Relevance Feedback and Query Expansion for Searching the Web: A Model for Searching a Digital Library

Alan F. Smeaton and Francis Crimmins
School of Computer Applications
Dublin City University
Glasnevin, Dublin 9, IRELAND
asmeaton@CompApp.DCU.ie

Abstract:

A fully operational large scale digital library is likely to be based on a distributed architecture and because of this it is likely that a number of independent search engines may be used to index different overlapping portions of the entire contents of the library. In any case, different media, text, audio, image, etc., will be indexed for retrieval by different search engines so techniques which provide a coherent and unified search over a suite of underlying independent search engines are thus likely to be an important part of navigating in a digital library. In this paper we present an architecture and a system for searching the world's largest DL, the world wide web. What makes our system novel is that we use a suite of underlying web search engines to do the bulk of the work while our system orchestrates them in a parallel fashion to provide a higher level of information retrieval functionality. Thus it is our meta search engine and not the underlying direct search engines that provide the relevance feedback and query expansion options for the user. The paper presents the design and architecture of the system which has been implemented, describes an initial version which has been operational for almost a year, and outlines the operation of the advanced version.

1. Information Retrieval in a Digital Library

Much of the push for the development and use of multimedia information has been on the development of the technology and so we have seen much work in networking, compression, transmission, storage, presentation and delivery. In order to make effective use of any kind of electronic information, as found in a digital library (DL), the organisation and manipulation of information by content is a crucial component. Thus information retrieval is a key technology in the development of DLs.

The development of information retrieval techniques over the last few decades has been precipitated upon its deployment in a centralised system and though the development of distributed IR systems such as WAIS (wide area information servers) has taken place these have not had anything other than minor impact on the field. Even the world wide web (WWW), the largest distributed collection in the world, is searched by the vast majority of users using centralised IR indexes. A fully operational large scale DL is likely to be based on a distributed architecture and because of this it is likely that a number of independent search engines may be used to index different but overlapping portions of the entire contents of the library. In any case, different media such as text, audio, image, etc., will be indexed for retrieval using different indexing techniques and by different search engines so any kind of retrieval or navigation techniques which provide a coherent and unified search over a suite of underlying independent search engines are thus likely to be an important part of navigating in a digital library.

It is well-known in information retrieval that techniques such as relevance feedback and query expansion provide an improvement in retrieval effectiveness over straightforward keyword weighting and matching. Implementing such techniques over a suite of underlying search engines is thus desirable from a user's point of view as it allows the individual search engines to remain relatively straightforward and uncomplicated while still delivering advanced search options to the user. Given that this approach is a possible paradigm for searching in a digital library and thus is important for the DL community, in this paper we present a technique and a design for an implementation of searching the WWW based on broadcasting a user's search to a number of conventional search engines and combining the results into one overall ranked list. Searching the WWW using such IR techniques is a noble task unto itself but it is also an appropriate model for the kind of IR system we outlined above as there exist a number of search engines which provide straightforward keyword weighting and term matching over overlapping portions of the entire web.

Our approach to retrieval is effectively a meta search engine and the concept and other systems which do this for the web are described in the next section of this paper. We also allow a user to feed back to our system which URLs are relevant to the query and which are not and from this information we can generate a ranked list of search terms which the user can choose to add to the search, or can have the system add them all. The general approaches taken to searching the web are examined in section 2 and meta search engines for web searching are presented in section 3. The algorithm we use to deliver our information retrieval technique and the architecture of our system, is described in section 4 which also includes some screendumps of the user interface. In section 5 we report on the status of our implementation and in section 6 we give an analysis of our approach. A final section presents some of our plans for extending this work and some final conclusions.

2. Searching the World Wide Web

Soon after the world wide web (WWW) was launched some years ago, several groups realised, independently, that effective access to information on the web could not be provided by allowing users to follow hypertext links serendipitously, or by attempting to maintain a classification or directory of the web's content. This led to the development of systems whose function was to constantly "crawl" the web, seeking new pages or updates of old pages and having discovered a new or recently updated page then download and index that page into its local database or index files. The local index could then be made available to the internet community for searching. Such web crawling programs provide the source of the input documents being indexed and when a user queries a search engine such as Lycos then the "documents" returned are actually pointers to the original documents on some remote WWW server.

