Assisting the searcher: utilizing software agents for Web search systems

Bernard J. Jansen and Udo Pooch

The authors

Bernard J. Jansen is Assistant Professor, School of Information Sciences and Technology, The Pennsylvania State University, University Park, Pennsylvania, USA.
Udo Pooch is E-Systems Professor, Computer Science Department, Texas A&M University, College Station, Texas, USA.

Keywords

Software tools, Information retrieval, User involvement

Abstract

Much previous research on improving information retrieval applications has focused on developing entirely new systems with advanced searching features. Unfortunately, most users seldom utilize these advanced features. This research explores the use of a software agent that assists the user during the search process. The agent was developed as a separate, stand-alone component to be integrated with existing information retrieval systems. The performance of an information retrieval system with the integrated agent was subjected to an evaluation with 30 test subjects. The results indicate that agents developed using both results from previous user studies and rapidly modeling user information needs can result in an improvement in precision. Implications for information retrieval system design and directions for future research are outlined.

Electronic access

The Emerald Research Register for this journal is available at
www.emeraldinsight.com/researchregister

The current issue and full text archive of this journal is available at
www.emeraldinsight.com/1066-2243.htm

Internet Research: Electronic Networking Applications and Policy
Volume 14 · Number 1 · 2004 · pp. 19-33
 · ISSN 1066-2243
DOI 10.1108/10662240410516291

Introduction

After over 50 years of development and research, there are still issues concerning the interaction of information retrieval (IR) systems and users, including improper query formulation, ineffectiveness in expanding results, and the inability to reduce results to a manageable number (Jansen *et al.*, 2000; Yee, 1991). Some researchers have developed IR systems with a host of advanced search features; however, searchers seldom utilize these features (Jansen and Pooch, 2001). Others researchers have explored systems that attempt to aid the user in locating the information desired. Classified as intelligent information retrieval, much of the previous work in this area has focused on developing entirely new IR systems or implementing new system front ends. However, many of these systems have not gained widespread use (Sparck-Jones and Willett, 1997).

Given the limited used of advanced searching features along with the acceptance of new systems, we propose using an agent paradigm where a software component is independently developed and then integrated with an existing IR system. The agent we developed provides searching assistance based on identified issues with interactive online searching and on actions taken by the user during the search process. This agent paradigm has the advantages of providing advanced searching features, providing assistance with using those features, and utilizing existing IR systems. This approach avoids the transition costs of totally replacing legacy IR systems. The feasibility and benefits of a software agent, developed in this manner, to an existing IR system have not been previously explored. The goal of this research was to determine whether or not the development methodology and integration were feasible and could achieve increases in system performance using accepted IR metrics.

We begin with a review of literature concerning intelligent IR systems' research and the concept of agents. The development methodology and the specific agent developed for this research are presented in detail. The results of a performance evaluation of an IR system with and without the integrated component are presented. Conclusions,

implications for IR system design, and directions for future research are discussed.

Literature review

The focus for much information system research has been to add advanced system features, such as multiple field searching and Boolean logic (Millsap and Ferl, 1993). However, Hunter (1991) has shown that users of IR systems generally do not use these advanced features. Most searchers utilize only the most basic of search features (Peters, 1993), and searchers also have a number of problems when they do utilize these advanced features (Jansen *et al.*, 1998). Yee (1991) highlights that searchers have difficulty finding appropriate subject terms, retrieve too many results, fail to appropriately reduce or increase the number of results, are unable to understand searching rules, or frequently retrieve zero results. Peters (1993) reports similar problems with searchers using online public access catalogs (OPAC) systems. Other studies (Jansen *et al.*, 2000; Silverstein *et al.*, 1999; Spink *et al.*, 2002) show that searchers exhibit like difficulties with Web search engines.

Research efforts to develop intelligent IR systems to assist searchers during the IR process have many times focused on developing entirely new systems or making major modifications to a system. Using artificial intelligence models, Fox (1987) developed a skeletal IR system as a testing mechanism for various search techniques. Chen and Dhar (1991) developed cognitive models of searching based on empirical user studies. From these cognitive models, they developed an intelligent IR system for key word selection and thesaurus browsing. Both of these efforts were principally prototype systems that explored alternative design methods.

Using expert knowledge, Croft and Thompson (1986) developed a relevance feedback system where the user supplied a natural language query or relevant document as a seed. Oddy and Balakrishnan (1991) developed a networked-modeled system using a highly parallel setting where an approximately one million node-and-edge network represented 10,000 document abstracts. These efforts required extensive system development in acquiring the domain knowledge and in constructing the document network. With the sizes of current document collections, this approach would require a significant increase in system development effort.

Researchers have also explored modifying IR systems by developing new user interfaces for IR systems. Brajnik *et al.* (1987) implemented an adaptive IR interface that utilized natural language queries. OAKDEC (Meadow, 1988) was a front end to a database management system that suggested to users what searching procedure to employ. Gauch and Smith (1993) developed an expert system interface for a rudimentary IR system. This expert system accepted from the user both a query and the number of passages desired. The system then executed and reformulated the query by adding or deleting terms until the desired number of passages was retrieved. System modifications such as these require less developmental effort compared to building an entirely new system. However, searchers must adapt to the new interface, and the new interfaces are usually not system or platform independent.

Taking a methodological approach, Ruthven *et al.* (2001) investigate various query modification techniques, specifically investigating term re-weighting and query reformulation within the paradigm of automated relevance feedback. The research concludes that the use of these two approaches can increase the effectiveness of relevance feedback, thereby improve search effectiveness.

Herlocker *et al.* (2000) examined how to design intelligent systems. Within the field of automated collaborative filtering, the researchers examine the optimal degree of transparency for systems offering automated assistance. They conclude that automated assistance can be a valuable component of a system. However, their experiments did not indicate any improvement in system performance.

Focusing on the Web, Middleton *et al.* (2001) investigate the issue of capturing user information preferences in the dynamic Web environment. The researchers take the approach of unobtrusively monitoring users' browsing behaviors. They then use a machine learning approach coupled with an ontology

representation in an attempt to extract user information preferences. Their system calculates a correlation between browsed Web pages and information topics. Using a time delay function, the system can calculate a topic history with the current topic weighted more heavily.

