Automatic Complexity Management; Personalised Document Retrieval from the World Wide Web

Martin Kermit[†], Martin Thorsen Ranang and Harald Nordgård-Hansen[‡]
School of Computer Sciences, Østfold College, Norway

The quest for useful information on the World Wide Web is in many cases tedious and time consuming. Intelligent agents have become common in information searching and have proven to help users to find what they need on the Web. This report presents a system which automatically presents new, relevant Web pages to its users based on each user's personalised usage pattern. The article describes the theory behind the intelligent agent called Exquiro, how it works and its implementation. The article concentrates on important parts of the theory behind it rather than experiments. An overview of Exquiro and a presentation of the individual parts of the system and how they interact with each other is presented and discussed. The article also addresses some of the issues known to be improvable.

Introduction

In the later years, the technology to distribute electronic information has reached the general public. The easy access to the Internet has made the World Wide Web a primary source of information. Unfortunately, the vast amount of information on the Web is by no means organised for optimal retrieval. The Web can easily be compared to a great library where books, tapes and magazines are thrown in heaps with no system indexing the entries. Even though familiar search engines like AltaVista, HotBot, Lycos, Fast[1] and others are improving to give better and more relevant search results, collecting information online is still often a time consuming task.

The inexperienced user of the Web who is searching for information will in many cases find either too much information or no information at all. Unfamiliarity with search tactics for specific search engines or failure to rephrase badly formulated queries are problems that often face professional users of the Web as well. Combined with the fact that popular search engines cover different parts of the Web, the power of these engines is diluted. Market studies indicate that in order to survive into the information jungle, users of the Web almost exclusively resort to search engines and Web-portals or repositories[2] to find the information they seek (Marchiori, 1997). Overall, the ability to search is not enough alone (Borgman, 1986; Fenichel, 1981).

To address this problem, intelligent search assistance is needed, and a novel design of a system that is able to give such assistance is proposed in this article. First is given an overview of the different aspects of search assistance, followed by a brief description of the mechanisms of the proposed scheme. This is followed by the complete documentation of the system, containing a description of the proxy server, details of the textual representation, a description of how similarity between documents are measured and an outline of the presentation of new, relevant Web pages to the user. In the end is the conclusion of this article together with future guidelines for the work.

Search Assistance and Web Agents

Software tools referred to as intelligent Web agents seems to be a remedy to problems regarding information retrieval from the Web. The quest for adequate information on the Web via individual search agents relies on what knowledge is available about the user from previous Internet activity. This knowledge should be used to construct individual user patterns. Based on such user patterns, the intelligent Web agent should be able to discover and ferret out the most relevant information from the Web to its users.

Earlier Work

The growing interest for aided Web navigation has led to the development of a variety of intelligent Web tools, like the Web agent described by Haverkamp and Gauch (1998) which also gives an extensive overview to intelligent agents in general. An early example of a practical system is the agent called 'WebWatcher' designed by Armstrong et al. (1995). The WebWatcher provides interactive advice to the Web-browser Mosaic. Another architecture was implemented in the system called 'Letizia' by Lieberman (1995). This agent performed limited searches on the Web while the user was browsing, trying to recommend the user where to go next. The Dublin Core Metadata Initiative[3] is another attempt to

[†]Author to whom correspondence should be addressed, e-mail address: martin.kermit@hiof.no
[‡]Current address: LinPro AS, Norway, http://linpro.no/

[1] http://www.altavista.com, http://hotbot.lycos.com, http://www.lycos.com,www.alltheweb.com

[2] A repository is a topic-based collection of links maintained by a human.

[3] http://purl.org/dc/

index Web documents both manually and automatic.

Other studies indicate that the hyperlink structure can be valuable for locating information on the Web. If there is a hyperlink from page *A* to page *B*, it can be assumed that the author of *A* recommends page *B* (Brin & Page, 1998). Intelligent agents exploiting the relation between the hyperlink structure of Web documents are described in (Marchiori, 1997; Dean & Henzinger, 1999) among others. The search engine known as Google[4] is based on this principle.

The focus in the present research will rely on the concept of textual similarity, like the WHIRL system proposed by Cohen (1998b,1998a), rather than the hyperlink structure. Text representation has been widely studied (Monson, 1997; Croft & Lewis, 1987; Lewis, 1992) and is well suited for the purposes of the Web agent presented here, which in the following is named Exquiro.