Since the growth of the web has really taken off a number of other search engines have joined Lycos in their respective attempts to index the entire WWW. Such additions include AltaVista, Excite, InfoSeek and many others. All of these search engines have many things in common including poor support for the concept of a search "session" between user and system. In practice when we search any system in response to our information needs we tend to have shifting or evolving requirements. These arise as a result of the natural evolution of our needs … as we see some documents on some sub-topic of our query we may feel this aspect to be satisfied and we may wish to concentrate on some other sub-topic of our query. In addition, as we examine the content of retrieved documents we expand our vocabulary of the domain of our search, i.e. we get to know more of what we are looking for and we discover good search terms, and thus we may be in a position to expand our original query with additional search terms and/or add specific user-determined weights to search terms based on documents seen so far.

Query expansion, as described above, is not present in most conventional search engines with the exception of AltaVista LiveTopics[1] yet it is known in experimental information retrieval to be an effective aid to the information retrieval task [Smeaton & van Rijsbergen 83]. Relevance feedback is the concept whereby a user's judgement as to the relevance or otherwise of a retrieved document relative to the query is fed back to the search system so that the system may use this information to improve the remainder of the search. This information could be used to automatically add extra search terms or to assign new weights to existing search terms. The usefulness of this as an IR technique has been known about for decades and one of the most successful techniques for doing this was published over 20 years ago [Robertson & Sparck Jones 76]. Technologies such as query expansion and relevance feedback which have been developed for some time and have been demonstrated as improving retrieval effectiveness on large, multi-gigabyte text collections such as TREC [Harman 96] but have not appeared in global web search engines. LiveTopics from AltaVista comes close to this but the query expansion in this case is based on top-ranked documents and not limited to relevant ones as judged by the user. MUSCAT[2] also provides similar functions though it is used for searching intranets or in searching some geographic sub-portion of the global web.

What conventional web search engines do and do well is attempt to index as large a portion of the web as possible and to maintain these indexes to be as current as possible. They generally provide term weighted retrieval, returning a ranked list of

[1] http://www.altavista.digital.com/
[2] The URL here is: http://www.muscat.co.uk/

URLs and they do this efficiently. Some provide functionality above the simple list of words as an input query by allowing query phrases, for example. In order to attract customers and in turn advertising revenue, web search engines compete on the size of their indexes or the portion of the web that they claim to have indexed. Thus the emphasis has been on web coverage rather than on effectiveness and efficiency of searching does not seem to be a problem.

Despite the large engineering efforts which go into delivering web search technology, users are easily dissatisfied with the service, in particular with the amount of non-relevant URLs retrieved. Clearly, by concentrating on coverage rather than effectiveness, web search engine developers have done the right thing at the time of rapid web growth. A search engine which effectively searched only a small portion of the web would probably attract few customers whereas an engine which delivered a search on all of the web but put the burden of sifting through retrieved URLs onto the user, would attract more custom. At this time, however, we need to see web search engine functionality enhanced and some of the more advanced IR techniques which are known to work, incorporated into the services.

3. WWW Meta Search Engines

The idea of using one or more IR search engines as a basis underlying a more sophisticated or elaborate search functionality is not new. In 1982 Morrissey [Morrissey 82] described an intelligent terminal which took a user's query expressed as a set of keywords and implemented term weighting and document ranking by broadcasting numerous independent searches to an underlying boolean IR system. This provided an IR functionality, weighted search terms and document ranking, on top of a less sophisticated boolean search engine. In the Harvest system [Bowman et al. 95] *brokers* provide the indexing and query interface to the gathered information. They achieve this by requesting information from information *gatherers* and other brokers. This layered approach allows efficient use of network bandwidth & resources. The GLOSS service [Gravano et al. 94] is one which suggests potentially good databases to search, based on word-frequency information for each database. A users query is sent to the GLOSS server which then evaluates it at the chosen databases.