Rather than develop entirely new systems or interfaces, Lieberman (1998) has suggested the concept of integration using software agents (Maes, 1994). Using the agent paradigm, the software component can be developed external to any particular system. Once developed, the software component can be integrated with a variety of systems within a particular genre, such as IR systems. The agent integration technique has been widely utilized in the Web browser area with applications such as Letizia (Lieberman, 1995) and Alexa (Kahle, 1999), among others. However, these agents do not aid in the search process in the classical sense but instead locate similar Web pages to one the user is currently viewing. In the IR arena, Lawrence *et al.* (1999) developed ResearchIndex, which incorporates a software agent to recommend articles based on a user profile. Chen *et al.* (2001) are developing an intelligent Web meta-indexer for Web searching, which is a stand-alone system that utilizes results from existing Web search engines. The development of the agent for our research generally adhered to a methodology similar to that outlined by Wooldridge *et al.* (1999), which is a methodology for developing one to a small community of agents.

System development

Specifically, our software agent offers assistance to the user during a search session. Referred to as the Agent to Improve Information Retrieval Systems (AI^{2}RS), the agent is operational but still prototypical in nature. Our aim at this stage of the research was to develop the functionality of the AI^{2}RS agent enough to determine if integration within the IR arena was feasible, provide targeted searching assistance, and develop a model of the user's information need based solely on typical user actions during the search process. Because we did not want the introduction of the software agent to add any cognitive load to the user during the search process, we rejected the approach of utilizing user completed profiles and questionnaires or user annotation of relevant documents. Instead, we desired that the AI^{2}RS agent would glean information solely from normal user actions during the search process in order to determine what assistance to offer. This approach is similar to Kamba *et al.* (1993) who used user actions to personalize an online newspaper. They used actions such as save, scroll, time, and window resize to supplement reader annotation of interests.

Development of the AI^{2}RS agent

Typically, software agents build a model of the user's information need and then take action or provide suggestions for the user (Maes, 1994). This approach is usually based on relative long-term interaction between the user and the system. Unfortunately, interactive sessions between searchers and IR systems are typically extremely short both in terms of the number of queries and time, especially with Web IR systems (Silverstein *et al.*, 1999; Spink *et al.*, 2002), and user interests are extremely varied. Much previous research in this area has utilized intrusive methods to gather additional information from searchers (Croft and Thompson, 1986; Gauch and Smith, 1993; Koenemann and Belkin, 1996), such as surveys, number of passages, or relevant judgments. Since we did not wish to place additional burdens on the user during the search process, a method of rapidly modeling the user's information desires was needed.

User modeling has become a research field in its own right because it is such a vital component of system interface design (Marchisio *et al.*, 1993). GOMS (Card and Moran, 1986) is probably the most well known theoretical framework in this area; however, it does not directly apply to IR systems. Some IR-specific user modeling theories have been developed; for example, the stratified model for IR (Saracevic, 1996) views the IR interaction as a dialogue between participants, the user, and the "system" through a common interface. The dialogue occurs between participants at different levels. Green and Benyon (1996) take the approach of using an entity relationship diagram for modeling information artifacts

Internet Research: Electronic Networking Applications and Policy
Volume 14 · Number 1 · 2004 · 19-33

(ER - MIA), focusing on interface objects rather than detailed modeling of user goals and tasks. Certainly, within the field of IR, the reformulation of the query has been the standard approach to user modeling. This reformulation has typically been with the use of relevance feedback, where a relevant document is used to modify the user query.

For our approach, we needed a methodology that focused on more than the query but also on other user actions during the search process. These users' actions must be recorded in a way that is convertible into code in order to modify a system. To accomplish this, we modified a technique used in adaptive hypermedia systems in which a model of the user is represented by a set of pairs (c, v) where c is a concept and v is a value (De Bra and Calvi, 1998). A concept is an idea, a subject, a preference, a submission, or a topic (i.e. a noun). A value is a measure that associates the user to that concept. The value can be Boolean or a numeric value.

We altered this approach for use in the IR search session. In our modified technique, a series of action-object pairs (a, o) models a searcher's information need during the session. On any IR system S, a user U has an information need I during a session s. We define s as the entire sequence of queries entered by a searcher during one episode of interaction with S. This definition is in-line with that proposed in Jansen and Pooch (2001). The sequence of (a, o) pairs is built using the searcher's normal interaction with the IR system and requires no additional actions by the user. An action a represents a specific interaction of the searcher with the system. An object o represents the receiver of the action a. Therefore, I is represented by ${}^{s}\sum(a, o)$ on any S. Again, this information need applies to those expressed during a search session. Naturally, there are information needs that may transcend multiple sessions. Our model does not yet address these information needs, although the model can be extended to address these situations.

The AI^{2}RS agent currently monitors the searcher's interaction with the system for five actions; a is an element from the set {bookmark, copy, print, save, submit}. There are currently three objects that the AI^{2}RS agent recognizes; o is an element from the set {documents, passages from documents, queries}. Using (a, o) pairs has several advantages compared to other methods of gathering information from a user during a session. Namely, the user does not have to take additional actions (e.g. answering questions, completing profiles, judging relevance) beyond those of typical of user-system interaction, yet the user's query is not the sole representation of the user's information need.

The valid object in the (a, o) pair varies with the type of action. Document objects are applicable to the actions of bookmark, print, and save. Passage objects are applicable to the action copy, and query objects are applicable to the action submit. For example, if a user bookmarks a document (e.g. www.thiswebsite.edu), the (a, o) pair would be (bookmark www.thiswebsite.edu). The AI^{2}RS agent builds the model of the searcher's information needs by recording, storing the series of (a, o) pairs during a session, and then offering assistance based on the series of (a, o) pairs recorded. Using this approach, one can model the user's information desires relatively rapidly.

When a session begins, the AI^{2}RS agent monitors the user for one of the five actions, via a communication line using an application program interface, a technique commonly used to integrate software programs. When the AI^{2}RS agent detects a valid action, it records the action and the specific object receiving the action. For example, if a searcher was viewing *this_document* and saved it, the AI^{2}RS agent would record this as (save *this_document*). The AI^{2}RS agent then offers appropriate search assistance to the user based on the particular action and the agent's analysis of the object. The more (a, o) pairs the AI^{2}RS agent records, the more complex the model of the information need.

Assistance offered by the AI^{2}RS agent

Given the scores of user-system interaction issues (Meadow, 1988; Yee, 1991), it was necessary for this stage of the research to narrow the AI^{2}RS agent's assistance. We focused on five user-system interaction issues. We present them along with a description of the assistance that the AI^{2}RS agent provides.

Structuring queries

Searchers have problems properly structuring queries, namely applying the rules of a particular system (Jansen *et al.*, 2000). Searchers have difficulty utilizing Boolean operators (e.g. AND, OR, NOT) and term modifiers (e.g. "+", "-", "!"). The difficulty centers both on when to use the appropriate operator and how to use it on a particular system. For example, some systems require users to capitalize Boolean operators, whereas other systems require no space between a term modifier and the query term. It is clear that searchers could benefit from assistance on when and how to use these operators and modifiers during the search process.