Requirements of Web Agents

Cheung et al. (1998) have given a description of the most important qualities of an intelligent Web agent with the capability of learning the behaviour of information users. According to this scheme, the goal of the agent is to discover the most relevant information to its users under the following requirements[5]:

1. The system must be compatible with most Web browsers. There should be no extra requirements for the browsers than the standard HTTP protocol.
2. The system should maintain a database with a full-text index on retrieved documents. The user access patterns should be extracted from these saved documents.
3. The system should learn both the access pattern of the user and when the information source is updated with new information. The latest version should be available before the user requests it.
4. The topics that the users are interested in should be discovered automatically. The system should also be able to adapt each users change of interest over time. This particular knowledge is used when the system tries to discover new relevant information for the user.
5. The system should avoid use of excessive network traffic and thus make efficient use of network resources. Searches for multiple users should be combined to reduce the total number of searches.

System Overview

Exquiro is based on the scheme outlined by Cheung et al. above and is unique in taking all five requirements into account. To meet these requirements, Exquiro needs three major components.

The first component is a robust and transparent mechanism for logging each individual user's Web browsing patterns and for getting access to the full text of the Web documents accessed. In order to achieve this, Exquiro depends on its users to be using a controlled proxy server. The proxy server keeps a cache of files and acts like a buffer to the Web. In this way, all communication is kept via the standard HTTP protocol, and requirement (1) is met. The cache is configured to log all the universal resource locators (URL) (Berners-Lee, Masinter, & McCahill, 1994) that are accessed by each user of the proxy server, fulfilling requirement (2). Requirement (3) is also automatically fulfilled since the proxy server always holds the latest downloaded version of a document accessed by any user of the system. The parts of Exquiro which are connected to the proxy server is outlined by a dashed box in Figure 1.

The second component is the extraction and classification of the users' access patterns. Each document accessed by a user is converted into a vector. The document vectors are used to find the similarity value between two documents. The assumption is that if two documents have a high similarity value and a user has already seen one of them, then the other document will probably be of interest to the user too. The set of similarity values for a given user is then used to extract a set of topics or groups of similar documents, which can be used to search for further documents of interest. This meets requirement (4). Since all the comparisons are made between documents that are already localised to the cache of the system, requirement (5) is also taken care of.

The third and final component is the selection and presentation of the search results to the user. For each user, a set of URLs that are both unread and likely to be of interest to the user is generated by comparing all documents in the cache to the interest groups of the user. These sets are stored in the database. When a user visits his personal page on Exquiro, the proposed URLs are retrieved from the database, formatted as HTML and presented to the user.

The Squid Proxy Server

Squid is an open source Unix-based server software package that among other services, also provides high performance caching for HTTP clients.

In brief, the Web browser does not directly communicate with the Web server holding a requested document. Instead, the browser requires the proxy server to fetch the document for it. When a new page is demanded, the data is downloaded by the proxy server and forwarded to the client. At the same time, the data is stored (for a longer or shorter period) in the server's disk-based cache. When the same Web page again is revisited by a client using the same proxy server, the proxy server can choose to supply the document from its cache, rather than refetching it over the net. A discussion of the strategies involved in figuring out when a document can safely be served from the cache is beyond the scope of this research, and thus omitted.

When using Squid as a proxy cache server (often called a Web accelerator), it is possible to configure the system (browser and server) in such a way as to let the server create a log of all the documents that it is fetching on behalf of each client. This can be done by various methods, ranging from assuming that a single IP source address for a request belongs

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

[5] In the original work, the requirements of Cheung et al. appears in a different order. The reordering is due to the presentation of Exquiro later.