A similar approach to using one search system on top of another has also been developed for searching the web with the emergence of meta-search engines, to which our work would be comparable. Examples of this type would be Highway 61, Inference Find!, Mamma, MetaCrawler, ProFusion and SavvySearch[3]. These systems all operate in essentially the same way, querying some underlying WWW search engines in parallel in order to answer user queries. Where they differ is in the

[3] The URLs are respectively : http://www.highway61.com
http://m5.inference.com/ifind http://www.mamma.com http://www.metacrawler.com
http://www.designlab.ukans.edu/profusion http://williams.cs.colostate.edu:1969

processing they perform on the results returned by the WWW search engines before presenting them to the user. Highway 61, Mamma, MetaCrawler and ProFusion combine their results by using a data fusion technique based on document score i.e. they sum the scores given to a document by the different engines. MetaCrawler and ProFusion offer broken link detection as well, although this results in an increase in query time.

Inference Find! clusters the documents returned by the search engines into groups based on their location i.e. what WWW site they are at. MetaCrawler also offers this as an alternative to ranking based on score. SavvySearch has a large number of underlying engines it knows about and concentrates on selecting a subset of these search engines to route a users query to.

One of the differences between our work and these other meta-search services is that the others all use HTML forms as their user interface. This limits the functionality and interaction which can be offered to users. As we will see later in this paper, our client/server architecture and its implementation in Java allows us to offer an improved and more interactive interface to the user but most importantly offers us more scope for development. Our system also incorporates some more effective IR, namely relevance feedback and query expansion.

The meta search engines which we mentioned above should not be confused with the so-called all-in-one pages such as All-In-One, CUSI, Find-It! and Search.com[4]. These are basically a compilation of the form interfaces of different search tools found on the web. They cover a number of general and specialised engines, divided into categories e.g. web, software, people, technical reports etc. There is no parallelism or combination of results involved as they simply redirect the browser to the relevant engine with the appropriate query.

[4] The URLs for these are: http://www.albany.net/allinone/ http://pubweb.nexor.co.uk/public/cusi/cusi.html http://www.iTools.com/find-it/find-it.html http://www.search.com

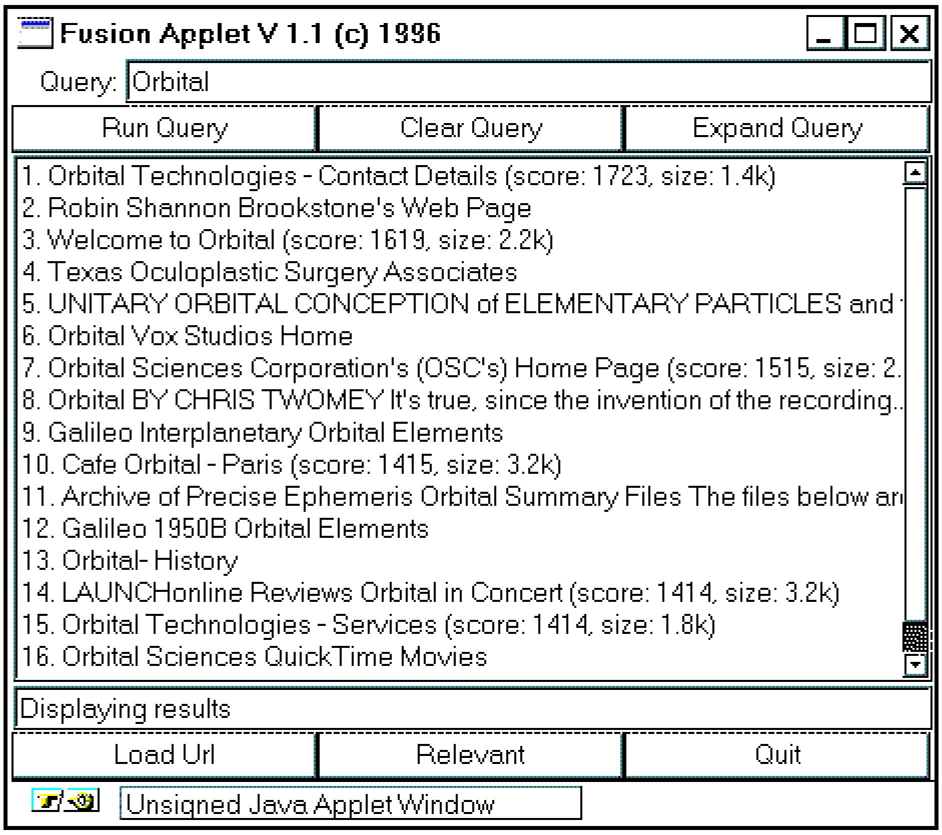
4. Our Meta Search Algorithm and Architecture for Retrieval

Our system uses a client-server architecture with the client being a Java applet running on the user's machine and the server program, also written in Java, running on a Sun Ultra Sparc 2. The client is lightweight in terms of the computational processing it performs in order to have it operational on low-spec machines and the emerging network computers. An architecture such as ours is termed a "knowledge-server" by Eriksson [Eriksson 96].