AI²RS assistance

The AI²RS agent uses the IR system query rules to determine the assistance needed in properly structuring the query. Once the user submits a query, the AI²RS agent records this as a (submit query) pair, checks the query's structure based on the system's syntactic rules, and corrects any mistakes.

Spelling

Searchers routinely misspell terms in queries (Yee, 1991), which usually drastically reduces the number of results retrieved. However, it is often difficult to detect these spelling errors because these queries frequently retrieve results from large document collections. For example, when searching on the Web with the query *digitial libraries* (note that digital is misspelled), the searcher will normally retrieve some documents, all of which have digital misspelled. However, the query will not retrieve other documents that relate to *digital libraries* and have the term *digital* spelled correctly. Given that there are some results, the user may not realize that the query contains a spelling mistake. It would be beneficial if the IR system would alert the user of possible misspellings and offer suggested corrections.

AI²RS assistance

A (submit query) pair alerts the agent to check for spelling errors. The AI²RS agent separates the query into terms. It then checks each term using an online dictionary. The AI²RS agent identifies terms that are not in the dictionary and then offers spelling suggestions for these query terms, or it alerts the user that all terms in the query were correctly spelled. The AI²RS agent's current online dictionary is *ispell* (Gorin, 1971), although the AI²RS agent can access any online dictionary using the appropriate application program interface (API).

Query refinement

In general, searchers do not refine their query, even though there may be other terms that relate directly to their information need (Bruza *et al.*, 2000). In fact, studies show that searchers seldom modify their queries, or do so incrementally (Jansen *et al.*, 2000), and then typically only one or two times. To aid in query refinement, the IR system could suggest to the searcher other terms that relate to the information need as specified by the current query. For example, if a query contained the term "mountain", possible suggestions could include hill, mount, peak, or volcano.

AI²RS assistance

With a (submit query) pair and a thesaurus, the AI²RS agent analyzes each query term and suggests synonyms and the contextual definitions of the query terms. The user can utilize these suggestions to refine the query. The AI²RS agent currently uses WordNet (Miller, 1998), but the AI²RS agent can utilize any online thesaurus with the appropriate API.

Managing results

Searchers have trouble managing the number of results (Gauch and Smith, 1993). If there are too many results, they have trouble reducing the number, and they have trouble increasing the number if there are not enough results (Yee, 1991). Generally, user queries are extremely broad, resulting in an unmanageable number of results. Research has shown that few searchers view more than the first 10 or 20 documents from the result list (Silverstein *et al.*, 1999). Additionally, when queries are too narrow and therefore return few or no results, searchers have difficulty modifying the query in order to broaden the search. Assistance in selecting new terms related to the information need would benefit the user.

AI²RS assistance

Using the (submit query) pair and the number of results, the AI²RS agent provides suggestions to improve query structure based on the number of results in the results list. If the number of results is greater than 20, the AI²RS agent provides suggestions to restrict the query. For example, the agent would suggest the use of the Boolean operator AND between terms, providing the suggested query to the user. If the number of results is less than 20, the AI²RS agent provides advice on ways to broaden the query. An example of broadening a query might be to reduce the number of terms in the query or use the Boolean operator OR. A results list length of 20 was selected based on studies such as Hunter (1991) that suggest the majority of users typically view no more than the first 10 or 20 documents.

Relevance feedback

Relevance feedback has been shown to be an effective search tool (Harman, 1992); however, searchers seldom utilize it when offered. Some research has focused on methods to automate this process (Koenemann and Belkin, 1996). In this study, we extend this research by automating the process using term relevance feedback (Mitra *et al.*, 1998).

AI²RS assistance

When a (bookmark document), (print document), (save document), or (copy passage) pair occurs, the AI²RS agent implements a version of relevance feedback using terms from the document or passage object. For example, if the user examines a document from the results list and performs one of the actions (i.e. bookmarking, printing, or saving), the AI²RS agent provides suggested terms from the document that the user may want to add to the query.

Some previous relevance feedback research has focused on automatically selecting terms from documents that the user annotated as relevant. Unfortunately, research has shown that this method fails after a small number of iterations (Witten *et al.*, 1994). The search rapidly narrows, missing the user's broader information need. This deterministic approach to relevant feedback may also miss documents and terms that may be partially relevant (Spink *et al.*, 1998). Additionally, if incorrect information enters the process, the system may experience query drift (Mitra *et al.*, 1998).

The AI²RS agent takes a different approach, utilizing a naïve algorithm. The agent first deterministically removes all terms that have no information value to the searcher, such as stop words, current query terms, all previous query terms by the user, and all previously suggested terms from the document or passage that received the action. From the remaining terms, the agent selects a subset of terms and offers them to the user as possible additional query terms. Although simple in this implementation, we believe that the optimization approach prevents the search from narrowing too rapidly. Naturally, evaluations would need to be done on the appropriate optimization algorithm to most effectively select terms, although Chen *et al.* (1998) have shown the three most well known optimization algorithms all perform equally well.

Alerting the user to assistance

The AI²RS agent communicates with the user via an interface button. If the user selects the AI²RS agent button, the feedback appears in a popup dialog box, along with a brief explanation of each type of assistance. Once the user views what the agent has to offer, the button disappears until the AI²RS agent has more assistance to offer. The user can ignore the feedback with no impact on the normal operation of the interface or IR system. Figure 1 shows the interface, agent dialog box, and text blocks with explanations.

AI²RS agent structure and algorithms

The AI²RS agent has several tasks to perform:

- it checks the user's query and makes structural corrections;
- it checks for spelling errors in the query and offers suggestions;
- it informs the user if the query subject is not in the document collection; and
- it offers suggestions to the user for other query terms.

The structure of the three main modules of the software agent is illustrated in Figure 2. The algorithms for all modules follow Figure 2.