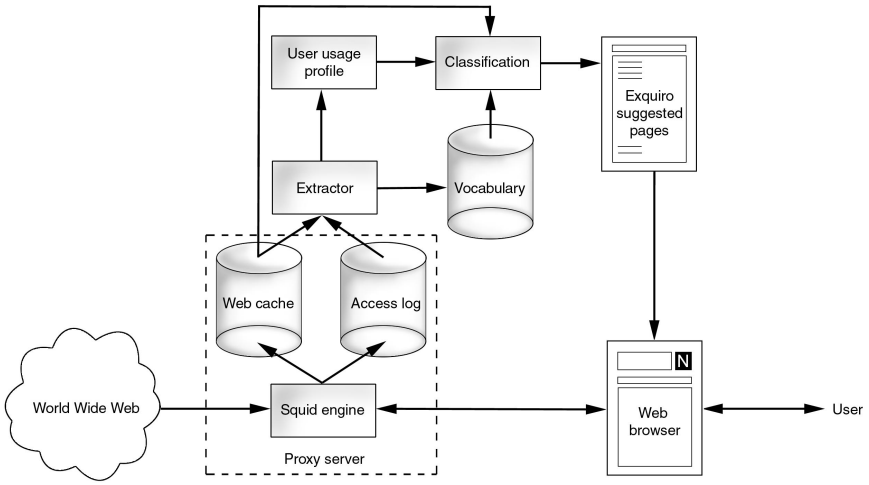

Figure 1. Simplified overview of Exquiro. Web pages on the Internet are retrieved by the user via the proxy server. The proxy server keeps log files monitoring the Internet traffic as well as storing copies of Web documents in a database. Another database generates a vocabulary of words retrieved from the documents. From this database, combined with the user access log files, Web documents are compared to each other and links to new documents in the Web cache are presented to new users likely to be interested. The dashed box indicates the parts of the proxy server.

to a single user in a given time frame, to using authenticated access to the proxy-server for all users.

In addition to the access log, the server creates a log file of documents as they are stored in the cache and when they later are deleted. Using these two logs, it is possible to maintain a database of all the different Web pages visited by the different users, and the corresponding file in the cache containing this document.

Document Representation

For a user access pattern to be generated, the informative content of the documents that the user recently has accessed from the Web has to be used. These recently[6] visited Web documents are directly available from the proxy server. Thus, this pattern has to be created on the basis of Web documents, or more specific, directly from HTML-files. Since HTML-documents consist of strings of characters a mapping into a suitable mathematical representation for further processing is needed.

The first step in the process of creating a suitable representation for HTML-files, is a simple preprocessing of the documents to remove HTML-specific tags and images from the Web pages. As a result, only the informative textual content and a list of weighted keywords from title tags are stored from each document.

The next step is to remove words that are extremely common in general and may be considered as noise to the input data.

When this is done, the next step is to find the smallest common denominator of each word, in a process called word stemming. After these operations are applied to the document, it is well suited for mathematical calculations. It contains less noise, less unique words but it still represents the information. This process can be viewed as an amplification of the informative content combined with a noise reduction and will be described in separate subsections in the following.

Removing Stop Words

The removal of stop words is a technique commonly used when working with document indexing and retrieval. This is done to improve the accuracy of the results and to reduce the number of redundant calculations (Salton, 1989).

In most texts there are some words that appear too frequently to be of any use for representing a document. Exquiro contains a list of such words, maintained manually by inspecting those terms with the highest frequency in all the documents in the system's cache.

The Vector Space Model

For each single user, there exist N documents in the Web cache stored by the proxy server. These N documents represent a set, denoted D, containing all recently visited documents for a single user. Further analysis will refer to the set of documents for a single user only.

Exquiro's text classification algorithms are based on a vector space model (VSM) for text representation (Joachims, 1999). The vector space model requires an existing vocabulary, T, that consists of all known words. New words are

[6] The setup of the proxy server determines the time scale that is referred to as recently. This is usually set to be the last week or month.

added as they appear in the Web cache. By this model a document is considered a vector denoted $\vec{d}$ with elements d_t of integers that corresponds to terms $t \in T$. With this notation, $\vec{d} \in \mathbb{R}^{|T|}$, where each dimension is a count of occurrences of a specific word, and $|T|$ the cardinality of T.

Word Stemming

The number of elements in the document vector will always be the same as the number of known words by the system up to date. Since the number of known words is very large, the calculations on such vectors are computationally expensive. To reduce the word space dimensionality, all words are stemmed[7] according to the Porter stemming algorithm (Porter, 1980). Applying the stemming algorithm on a word results in an atomic word stem where the suffix has been removed. Such word stems are referred to as *terms*. By using the stemming algorithm, the space dimensionality of known words is reduced, since many words are mapped to a single term. An example are the words *computing*, *computer* and *computes* which all are mapped to the term *comput*.

It can be argued against the Porter stemming algorithm that it sometimes is being to aggressive in conflation, like "policy"/"police" and "executive"/"execute" are conflated. Sometimes the algorithm is missing others, i.e. "European"/"Europe" and "matrix"/"matrices" are not conflated (Xu & Croft, 1998).

Some improvements to the Porter stemming algorithm has been performed, but the principles remain the same as these have proven to give good results for data mining in electronic text collections (Walker & Truman, 1997).