Our algorithm for retrieving information from the web begins by inviting the user to input a set of keywords or search terms into the client applet. These may be individual words or they may be phrases delimited by inverted commas. When the user has input a query and pressed the "Run Query" button, this query is then passed back to our server from where it is broadcast in parallel to 6 web search engines, AltaVista, Excite, Infoseek, Lycos, OpenText and WebCrawler.

In passing on a user's search to a web search engine we request each system to return its top 100 ranked URLs. This is because of the very poor overlap we have observed in the top-ranked document lists from different web search engines [Smeaton & Crimmins 97]. Of the web search engines we interrogate, only WebCrawler supports a direct request for the top 100 ranked URLs. AltaVista, Excite, InfoSeek and OpenText return only 10 URLs per search request so we break our user's query into 10 individual queries for each of these engines requesting the top 10 URLs, URLs ranked 11 to 20, URLs ranked 21 to 30, etc. Lycos can be interrogated to retrieve a maximum of 40 URLs with one search so this is handled by breaking into 3 separate threads to get the top 100. This yields a total of 44 parallel threads or requests to search engines.

After a time-out period (currently set to 25 seconds) we perform a data fusion operation on the URL lists returned from search engines at that point. This data fusion is performed on our server machine and is based on rank position rather than retrieval status value (RSV) or URL score as not all search engines return scores. For those engines that do return URL scores the range for these is not consistent across search engines with some search engines having no upper limit for a URL's score. The ranked results are stored in a hash table, with the URLs being used to generate the hash code. Thus duplicate objects have their ranks summed, and the objects are penalised if they have not been retrieved by a particular search engine. The table is then sorted into ascending order based on rank and the data fusion process is complete. More details on this process are available in [Smeaton & Crimmins 97]. We take the fused ranking of URLs and send them back to the user's client applet for display to the user. Figure 1 shows a screendump of the client applet where the user has input a single-term query, "Orbital" which has been processed and the top URLs in the fused ranking from the search engines are displayed. The user can double-click any of these or select one and press the "Load URL" button to have that web page retrieved from the web and displayed in the user's WWW browser. As more and more of the responses from the search engines which have not returned before the first time-out period come back to our server machine, an updated URL ranking is generated and this updated overall ranking periodically gets propagated back to the user's applet window.



Orbital

1. Orbital Technologies - Contact Details (score: 1723, size: 1.4k)
2. Robin Shannon Brookstone's Web Page
3. Welcome to Orbital (score: 1619, size: 2.2k)
4. Texas Oculoplastic Surgery Associates
5. UNITARY ORBITAL CONCEPTION of ELEMENTARY PARTICLES and
6. Orbital Vox Studios Home
7. Orbital Sciences Corporation's (OSC's) Home Page (score: 1515, size: 2.
8. Orbital BY CHRIS TWOMEY It's true, since the invention of the recording..
9. Galileo Interplanetary Orbital Elements
10. Cafe Orbital - Paris (score: 1415, size: 3.2k)
11. Archive of Precise Ephemeris Orbital Summary Files The files below ar
12. Galileo 1950B Orbital Elements
13. Orbital- History
14. LAUNCHonline Reviews Orbital in Concert (score: 1414, size: 3.2k)
15. Orbital Technologies - Services (score: 1414, size: 1.8k)
16. Orbital Sciences QuickTime Movies

Displaying results

Figure 1: Screendump of a user's search having retrieved an initial ranking of URLs