Internet Research: Electronic Networking Applications and Policy
Volume 14 · Number 1 · 2004 · 19-33

Figure 1 AI²RS agent interface and dialog box

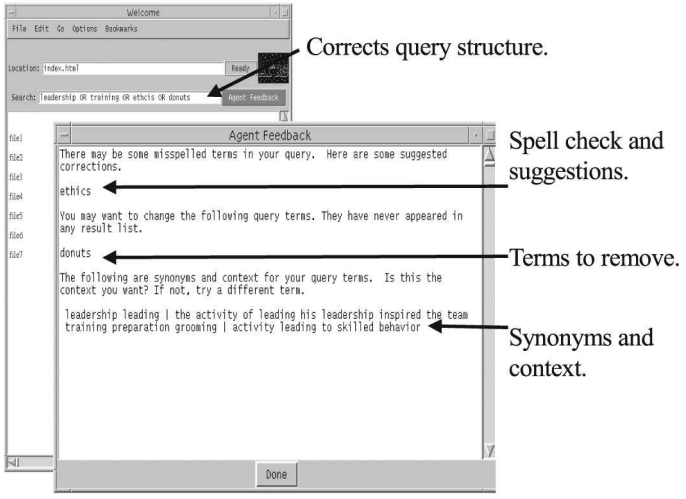
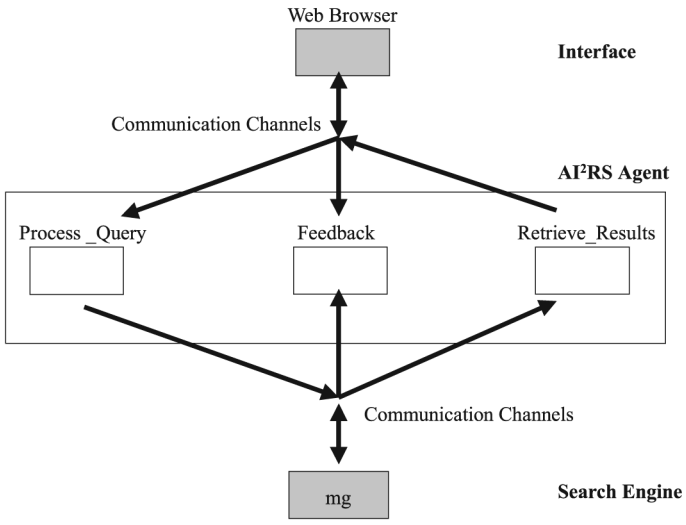


Figure 2 AI²RS modular structure



Process query

The process query module performs the majority of the agent actions. The software agent records user actions in a transaction log of user actions. Specifically, the module records <action><object>. Actions recorded are save, bookmark, print, copy, and no action. Objects are documents or a portion of a document. The process query module stores these actions and objects until the feedback module queries the transaction log and retrieves the actions and the objects.

When a session begins, the user log module checks to see if the user log is empty and if not, it empties it. While the user is logged in, the user log saves the user's actions and source to the file log, adds to the list the action query and the corresponding query for that particular action, and appends to the list if the user bookmarks, saves, prints, or copies a section of text and the source of that action. This process continues as long as the user is logged in. Prior to exiting the browser, the user log file is emptied.

The module then checks and corrects mistakes in query structure based on the user model and known characteristics from the empirical study. It accepts the query as a string and then checks the query for violation of the rules. If violations are found, they are corrected. Four specific rules are enforced. First, Boolean operators AND, OR, and NOT must be capitalized. Second, there must be a space before term modifiers (+ -). Third, there must not be a space after a term modifier (+ -). Finally, there must be a space before a starting parenthesis or quotation mark and a space after the ending parenthesis or quotation mark.

It then converts the query according to the rules of the particular search engine. The query is then sent to the search engine via the appropriate communication channel for that particular system. Next, the agent retrieves the search engine results via the retrieve results module.

Retrieve results

The retrieve results module receives results from the search engine and passes the results to the feedback module. This is second of two modules that must be modified for the particular search engine and graphical user interface. The retrieve results module accepts the results from the search engine. It then reformats the search results to a form suitable for the particular interface. If the interface is specifically designed for this system then this module is not needed. Once the results are returned from the search engine, the feedback module begins its analysis.

Feedback

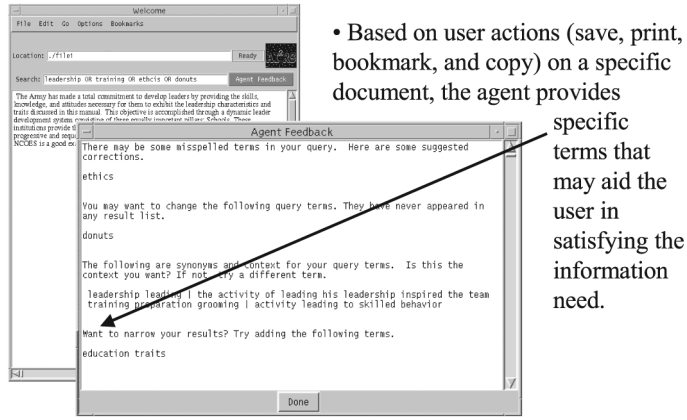
The feedback module provides four types of information to the user. It offers:

Internet Research: Electronic Networking Applications and Policy
Volume 14 · Number 1 · 2004 · 19-33

(1) spelling suggestions for query terms;
(2) terms from the query that have not appeared in the current results list nor in any former results list;
(3) synonyms for query terms along with contextual definitions; and
(4) suggestions to improve query structure.

Once the user looks at a document from the results list, the agent provides relevance feedback on that document and returns a list of terms from the document that the user may want to add to the query. (For all the above, see Appendix.)

Figure 3 Interface for a specific implementation of AI^{2}RS



Implementation interface

Many interfaces, such as most Windows applications, provide information on user actions to other programs or to the system. Some interfaces provide this information via an API. Other interfaces, such as Netscape, provide developers with the actual source code, allowing them to access all user actions within the interface. Other interfaces, such as Internet Explorer, permit wrappers (i.e. small applications that surround another application) to access information from the interface. The amount of information available will vary from interface to interface depending on the method the interface uses to transfer information. The more information that the user provides, the better the implementation of the software agent (Lieberman, 1998).

For this research, the Web browser interface was coded in Tool Command Language/Toolkit (Tcl/Tk), which provides a similar interface to other Web browsers and provides the maximum amount of information for research. Using an interface where the source code was available allowed the agent to monitor all user actions. It also facilitated usability testing and agent integration into the interface. Finally, in the area of testing, it facilitated in the evaluation of the interface and the search engine both without the agent (i.e. turn agent off) and with the agent (i.e. turn agent on). The interface and the agent dialog box are illustrated in Figure 3.