Term Frequency / Inverse Document Frequency

After the removal of stop words and the word stemming has been performed, the resulting word stems, or terms, t, are used to represent the document vectors. The document representation used is the so called term frequency / inverse document frequency (TF-IDF) weighting scheme proposed by Salton (Salton, 1989). The TF-IDF defines a "bag-of-words"-representation of documents (Joachims, 1997) described by equation (1) for each vector term d_t,

$$d_t = \log(\mathrm{TF}_{\vec{d},t} + 1) \cdot \log(\mathrm{IDF}_t). \tag{1}$$

Here, $\mathrm{TF}_{\vec{d},t}$ is the number of times that the term t occurs in document $\vec{d}$ and IDF_t is the inverse document frequency for a term t. The inverse document frequency for t, is given by

$$\mathrm{IDF}_t = \frac{N}{n_t} \tag{2}$$

where N is the total number of documents and n_t is the total number of documents that contain the term. A single document is thus represented as

$$\vec{d} = [d_1, ..., d_t, ..., d_{|T|}]^t, \qquad t \in [1, |T|] \tag{3}$$

The total number of documents are stored in a matrix **D** such that

$$\mathbf{D} = [\vec{d}_1; ...; \vec{d}_n; ...; \vec{d}_N], \qquad n \in [1, N]. \tag{4}$$

The matrix **D** is thus the formal representation of the set D of N documents that a user has visited recently and stored as Web cache in the proxy server.

Similarity and Correlation between Documents

The documents are now represented as vector quantities, and the angle between two document vectors can then be used to identify how much they have in common. The inner product contains the cosine to this angle, and is defined between $\vec{d}_n = [d_{n1}, ..., d_{nt}, ..., d_{n|T|}]$ and $\vec{d}_m = [d_{m1}, ..., d_{mt}, ..., d_{m|T|}]$ as

$$\langle \vec{d}_n, \vec{d}_m \rangle = \sum_{t \in T} d_{nt} d_{mt}. \tag{5}$$

The scalar quantity, $\langle \vec{d}_n, \vec{d}_m \rangle$, is then a measure of how much two documents, $\vec{d}_n$ and $\vec{d}_m$, are related. The similarity value between the two documents is the normalised inner product given by (Cohen & Hirsh, 1998),

$$\mathrm{Sim}[\vec{d}_n, \vec{d}_m] = \frac{\langle \vec{d}_n, \vec{d}_m \rangle}{||\vec{d}_n|| \cdot ||\vec{d}_m||}. \tag{6}$$

The quantity $\mathrm{Sim}[\vec{d}_n, \vec{d}_m]$ thus gives a measure of how two documents are related to each other. Since $\mathrm{Sim}[\vec{d}_n, \vec{d}_m] \in [0, 1]$, the value 0 occurs if the documents $\vec{d}_n$ and $\vec{d}_m$ have no words in common, and 1 if the two documents are identical. Because the similarity value always falls in between two extremes, an activation function, $f(x)$, is applied to decide whether there exists a correlation or not. In this case, a hard limiting threshold function, given by

$$f(x) = \begin{cases} 0 & \text{iff} \quad x < \theta \\ 1 & \text{iff} \quad x \geq \theta \end{cases} \tag{7}$$

is used. The threshold function is used to measure the final correlation between two documents, and could in general be any increasing function of x, depending on the desired weighting scheme. In this work, only the hard limiting threshold function is used.

The correlation provided by the threshold function is given by the correlation operator $\hat{\mathrm{C}}$,

$$\hat{\mathrm{C}}(\vec{d}_n, \vec{d}_m) = f(\mathrm{Sim}[\vec{d}_n, \vec{d}_m]). \tag{8}$$

In this case, using the hard limiting threshold function for $f(x)$, $\hat{\mathrm{C}}$ becomes a binary operator indicating whether two documents are related or not.

[7] stemming is also known as truncation

	$\vec{d}_1$	$\vec{d}_2$	$\vec{d}_3$	$\vec{d}_4$	$\vec{d}_5$	$\vec{d}_6$	$\vec{d}_7$	w
$\vec{d}_1$	-	0	1	1	0	0	0	2
$\vec{d}_2$	0	-	0	1	0	1	0	2
$\vec{d}_3$	1	0	-	0	0	0	0	1
$\vec{d}_4$	1	1	0	-	0	0	1	3
$\vec{d}_5$	0	0	0	0	-	1	0	1
$\vec{d}_6$	0	1	0	0	1	-	0	2
$\vec{d}_7$	0	0	0	1	0	0	-	1

Table 1
Correlation matrix for the 7 documents in the example. The rightmost column indicates the total weight for each document.