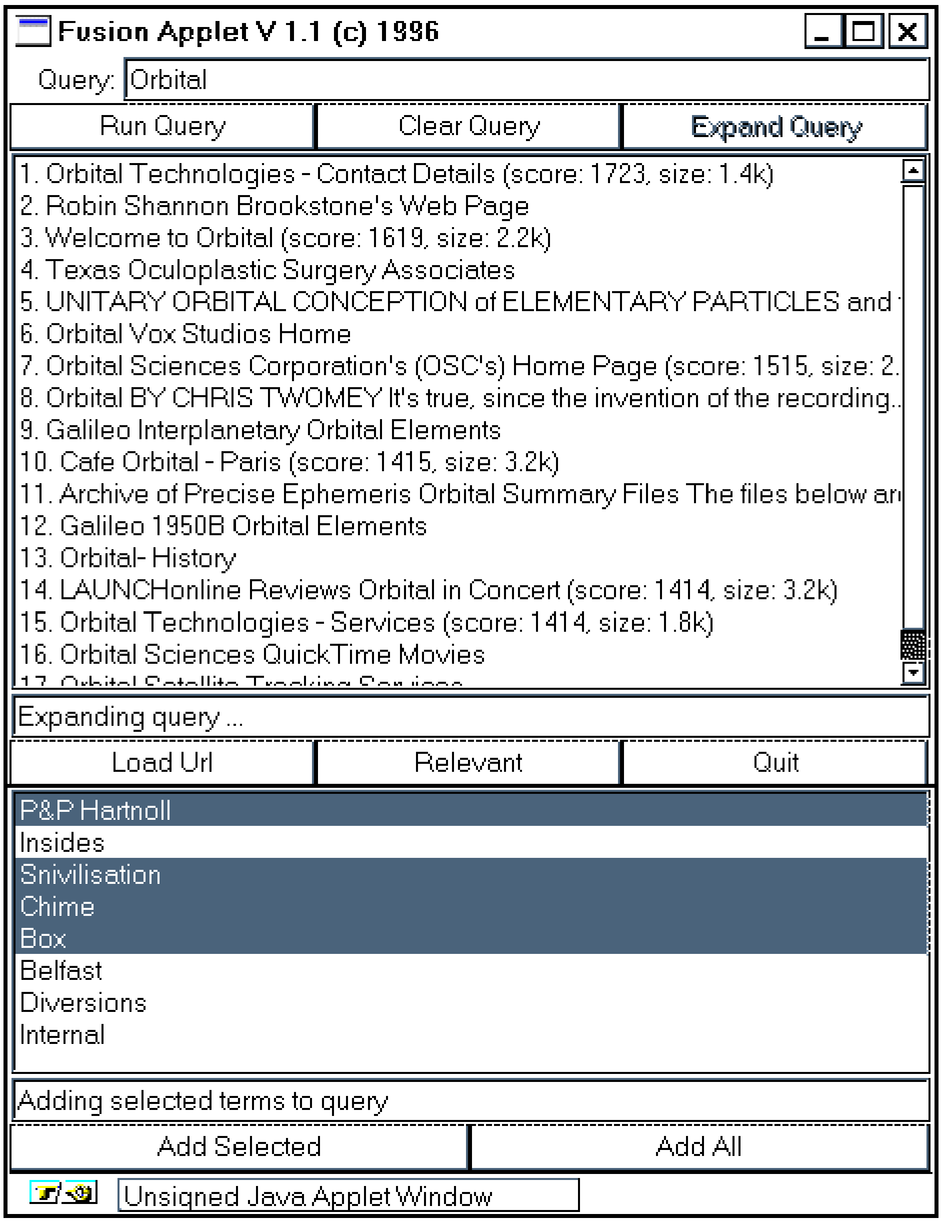
As a user selects URLs on this list, which is scrollable, the full URL is displayed on the status line showing *Displaying results* in Figure 1. On viewing a URL a user may mark URLs in the retrieved list as being relevant (using that button) and this is shown by a colour change for the relevant URLs in the applet's results listing (not shown on the diagrams).

Having viewed some URLs and marked some of them as relevant, a user may invoke the "Expand Query" command by pressing that button. This sends the list of URLs marked as relevant back to our server process which then retrieves those full pages, in parallel, from the web (the earlier retrieval of these pages had been done by the user for display on the user's machine and not on our server and hence they are not cached for us). Pages not returned from the web within a time-out period are discarded from further processing. The text of the retrieved URLs is then analysed by removing HTML tags and stopwords from each, and stemming the remaining text using Porter's word stemmer [Porter 80]. From this we extract a list of candidate search terms which are word stems. Each of these candidate search terms which are not an original query term, taken from known relevant URLs, is then scored using a search term ranking formula.