Managing gigabytes search engine

The managing gigabytes or MG (Witten *et al.*, 1994) search engine was the back end chosen for this research. MG was selected as the search engine for this research because of its availability, maturity as an application, and power. MG is a public domain search engine with access to the source code. The MG system is a full-text retrieval system, allowing one to create a database out of a document collection and then do queries to retrieve relevant documents. It is full-text in the sense that every word in the text is indexed, and the query operates only on this index to do the searching. It has been in use for several years, is well documented (Witten *et al.*, 1994), and online help is available (e.g. www.mds.rmit.au/mg). Therefore, it is a reliable and dependable system. MG has been used on large collections such as the Commonwealth Acts of Australia that contains approximately 132Mb of information; it has also been used on larger collections, such as the 3Gb TREC collection. This indicates that MG can handle significant document collections.

Empirical test

To adequately evaluate whether or not the AI^{2}RS agent could improve the performance of an existing IR system, we contrasted the performance of the MG system and a MG-AI^{2}RS system. Surprisingly, given the great amount of research in intelligent IR systems, there have been few evaluations with real users. Most features have been tested in isolation, such as relevance feedback (Koenemann and Belkin, 1996), and query reformulation (Gauch and Smith, 1993). In general, there have been few user studies.

For this evaluation, no MG code was modified to integrate the AI^{2}RS agent with the MG system other than establishing the API. For the test, both systems were installed on a SPARC book 3 running Solaris 2.5. The MG and the MG-AI^{2}RS system ran on the same computer, displayed the identical interface, and utilized the identical document collection. The test was conducted in a usability lab where user interactions with the systems were recorded in a transaction log. The transaction log also recorded the documents returned for each query. Using this information, the precision of both the base and the AI^{2}RS systems could be calculated.

Test population

The subjects for the evaluation were 30 freshman college students (26 males and four females) in their second semester at a four-year undergraduate university. Freshmen at this university must purchase a computer with standard software prior to the beginning of their freshman year, and each dorm room is connected to the university network, providing access to the Internet. Since students must reside in the dorms, all subjects had at least six months of experience using a microcomputer in a networked environment. Additionally, all freshmen are required to read an online newspaper each day and perform online research in the normal preparation for their courses. Therefore, the subjects were familiar with a Web browser, which is an issued software package on their computers. They were also familiar with Web and other IR systems, but were provided no formal classes on searching techniques, either for this experiment or in their normal classes.

Document collection

The document collection utilized for the testing was the Text REtrieval Conference (TREC), Vols 4 and 5, from the TREC information retrieval test collections. The test collection was identical for both systems. Since TREC information is available at: http://trec.nist.gov/data.html, we present only the details pertinent to this research.

We selected Vols 4 and 5 because they consist of a large database of approximately 2Gb, contain a substantial number of approximately 550,000 documents, and represent a diverse collection in terms of document length and subject matter. As shown in Table I, the contents of the collection are broken into five major areas.

Along with a set of documents, each TREC collection also contains both a set of information needs (i.e. topics) that can be answered by some of the documents and the relevance judgments on a particular topic for each document. Each TREC topic has a number, a short title, a brief description, and a narrative describing what is considered to be a relevant document. The topics utilized for this evaluation were "Number 301: International organized crime" and "Number 340: Land mine ban".

For the TREC collections, relevance is defined as:

> If you were writing a report on the subject of the topic and would use the information contained in the document in the report, then the document is relevant (see http://trec.nist.gov/data/testq_eng.html).

A document is judged relevant if any portion, no matter how small, is pertinent. Only binary relevant judgments (i.e. relevant or not relevant) are made. Table II shows the number of relevant documents and the corresponding percentage for the test collection.

There were 474 relevant documents for "Topic 301: International organized crime" in

Table I Division of documents in the collection

Sub-collection	Approx. no. of documents
The Congressional Record of the 103rd Congress	30,000
The 1994 Federal Register	55,000
Selected Financial Times articles from 1992 to 1994	210,000
Data provided from the Foreign Broadcast Information Service	130,000
Selected Los Angeles Times articles from 1989 and 1990	130,000
Total	555,000

Table II Relevant documents in the collection

Topic number	Relevant documents	Percentage of 555,000
301	474	0.09
340	81	0.01
Total	555	0.10

the document collection. There were 81 relevant documents for "Topic 340: Land mine ban". The total number of relevant documents in the collection was 555, representing approximately 0.1 per cent of the collection.

The MG system returns results in ranked relevance order. Two widely accepted metrics of IR system performance are recall and precision. Typically, evaluation of precision requires determining an appropriate place in the results list to satisfy the user's information need, namely a metric known as relative precision. For this research, we were interested in the AI^{2}RS agent's effect on precision within the top 20 ranked documents. Thus, if more than 20 documents were returned for a particular query, documents numbered 21 and higher in the results list were ignored. If the query returned fewer than 20 documents, that number was utilized to calculate precision for that particular query.

Experimental setting

A within-subject design was utilized. This method controls for individual variability (Nielson, 1993). When using within-subject testing, one threat to the validity of results is that learning may occur from the first test to the second, falsely inflating the second test scores. This was not considered a significant issue in this evaluation because the subjects were familiar with the Web browser interface and had previously used search engines. Since the purpose of the evaluation was to see if the introduction of the AI^{2}RS agent improved IR system performance, we decided not to counter balance the systems. All subjects used the MG system first and then the MG-AI^{2}RS system. Since the interface was a Web browser, and all the subjects were accustomed to using a Web browser, it seemed reasonable not to counter balance the systems. In order to control for possible differences in the difficulty of the search topics, the topics were counterbalanced (i.e. each topic was used half the time on each system). For example, for one subject the topic order was Topic 301 and then Topic 340. For the next subject, the topic order would be Topic 340 and then Topic 301.

For individual procedures, each of the subjects was provided a short statement instructing them to search on a given topic in order to prepare a report, which is in line with the definition of relevance judgments for the TREC documents. They were then directed to search as they normally would when conducting online research for a course assignment, such as saving, printing, bookmarking documents they found of interest. The subjects would begin searching using the MG system. The search process continued until the subject determined that there were no relevant documents in the collection or when five minutes had passed. We determined the length of the search session by measuring the length of time it would take to implement a "typical" Web search session, as outlined in (Jansen *et al.*, 2000). All subjects utilized the full five minutes.

The subjects were then given the other topic and the same instructions. The subjects then began the search process utilizing the MG-AI^{2}RS system. Each of the test subjects was notified that the system contained an automatic feature to assist them while they were searching. When the system had searching advice to offer, an assistance button would appear on the browser. The user could access the assistance by clicking the button, or they could ignore the offer of assistance. All subjects utilized the AI^{2}RS agent assistance at least once during the search process. Again, all subjects took the full five minutes for the search. There was no limit to the number of queries a subject could enter.

