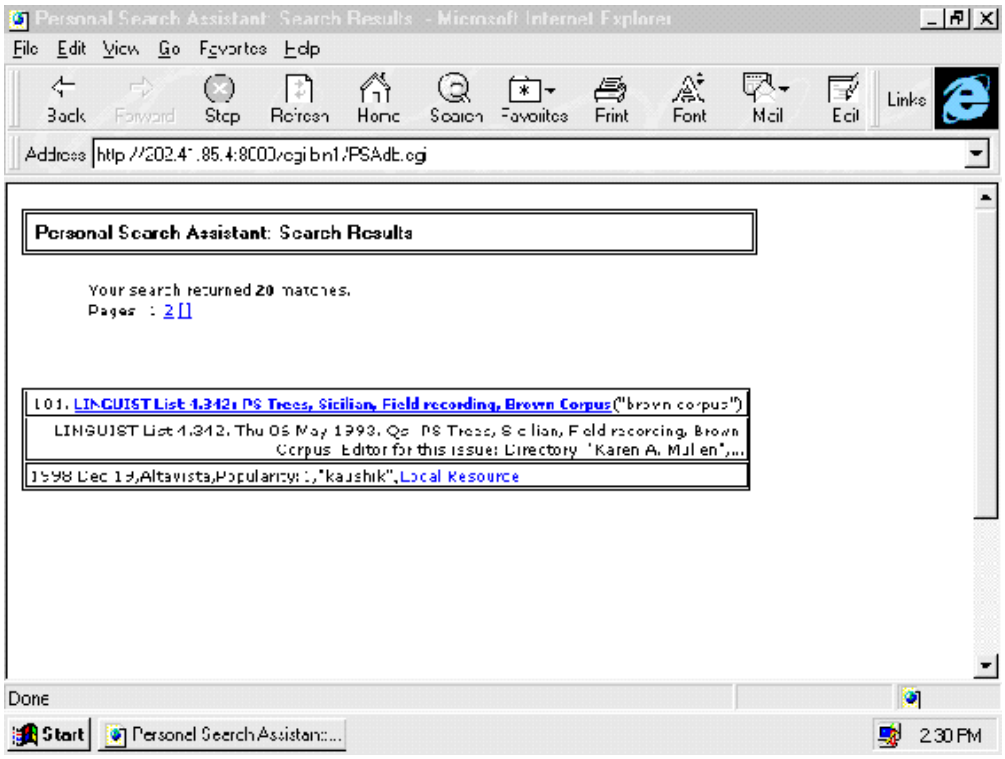
can be specified by the user.

Most of the major meta/search engines on the web returns results online to the user. The response time of the engines may be minimal but the total time spent by the user includes the time spent to locate relevant results within the hits returned. Taking into consideration the minimal availability of bandwidth at many locations the response time may average 7-8 seconds and a much larger value represents the time taken to locate relevant hits. PSA compares very favorably under such circumstances.

Form Of Output

The form of output available from PSA is very similar to the altavista search page . The other major terms specified in the output include Popularity (a rank of how popular the link is) and the search engine which returned the result.

A section of the output page of PSA is shown below.



Conclusions and Future Directions

Personal Search Assistant is a Meta search engine for world wide web searching. An architecture for Meta -searching on the web has been defined during the course of this research. PSA differs from other traditional Meta-search engine architectures since it works by gradually updating a local repository rather than present the results immediately to the user. The use of History information from the Local Repository for optimizing future search request is another major deviation. PSA prototype system has been implemented using CGI scripting in Perl (5.0) which runs on Apache Web Server (1.3.0). With the coming years Web searching would definitely need rethinking with the enormous growth of the web and the premium on user time. The direction which PSA would take in the future would move towards a greater level of

personalization. This would require efficient use of history information at every aspect o searching and returning results to the user.
Since PSA also provides a facility for searching online as in other search engines , the only drawback would be that it uses other Search Engines as service providers as any other Meta-search engine does. This does not seem to be a major problem since addition of new search engines and removing existing ones could be done with very little modifications.
Search services of the future need to merge agent architecture with existing search architectures. Although existing Search Engines have very efficient search architectures they have hardly any time for intelligence. Search services have to move towards this goal in the future.
The current implementation of PSA has concentrated on constructing a suitable architecture to solve some of the problems with web searching. The architecture provides ample scope for extending the current implementation towards a full fledged search assistant. A Major contribution of PSA would be the construction of concept databases and user profiles which could be shared or exchangable among users of PSA, apart from an architecture which would reduce the amount user effort spent on web searching. The concept database could eventually grow to store information about the conept in a more general sense including common search terms, glossory, and people related to the concept.
The current user profiling scheme using a simple accumulative scheme although effective needs to move towards constructing classifiers. The use of a classifier would enable profiles to be shared among users of PSA. Major decisions need to be taken about the information to be retained in the local repository as history information, keeping in mind the classifier would be only as accurate as the number of features on which it is constructed. Currently PSA stores only information which would be required for minimal personalisation which however has shown to be effective.
The ultimate goal of the PSA project is to build a search assistant which works on behalf of the user constructing and updating a local virtual web.

References

[1] Mark Ackerman, Daniel Billsus, et al., Learning Probabilistic User Profiles, AAAI Journal, Summer 1997.

[2] C. Mic Browman, Petr B. Danzig, Darren R. Hardy et al., A Scalable Customizable Discovery and Access System. Technical Report CU_CS_732-94, Department of Computer Science, University of Colorado, Boulder, July 1991.

[3] Heting Chu and Marliyn Rosenthal, Search Engines for the World Wide Web: A Comparative study and evaluation methodology, Proceedings of the 1995 world wide web conference,1995.

[4] Daniel Dreilinger, Integrating Hetrogeneous WWW Search Engines.
[HREF1]

[5] Orien Etzioni, The World Wide Web: Quagmire or Gold Mine, Communications of the ACM, November 1996/Vol 39 No 11.

[6] Selberg, E. and Etzoini, O., The Metacrawler architecture for resource aggregation on the web, IEEE Expert, January-February:11-14,1997.

[7] Selberg, E. and Etozioni, O., Multi Engine Search and Comparison using the Metacrawler, Proceedings of the 1995 World Wide Web Conference, 1995.

[8] Steve Lawrence and C. Lee Giles, Inquirus, The NECI Meta Search Engine, Proceedings of the 1997 World Wide Web Conference, 1997.

[9] Vernon , H., Performance of four World Wide Web index services : Infoseek, Lycos, Webcrawler and WWWWorm.

[HREF2]

[10]Victor Lesser, Bryan Horling, Frank Klassner et al., A Next Generation Information Gathering Agent., Umass Computer Science Technical Report 1998-72.

[11]Victor Lesser, Bryan Horling, Frank Klassner et al., A Resource Bounded Information Gathering Agent., Umass Computer Science Technical Report 1998-03.

[12] Brian Pinkerton. Finding What People Want, Experiences withg webcrawler., In proceedings of the second world wide web conference 1994: Mosaic and the web, Chicago IL USA, October 1993.

Hypertext References

HREF1
ftp://132.239.54.5/savvy/report.ps.gz

HREF 2
http://www.winona.msus.edu/services-f/library-f/webind.htm

Copyright

[Proceedings]

***AusWeb99**, Fifth Australian World Wide Web Conference, Southern Cross University, PO Box 157, Lismore NSW 2480, Australia Email: "AusWeb99@scu.edu.au"*