In [Efthimiadis 95], 8 different formulae for ranking candidate search terms for query expansion were evaluated using an operational information retrieval system. In this work the evaluation was based on how close the formula ranking matched the choice of search terms to add as made by a user. Thus the best of the formulae ranked candidate search terms in such a way that the highest-ranked ones were the ones chosen by a user in an operational setting. Of the 8 formulae tried, the successful ones (Porter, *emim* and the *wpq* formulae) all used the following parameters when scoring an individual search term:

> N is total number of documents in the collection, n is the number of those documents indexed by term t, R is the size of the sample of relevant documents identified so far by the user, and r is the number of those relevant documents which are indexed by term t.

For the case of searching the web where both N and n are unknown, these parameters are difficult to estimate so part of our work involves determining the most appropriate formula by which to rank candidate search terms. It would seem that the simplest one, r/R, may be the best to use. It is certainly the easiest to implement, but our work will reveal whether it is appropriate or not. Once the candidate search terms have been ranked the top ones scored above a threshold are sent back to the user's applet for display, not as word stems but as the set of word form occurrences which are reduced to that word stem. This causes a new pane to open up on the user's applet as shown in Figure 2.



Query: Orbital

1. Orbital Technologies - Contact Details (score: 1723, size: 1.4k)
2. Robin Shannon Brookstone's Web Page
3. Welcome to Orbital (score: 1619, size: 2.2k)
4. Texas Oculoplastic Surgery Associates
5. UNITARY ORBITAL CONCEPTION of ELEMENTARY PARTICLES and
6. Orbital Vox Studios Home
7. Orbital Sciences Corporation's (OSC's) Home Page (score: 1515, size: 2.
8. Orbital BY CHRIS TWOMEY It's true, since the invention of the recording..
9. Galileo Interplanetary Orbital Elements
10. Cafe Orbital - Paris (score: 1415, size: 3.2k)
11. Archive of Precise Ephemeris Orbital Summary Files The files below ar
12. Galileo 1950B Orbital Elements
13. Orbital- History
14. LAUNCHonline Reviews Orbital in Concert (score: 1414, size: 3.2k)
15. Orbital Technologies - Services (score: 1414, size: 1.8k)
16. Orbital Sciences QuickTime Movies

Expanding query ...

P&P Hartnoll
Insides
Snivilisation
Chime
Box
Belfast
Diversions
Internal

Adding selected terms to query

Unsigned Java Applet Window

Figure 2: State of the applet after the user has marked some URLs as relevant and requested the original query to be expanded.

In the worked example in Figure 2, the system has returned the terms P&P Hartnoll, Insides, Snivilisation, Chime, Box, Belfast, Diversions and Internal, ranked in that order, as candidates for addition to the query. For this worked example, none of the expansion terms suggested occurred in any of the relevant URLs in forms other than as the word form indicated, but if, for example, the term “Chime” had occurred as the word forms “Chimes” or “Chimed” then the entry on the applet would have read “Chime, Chimes, Chimed” or as appropriate. We achieve this by saving the original word form occurrence in the web page as well as the stemmed version and performing a simple lookup before display. For the example in question the user is searching for information on the band Orbital, whose songs and albums include the titles “Snivilisation”, “Chime”, “Box” and “Diversions” so it is no surprise that only these word forms occur in URLs marked as relevant by the user.

When the user receives the list of candidate extra search terms he can select some or none of these and they are added to the query. Figure 2 shows the user has selected the terms P&P Hartnoll, Snivilisation, Chime and Box to add to the query and has just pressed the button marked “Add selected”. As an alternative the user could have added all suggested terms to the query. Once the addition of extra terms to the query has been completed the user can re-run the query and the system will close the term expansion pane, re-run the query against the 6 search engines and the cycle can continue.

The idea of using terms from documents which have been retrieved from one system to expand queries for all other systems can be problematic in some scenarios. It is not an issue in this case however because the systems we are querying are all indexing the same collection.

5. Status

A preliminary version of our meta search engine which implements our architecture and combines the ranked output from 6 web search engines but does not incorporate relevance feedback or query expansion, has been operational since July 1996 [Smeaton & Crimmins 97]. This system has served tens of thousands of web searches for the internet community world-wide without any technical problems or advertising to generate customers. There has been no performance impact from this web search system on the server machine which supports the data fusion and communication with the web search engines. The communications with the underlying web search engines has been multi-threaded and our architecture has been proven as valid.