The users were video taped during the searching process and a transaction log recorded user-system interactions. In order to add further robustness to the analysis, the subjects were instructed to think out loud during the searching process. In analyzing the video, we coded the utterances using verbal protocol analysis (Ericsson and Smith, 1984), specifically the thinking-aloud protocol where the verbalization occurs in conjunction with a task. These coded utterances, along with data from the transaction logs, were utilized to further clarify user interactions with the system. After the search session, each searcher completed a subjective evaluation of the automated assistance. The combination of the protocol analysis, transactions log, and subject evaluations provided a robust data source to conduct our analysis.

We based our evaluation design on the interactive track of the TREC conference. Table III presents a comparison of the two evaluations formats.

Experimental results

There were 135 total queries submitted on the MG system and 130 queries submitted on the AI^{2}RS system. Since many of these queries were duplicates, the precision for only the unique queries was calculated. There were 81 unique queries executed on the unimproved system and 94 unique queries executed on the AI^{2}RS system. The results were analyzed using an independent sample *t*-test and are reported in Table IV.

This analysis revealed a significant difference between the two groups ($t = -3.4187$; $p < 0.01$). This analysis shows that performance measured by precision within the top ten documents (P@10) of the AI^{2}RS system was significantly better than the precision performance of the base system. Other statistics for the number of relevant documents retrieved by the MG system and the MG-AI^{2}RS system, which are noted in ( ), are mean = 0 (1.34), mode = 0 (0), min = 0 (0), and max = 2 (8).

There was no significant difference between P@10 between the topics. Although the content collection contains more relevant documents for Topic 301 versus Topic 340, the MG system ranks output. So, it would be expected that there would be no significant difference between the two topics within the top ten rankings.

Naturally, when there is an increase in precision, there is typically a decrease in recall. However, given that we were concerned with only the first 20 documents, recall was not a reasonable metric for this evaluation. Given that most users, especially on the Web, view only the first few documents (Jansen and Pooch, 2001), the impact of recall for most searches is dramatically less than that of precision. We acknowledge that there are situations where recall is important. In these cases, the agent is adaptable enough to provide assistance to aid in recall. However, we did not address recall in this study.

Number of times that agent was utilized

Table V presents information concerning the user-agent interaction.

The agent was accessed by 30 of the 30 subjects. The mean number of interactions with the agent per subject was 2.27. The most utilized assistance was suggested query refinement (56 per cent of the interactions). The least utilized assistance was relevance feedback terms (3 per cent). The relevance feedback option was activated 26 times by book marking, saving, printing, or copying and pasting. A total of 24 of the subjects implemented at least one of the agent's suggestions during their respective sessions. Six searchers implemented no offered assistance. Most users implemented one of the agent's recommendations per access; however, some users utilized as many as three.

Table III Comparison of TREC-8 and AI^{2}RS evaluations

	TREC-8	AI^{2}RS
Subjects	12	30
Documents	210,158	550,000
Size (MB)	564	2,000
Time (min)	20	5
Relevant (per cent)	1.04	0.01
Relevant (number)	2,178	555
Type	Within	Within

Table IV Precision evaluation results for base and AI^{2}RS systems

System	Precision mean	SD	*t*
Base	0.01	~0.00	
AI^{2}RS	0.13	0.35	–3.4187*

Note: * $p < 0.01$

Post-evaluation survey of workload testing

We also investigated the effect of the automated assistance on the searcher using the ergonomic metric of workload. Workload is a measure of the effort that a certain task requires. Workload is a good comparative measure when changes are introduced into an application. Workload measurements address the question: Will the changes make it more difficult for the user to accomplish the task at hand?

The instrument used for this testing was the subjective workload assessment technique

Table V Number of times agent assistance utilized

Spelling	Query refinement	Terms from relevance feedback	Managing results	Total
9 (12%)	28 (56%)	2 (3%)	20 (29%)	59 (100%)

(SWAT) as outlined in Boff and Lincoln (1988). The SWAT method is a widely utilized workload assessment method developed by the US Air Force originally to assess cockpit workload. A SWAT evaluation has the subject evaluate a system in three areas, which are time, mental effort, and stress. The subject rates the system as a one, two, or three (best to worst) in each area.

Therefore, SWAT evaluation ratings range from three (best possible evaluation) to nine (worst possible evaluation). A rating of three indicates that the application change took no additional time, no additional mental effort, and caused no additional stress. This evaluation is not relative to another system but rather addresses the effect of some change. For this evaluation, the SWAT evaluated the effect of the agent on the subject's search process.

After utilizing the AI^{2}RS system, each subject completed a SWAT evaluation. The result of the workload analysis is presented in Table VI.

The AI^{2}RS agent received a mean score of 5.37, indicating that the introduction of the AI^{2}RS agent to the existing application caused some additional workload for the user. The median SWAT evaluation was six. Based on evaluation of the videotapes, it appears that the majority of the additional workload was a result of the manner in which the agent was providing the feedback. Some of the users seemed compelled to review the agent feedback whenever feedback was available. The reading of the agent's feedback also appeared to take some additional mental effort and caused some stress in some subjects. These observations point to possible corrections in future versions and enhancements of the AI^{2}RS system.

Table VI Results of SWAT evaluation

	SWAT		
Category	Average	SD	Median
Time	1.97	0.67	2
Mental effort	1.80	0.48	2
Stress	1.60	0.50	2
Total	5.37	1.13	6

Overall, the AI^{2}RS system fared well in the workload evaluation.

Conclusion and significance of research

This research integrated a software agent with an existing IR system in order to improve searching performance. The software agent offers assistance with query structure, spelling, query refinement, managing results, and relevance feedback. The base IR system and agent-system were evaluated with a 30 subject within group evaluation. The results indicate that assistance during the search process can improve searching performance as measured by precision.

The research demonstrates that:

- the technique utilized to model the user's information need permits rapid modeling of a user's information need within a single IR session;
- a software agent that provides searching assistance can be developed that is relatively system and platform independent; and
- utilizing the (a, o) pair model, the AI^{2}RS agent provides this assistance without any additional actions by the user during the search process.

The assistance is derived solely from the normal actions of a searcher during the session. Third, the evaluation results demonstrate that integration may be a feasible avenue of research for improving IR systems.

The next version of the AI^{2}RS agent is implemented on an IBM-compatible platform, in the Windows operating environment and Microsoft Internet Explorer as the interface. The AI^{2}RS agent integration occurs via API wrappers to the browser. In future research, we will increase the type of searching assistance provided and then measure if there is a corresponding increase in IR system performance. Although one might assume that as searching features increased so would system

performance, Saracevic and Kantor (1988) have shown that this occurrence is not necessarily the case. Paradoxically, it appears that the increased use of advanced features and even increased user preparation time may have an adverse effect on search performance. Identifying an appropriate level of assistance is one aim of future research.