	$\vec{d}_1$	$\vec{d}_2$	$\vec{d}_3$	$\vec{d}_4$	$\vec{d}_5$	$\vec{d}_6$	$\vec{d}_7$	w
$\vec{d}_1$	-	0	-	-	0	0	0	0
$\vec{d}_2$	0	-	0	-	0	1	0	1
$\vec{d}_3$	-	0	-	-	0	0	0	0
$\vec{d}_4$	-	-	-	-	-	-	-	0
$\vec{d}_5$	0	0	0	-	-	1	0	1
$\vec{d}_6$	0	1	0	-	1	-	0	2
$\vec{d}_7$	0	0	0	-	0	0	-	0

Table 2
Updated correlation matrix for the 7 documents in the example after all correlation to Group G_1 is removed.

Categories of Documents

Correlation between documents is the main feature to create clusters or groups of documents related to each other. For all documents $\vec{d}_n$ the correlation between itself and all other documents is measured and added up, giving a corresponding weight, w_n given by

$$w_n = \sum_{n \neq m} \hat{\mathrm{C}}(\vec{d}_n, \vec{d}_m). \tag{9}$$

Thus, for a set of N documents, a weight vector $\vec{w}$ is formed where each vector element w_n represents how many other documents in the set are related to $\vec{d}_n$, $\vec{w} = [w_1, ..., w_n, ..., w_N]^t$.

From $\vec{w}$ the largest element, w_L is extracted. This element is the weight that corresponds to the document $\vec{d}_L$ with the largest sum of related documents in the set D, represented as the matrix **D**,

$$\vec{d}_L = \{\vec{d}_n | \max_n w_n\}, \qquad n = 1, 2, ..., N. \tag{10}$$

This document forms the first group or category of documents, G_1, given by

$$G_1 = \{\vec{d}_m | \hat{\mathrm{C}}(\vec{d}_L, \vec{d}_m) = 1\}. \tag{11}$$

The document $\vec{d}_L$ is for the further referred to as the group leading document for group G_1. For P new groups $G_2, G_3, ..., G_P$ to be generated, existing weights within the defined group G_1 has to be removed and recalculated. The new weights are iteratively updated by

$$w_m = \sum_{n \neq m} \{\hat{\mathrm{C}}(\vec{d}_n, \vec{d}_m) | \{\vec{d}_n, \vec{d}_m\} \notin G_p\}, \tag{12}$$

given that $G_p, p = 1, 2, ..., P$ represents the p groups registered at the current iteration. This scheme is either repeated up to a predefined number of groups, P, that are desired from the set of N documents or until all weights are removed. In the latter case, all documents will be categorised into at least one group, thus

$$D = G_1 \cup G_2 \cup, ..., \cup G_P = \bigcup_{p=1}^{P} G_p. \tag{13}$$

A Categorisation Example

To illustrate the classification procedure, a small example given that 7 documents exist in the Web cache is given. The topics of the 7 documents are as follows:

- $\vec{d}_1$ - How to play the guitar for beginners.
- $\vec{d}_2$ - A new computer program for midi sequencing.
- $\vec{d}_3$ - News group for guitar players.
- $\vec{d}_4$ - Various music instruments.
- $\vec{d}_5$ - Latest news in processor technology.
- $\vec{d}_6$ - An online store selling cheap computers.
- $\vec{d}_7$ - The home page of a symphony orchestra.

Assume that the correlations have been measured between the 7 documents according to the outlined description giving the correlation matrix shown in Table 1. The correlation matrix is symmetric, indicating with 1 or 0 whether there exists a correlation or not. The diagonal elements cancel out, since self-correlation of documents is not counted. The rightmost column adds up the number of documents that each document, $\vec{d}_n$ is correlated to, giving the corresponding weight, w_n for that document.

In this example, document $\vec{d}_4$ has the largest weight, and thus becomes the group leading document for its group. This group, labeled G_1 consists of all documents which has contributed to the weights of the group leading document. In this case, $G_1 = \{\vec{d}_1, \vec{d}_2, \vec{d}_4, \vec{d}_7\}$.

To define the next group, G_2, all correlation between the members of G_1 is removed. The new, updated correlation matrix is shown in Table 2 with new weights not including correlation within G_1. It should be marked that not only correlation with the group leading document is removed, but also in between the other elements of the group.