A first version of the user interface to the extended meta search engine described above was developed using Java’s Abstract Windowing Toolkit (AWT) and is presently being subjected to some usability trials and analysis. The screendumps included in this paper are from the applet displayed on a PC running Windows95. The usability analysis will likely result in some changes to the interface as described earlier but these will be minor and the basic functionality will not change. The multi-threaded communication between our server and 6 conventional web search engines has been developed, coded and tested as has the data fusion technique. The basic architecture has been implemented and tested. We are presently testing the query expansion component before releasing the updated system to the internet community for general use, probably by September 1997.

6. Analysis

Technology transfer from research to product is painfully slow in information retrieval though as a result of the need for searching the web and initiatives such as TREC [Harman 96] this is speeding up. For example, probabilistically-based weighting of search terms was demonstrated as having a positive effect on retrieval effectiveness in the late 1970s yet its appearance in products has been relatively recent. One way in which the usefulness of automatic search term weighting and relevance feedback was demonstrated was via the intelligent terminal front end [Morrissey 82]. Nowadays, term weighting is *de rigeur* and present in almost all web search engines.

Developing tools for effectively searching the web is an enormous technical challenge and just keeping up with the growth rate is enough to keep search engine vendors fully occupied. This has meant a minimal functionality as far as the search engines are concerned and lots of effort put into engineering systems which can index as high a portion of the web as possible. This can be clearly seen in the advertising and promotions associated with web search engine vendors where the claims are of how many web pages an engine has indexed. In intranets and in indexes created for single or a small number of sites, IR techniques such as relevance feedback and query expansion can be supported. However, as the global web grows and finding relevant information effectively becomes increasingly difficult due to sheer size, web search engine developers will have to use better IR techniques and start to compete on functionality and effectiveness, not just on the size of their indexes.

One may ask whether or not novice users of an IR system, which is effectively almost all web search engine users, have the wherewithal to effectively use a query expansion technique such as we have here. In [Koenemann & Belkin 96] a series of experiments are described in which a group of IR novice users are asked to find documents from a collection using one of 4 versions of an information retrieval system. The versions of the system used each conceal to varying degrees, the presence of relevance feedback and query expansion. The question explored in the paper is how much of the relevance feedback and query expansion functionality should be hidden from novice and occasional users and how this impacts on the effectiveness of their searches and their perceived usability of the system. The results show a clear benefit from making users aware of these operations and having users invoke them manually rather than having the system perform the operation in the background. These results give support for our decision to allow occasional and novice users of our system the opportunity to determine whether URLs are relevant, invoke a query expansion, choose which candidate search terms are then added to an expanded query and then re-run their query.

Our work described here suggests a mechanism by which more elaborate information retrieval techniques which are known to yield more effective IR, can be used on top of conventional search engines. Our particular implementation and architecture, running a fusion server program to control multi-threaded searches to search engines, most of which are composed of mini-threads to yield a top-100 for each search engine, and generating a ranked list of candidate search terms, needs to be cumbersome because we sit on top of conventional web search systems. This is because our searches to those engines appear as direct user searches and as part of our implementation we have had to write routines to parse the HTML pages returned from searches in order to extract URLs and document ranks. Our work shows that techniques such as suggesting candidate search terms could be incorporated as part of the search service offered by current engines without major modifications.

On the other hand, the current interaction between a user and a web search engine is lightweight and stateless; each search is treated independently and the interactions are based around cgi-bin programs and form-filling. By developing a search tool as a Java applet we have allowed the search to be state-based, with the client storing its state and the server program being made aware of the various stages of a user's search through the client-server interaction, i.e. initial search, supplementary search with known relevant documents, secondary search, and so on. In this way we have been able to move web searching forward from consisting of multiple independent single searches to being able to handle a user's search session, and that is progress.