For our preliminary evaluation, we used only two topics and limited the session duration (five minutes). Although advantageous as an initial evaluation, a more robust evaluation (e.g. greater selection of topics, longer session length, etc.) must be conducted to obtain statistically significant results concerning changes in precision. In our evaluation of the next version of the AI^{2}RS agent, we will isolate the different types of assistance during the search process to measure the effect of each. We aim to rank the assistance in terms of greatest to least impact on improving the search process. Once algorithmic improvements are made, we plan to test our system against other methods of automated assistance.

Moreover, we are refining the (a, o) pair model of user system interaction. Jansen *et al.* (1999) classified search-user interactions on Web systems by isolating sequencing of action. Spink *et al.* (2000) also examined the sequence of users' actions on Web systems, focusing on relevance feedback. Building on this research, we are working to improve the (a, o) pair model to identify session patterns in addition to individual search actions during a session. The follow-on step is to take the (a, o) pair model to multiple sessions for an individual user. We also want to isolate possible differences in the meaning of individual or sequences of actions.

There are several algorithm methods to pursue. The method we utilized can be refined. In terms of relevance feedback, we would like to improve the optimization algorithm for term relevance feedback by utilizing simulated annealing to select the terms for suggestion. There are several factors worth considering in choosing a subset of terms from the relevant documents. Other optimization techniques, such as neural nets and genetic algorithms, have been widely utilized in IR. However, there seems to be little utilization of simulated annealing (Jansen, 1997), even though Chen *et al.* (1998) have shown that simulated annealing is at least as effective as neural nets and genetic algorithms. Simulated annealing has the advantage of being relatively simpler to implement. The difficulty with simulated annealing is determining what the cost factors are in the search process.

Overall, the results of the research conducted so far are promising. They indicate that the computer technology and knowledge of user-searching techniques currently exists to make IR systems more active assistants in the search process. Existing IR systems can be enhanced to assist users in finding the information they desire, forgoing the need to develop entirely new systems. Hopefully, this research is another step toward improving the search process and, thereby, leading to more efficient utilization of the tremendous amount of information searchers must confront everyday.

References

Boff, K.R. and Lincoln, J.E. (1988), *Engineering Data Compendium: Human Perception and Performance*, Harry G. Armstrong Aerospace Medical Research Laboratory, Wright-Patterson AFB, Dayton, OH.

Brajnik, G., Guida, G. and Tasso, C. (1987), "User modeling in intelligent information retrieval", *Information Processing & Management*, Vol. 23 No. 4, pp. 305-20.

Bruza, P., McArthur, R. and Dennis, S. (2000), "Interactive Internet search: keyword, directory and query reformulation mechanisms compared", *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Athens, Greece, 24-28 July*, pp. 280-7.

Card, S. and Moran, T. (1986), "User technology: from pointing to pondering", in Baecker, R.M., Buxton, W. and Grudin, J. (Eds), *Readings in Human-Computer Interaction: Toward the Year 2000*, 2nd ed., Vol. 2, Morgan Kaufman, San Francisco, CA, pp. 20-33.

Chen, H.C. and Dhar, V. (1991), "Cognitive process as a basis for intelligent retrieval-systems design", *Information Processing and Management*, Vol. 27 No. 5, pp. 405-32.

Chen, H.C., Shankaranarayanan, G., She, L.L. and Iyer, A. (1998), "A machine learning approach to inductive query by examples: an experiment using relevance feedback, ID3, genetic algorithms, and simulated annealing", *Journal of the American Society for Information Science*, Vol. 49 No. 8, pp. 693-705.

Chen, Z.X., Meng, X.N., Fowler, R.H. and Zhu, B. (2001), "Features: real-time adaptive feature and document learning for Web search", *Journal of the American Society for Information Science*, Vol. 52 No. 8, pp. 655-65.

Croft, W. and Thompson, R. (1986), *I3: A New Approach to the Design of Document Retrieval Systems*, COINS Technical Report 87-58, University of Massachusetts, Amherst, MA.

De Bra, P. and Calvi, L. (1998), "AHA: a generic adaptive hypermedia system", *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia, Pittsburgh, PA, 20-24 June*, pp. 283-4.

Ericsson, K.A. and Simon, H.A. (1984), *Protocol Analysis: Verbal Reports as Data*, MIT Press, Cambridge, MA.

Fox, E. (1987), "Development of the CODER system: a testbed for artificial intelligence methods in information retrieval", *Information Processing and Management*, Vol. 23 No. 4, pp. 341-66.

Gauch, S. and Smith, J. (1993), "An expert system for automatic query reformulation", *Journal of the American Society for Information Science*, Vol. 44 No. 3, pp. 124-36.

Gorin, R.E. (1971), *Developer of ispell*, available at: http://fmg-www.cs.ucla.edu/geoff/ispell.html (accessed 12 February 2000).

Green, T. and Benyon, D. (1996), "The skull beneath the skin: entity-relational models of information artifacts", *International Journal of Human-Computer Studies*, Vol. 44 No. 6, pp. 801-28.

Harman, D. (1992), "Relevance feedback revisited", *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Copenhagen, Denmark, 21-24 June*, pp. 1-10.

Herlocker, J.L., Konstan, J.A. and Riedl, J. (2000), "Explaining collaborative filtering recommendations", *Proceedings of ACM 2000 Conference on Computer Support Collaborative Work, Philadelphia, PA, 2-6 December*, pp. 241-50.

Hunter, R. (1991), "Successes and failures of patrons searching the online category at a large academic library: a transaction log analysis", *Research Quarterly*, Vol. 30, pp. 395-402.

Jansen, B.J. (1997), "An information retrieval application for simulated annealing", *Proceedings of the 2nd ACM Conference on Digital Libraries, Philadelphia, PA*, pp. 259-60.

Jansen, B.J. and Pooch, U. (2001), "Web user studies: a review and framework for future work", *Journal of the American Society of Information Science and Technology*, Vol. 52 No. 3, pp. 235-46.

Jansen, B.J., Spink, A. and Saracevic, T. (1999), "The use of relevance feedback on the Web: implications for Web IR system design", *Proceedings of the 1999 World Conference of the World Wide Web and Internet, Honolulu, HI*.

Jansen, B.J., Spink, A. and Saracevic, T. (2000), "Real life, real users, and real needs: a study and analysis of user queries on the Web", *Information Processing and Management*, Vol. 36 No. 2, pp. 207-27.

Jansen, B.J., Spink, A., Bateman, J. and Saracevic, T. (1998), "Real life information retrieval: a study of user queries on the Web", *SIGIR Forum*, Vol. 32 No. 1, pp. 5-17.

Kahle, B. (1999), *Alexa Internet Press Releases* (Web site). Alexa Internet, available at: www.alexa.com/company/recent_articles.html (accessed 12 February 2000)

Kamba, T., Bharat, K. and Albers, M. (1993), "*The Krakatoa Chronicle*: an interactive personalized newspaper on the Web", *The World Wide Web Journal*, Vol. 1.

Koenemann, J. and Belkin, N. (1996), "A case for interaction: a study of interactive information retrieval behavior and effectiveness", *Proceedings of the CHI '96 Conference on Human Factors in Computing Systems, Vancouver, BC, Canada, 13-18 April*, pp. 205-12.

Lawrence, S., Giles, C.L. and Bollacker, K. (1999), "Digital libraries and autonomous citation indexing", *IEEE Computer*, Vol. 32 No. 6, pp. 67-71.

Lieberman, H. (1995), "Letizia: an agent that assists Web browsing", *Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, Quebec, Canada, August*, pp. 924-9.

Lieberman, H. (1998), "Integrating user interface agents with conventional applications", *Proceedings of the 1998 ACM International Conference on Intelligent User Interfaces*, pp. 39-46.

Maes, P. (1994), "Agents that reduce work and information overload", *Communications of the ACM*, Vol. 37 No. 7, pp. 30-40.

Marchisio, L., Ronco, E. and Saracco, R. (1993), "Modeling the user interface", *IEEE Communication Magazine*, Vol. 31, pp. 68-74.

Meadow, C.T. (1988), "OAKDEC, a program for studying the effects on users of a procedural expert system for database searching", *Information Processing and Management*, Vol. 23 No. 4, pp. 449-57.

Middleton, S.E., Roure, D.C.D. and Shadbolt, N.R. (2001), "Capturing knowledge of user preferences: ontologies in recommender systems", *Proceedings of 1st International Conference on Knowledge Capture, Victoria, BC, Canada*, pp. 100-7.

Miller, G. (1998), *WordNet: An Electronic Lexical Database*, MIT Press, Cambridge, MA.

Millsap, L. and Ferl, T. (1993), "Search patterns of remote users: an analysis of OPAC transaction logs", *Information Technology and Libraries*, Vol. 11 No. 3, pp. 321-43.

Mitra, M., Singhal, A. and Buckley, C. (1998), "Improving automatic query expansion", *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne, Australia, 24-28 August*, pp. 206-14.

Nielson, J. (1993), *Usability Engineering*, AP Professional, New York, NY.

Oddy, R.N. and Balakrishnan, B. (1991), "PTHOMAS: an adaptive information retrieval system on the connection machine", *Information Processing and Management*, Vol. 27 No. 4, pp. 317-35.

Peters, T. (1993), "The history and development of transaction log analysis", *Library Hi Tech*, Vol. 42 No. 11, pp. 41-66.

Ruthven, I., Laimas, M. and von Rijsbergen, K. (2001), "Empirical investigation on query modification using abductive explanations", *Proceedings of 24th Annual International ACM SIGIR Conference on Research and*

Development in Information Retrieval, New Orleans, LA, pp. 181-9.

Saracevic, T. (1996), "Modeling interaction in information retrieval (IR): a review and proposal", *Proceedings of the 59th American Society for Information Science Annual Meeting, Baltimore, MD, 19-24 October*, pp. 3-9.

Saracevic, T. and Kantor, P. (1988), "A study of information seeking and retrieving. II. Users, questions and effectiveness", *Journal of the American Society for Information Science*, Vol. 39 No. 3, pp. 177-96.

Silverstein, C., Henzinger, M., Marais, H. and Moricz, M. (1999), "Analysis of a very large Web search engine query log", *SIGIR Forum*, Vol. 33 No. 1, pp. 6-12.

Sparck-Jones, K. and Willett, P. (Eds), (1997), *Readings in Information Retrieval*, Forward, Morgan Kaufman, San Francisco, CA.

Spink, A., Greisdorf, H. and Bateman, J. (1998), "From highly relevant to not relevant: examining different regions of relevance", *Journal of the American Society for Information Science*, Vol. 34 No. 5, pp. 599-621.

Spink, A., Jansen, B.J. and Ozmutlu, C. (2000), "Use of query reformulation and relevance feedback by Web users", *Internet Research – Electronic Networking Applications and Policy*, Vol. 10 No. 4, pp. 317-28.

Spink, A., Jansen, B.J., Wolfram, D. and Saracevic, T. (2002), "From E-sex to E-commerce: Web search changes", *IEEE Computer*, Vol. 35 No. 3, pp. 107-11.

Spink, A., Ozmutlu, S., Ozmutlu, H.C. and Jansen, B.J. (2002), "US versus European Web searching trends", *SIGIR Forum*, Vol. 32 No. 1, pp. 30-7.

Witten, I., Moffat, A. and Bell, T.C. (1994), *Managing Gigabytes: Compressing and Indexing Documents and Images*, Van Nostrand Reinhold, New York, NY.

Wooldridge, M., Jennings, N.R. and Kinny, D. (1999), "A methodology for agent-oriented analysis and design", *Proceedings of the 3rd Annual Conference on Autonomous Agents, Seattle, WA*, 1-5 May, pp. 69-76.

Yee, M. (1991), "System design and cataloging meet the user: user interfaces to online public access catalogs", *Journal of the American Society for Information Science*, Vol. 42 No. 2, pp. 78-98.

Appendix. Agent pseudo-code

Process query

On user log on, check and clear user log
While user log on
  For each user action {enter, save, bookmark, print, copy, no action} on object {document, portion of document, query}, Save <action><object>.
  For each {enter query}
    Check query and corrects mistakes in query structure
    Send query to search engine
    For each query term
      Check for spelling errors, record spelling correction or no spelling errors message
      Check for synonym, record synonyms and contextual definitions
      Check for terms that neither appear in the current results list nor in any former results list, record such terms
    For each {save, print, bookmark: document} or {copy portion of document}
      Select two terms from either document or portion of document, record

Retrieve results

Accepts the results from the search engine
Reformat the search results for interface

Feedback

On click_Agent_Feedback Button
  Provide suggested spelling correction or no spelling errors message
  Provide synonyms and contextual definitions
  Provide list of terms that do not appear in the current results list nor in any former results list
  Provide relevance feedback terms