7. Plans and Conclusions

There are a number of issues that remain to be resolved concerning the implementation of our search tool. We need to investigate a range of search term ranking formulae in order to determine which is the most appropriate for our system. We intend to do this by implementing a number of such formulae including the simple *r/R* as well as a number which assume that URLs in the list retrieved for the user which have been visited but not marked as relevant, are all non-relevant. In addition we are also considering introducing the concept of non-relevant URLs into the interface. Both these extensions are geared towards building a sample of non-relevant URLs where the appearance of a search term in such pages is counted as a negative contribution to the scoring of that search term for term ranking.

The evaluation of web-based IR systems is difficult because the traditional measures of precision and recall are usually based on relevance judgements of closed document collections. In our implementation we will automatically select one of the set of implemented term ranking formulae for each user search in a round robin fashion. We will also log the search terms actually added by users and this will yield the data required to correlate the search term ranking with the actual search terms chosen from a population of real user searches. As with the work reported in [Efthimiadis 95], the yardstick against which we measure is the user's choice of search terms and this work should point to the most appropriate formula to use. All this can be handled by updating the version of our Java server program running on our machine.

As we prepare to launch the second version of our web search system (the first version has been in operation since Summer 1996 but with no query expansion) we must consider the issues of scaleability. To date the first version of the search system has had no real impact on our resources as each user search has been relatively lightweight for our machines. The system described in this paper is much more computationally demanding, managing over 40 multi-threaded searches to search engines for each iteration, plus downloading relevant URLs, stemming them and generating a ranked list of search terms, and all this for each user search. We will need to monitor the use of our system but should it become hugely popular then the distribution of the fusion server process onto multiple servers throughout the net could be done. Our approach of distributing functionality between the client and the server allows us to use resources such as thesauri which would be impractical to locate at the client side. It also allows us to use our system as a research test bed by implementing different techniques at the server side without the user having to know or worry about them.

In the work reported in this paper we have shown an algorithm for incorporating query expansion into searching of the global WWW. The vast majority of the system is implemented and is an extension of an operational system. Our work shows that advanced IR techniques which are known to improve retrieval effectiveness, can be developed for searching the web and this is surely an improvement on the techniques currently available.

References

[Bowman et al. 95] "Harvest: A Scalable, Customizable Discovery and Access System", C. Bowman, P. Danzig, D. Hardy, U. Manber, M Schwartz and D. Wessels. Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado, Boulder, Colorado, USA, March 1995.

[Efthimiadis 95] "User Choices: A New Yardstick for the Evaluation of Ranking Algorithms for Interactive Query Expansion", E.N. Efthimiadis, *Information Processing and Management*, 31(4), 605-620, 1995.

[Eriksson96] "Expert Systems as Knowledge Servers", H. Eriksson, *IEEE Expert: Intelligent Systems and their Applications*, 11(3), 14-19, 1996.

[Gravano et al. 94] "The Effectiveness of GLOSS for the Text Database Discovery Problem" L. Gravano, H. Garcia-Molina and A. Tomasic. In *Proceedings of the 1994 ACM SIGMOD Conference*, May 1994.

[Harman 96] "The Fourth Text Retrieval Conference (TREC-4)", D.H. Harman (Ed.), NIST Special Publication 500-236 1996.

[Koenemann & Belkin 96] "A Case for Interaction: A Study of Interactive Information Retrieval Behaviour and Effectiveness", J. Koenemann and N. J. Belkin, in *Proceedings of CHI'96*, 1996.

[Morrissey 82] "An Intelligent Terminal for Implementing Relevance Feedback on Large Operational Retrieval Systems", J. Morrissey, in: SIGIR82, *Lecture Notes in Computer Science No 146, Research and Development in Information Retrieval, Springer-Verlag,* Berlin, 1982.

[Porter 80] "An Algorithm for Suffix Stripping", M.F. Porter, *Program*, 14(3), 130-137, 1980.

[Robertson and Sparck Jones 76] "Relevance Weighting of Search Terms", S.E. Robertson and K. Sparck Jones, *Journal of the ASIS*, 27, 129-146, 1976.

[Smeaton & van Rijsbergen 83] "The Retrieval Effects of Query Expansion on a Document Retrieval System", A.F. Smeaton and C.J. van Rijsbergen, *The Computer Journal*, 26(3), 239-246, 1983.

[Smeaton & Crimmins 97] "Using a Data Fusion Agent for Searching the WWW", A.F. Smeaton and F. Crimmins, *poster presented at the WWW6 Conference*, Santa Clara, California, April 1997.