From the new correlation matrix, the largest weight is again chosen as the group leading document for the next

group, G_2. In this case, the group leading document for G_2 is $\vec{d}_6$. The new group is then defined as $G_2 = \{\vec{d}_2, \vec{d}_5, \vec{d}_6\}$. Since no other correlations exist after removal of the correlation between the elements of G_2, all documents are categorised into two groups,

$$D = \{\vec{d}_1, \vec{d}_3, \vec{d}_4, \vec{d}_7\} \cup \{\vec{d}_2, \vec{d}_5, \vec{d}_6\} = G_1 \cup G_2, \qquad (14)$$

which completes the example.

Suggesting Web pages for the User

After finding the group leading documents of the groups created using the procedure described above, these can be used to provide the basis for new personalised information retrieval by the system. Each group leading document now represent a distinct topic that is of interest to the user.

The search for new information for the user takes place in the proxy server's Web cache. The group leader documents of each user are compared with all documents that are *not* a part of the user's profile, using the correlation scheme previously outlined. This allows Exquiro to present a page with links to those documents that are both new to the user and have the highest similarity to the most interesting topics the user has been looking at.

The problem of gaining enough data to suggest good resources while the number of users of the system are fairly low, is handled by building a separate agent into the framework. This agent is simulating a set of anonymous users with mostly the same interests as the real users, by extracting keywords from all of the group leader documents for all users, submitting these to various search engines, and retrieving a set of the documents found by the search. These are then of course filtered by the Exquiro system prior to presentation to the users to improve relevancy.

Conclusions and Areas of Further Study

In this report, a novel design of a personalised intelligent Web agent has been presented. Important qualities that a Web agent are expected to contain, are maintained. By having system users connected to the Web via a proxy server mechanism, user profiles are created, and leading documents representing the user's interest are extracted. Based on these leading documents, new and relevant Web pages are presented for the user.

One of the major problems with the current system is the computational load imposed by the quadratic nature of the algorithm. A related problem is the network imbalance caused by all users having to use the same proxy server. These problems would both point to a solution utilising a large network of servers, where users would be using "close" servers. A solution would then have to be found for sharing documents between servers in order to have the largest possible set of alternatives to suggest to the user.

Solutions are currently being developed for authenticating users by using a nickname/password pair on the proxy server. The speed problem is also being addressed by profiling the code and working to implement as efficient code as possible in the critical areas.

The system described here is named Exquiro, and is operative for usage at http://www.exquiro.net/.

Acknowledgements

We would like to thank the School of Computer Sciences, Østfold College for allowing us to pursue this idea over the last years and supporting us while we were always asking for larger disks and more computing power. We would also like to thank the various students that have been involved in trying out our ideas in more than usually challenging projects.

References

Armstrong, R., Freitag, D., Joachims, T., & Mitchell, T. (1995). A learning apprentice for the world wide web. In *Aaai spring symposium on information gathering from heterogeneous, distributed environments.*

Berners-Lee, T., Masinter, L., & McCahill, M. (1994). *Uniform Resource Locators (URL)* (RFC No. 1738). IETF Network Working Group.

Borgman, C. L. (1986). Why are online catalogs hard to use? *Journal of American Society for Information Science, 37*(6), 387–400.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th international world-wide web conference* (pp. 107–117). Brisbane, Qld.

Cheung, D. W., Kao, B., & Lee, J. (1998). Discovering user access patterns on the world wide web. *Knowledge-Based Systems, 10*, 463–470.

Cohen, W. W. (1998a). The whirl approach to integration: An overview. In *Aaai-98 workshop on ai and information integration.*

Cohen, W. W. (1998b). A web-based information system that reasons with structured collections of text. In *Second international conference on autonomous agents (agents'98).* Minneapolis/St. Paul.

Cohen, W. W., & Hirsh, H. (1998). Joins that generalize: Text classification using whirl. In *Kdd.*

Croft, W. B., & Lewis, D. D. (1987). An approach to natural language processing for document retrieval. In *10th annual international acm sigir conference on research and development in information retrieval (sigir-87)* (pp. 26–32). New Orleans, LA.

Dean, J., & Henzinger, M. R. (1999). Finding related pages in the world wide web. *Computer Networks, 31*, 1467–1479.

Fenichel, C. H. (1981). Online searching: Measures that discriminate among users with different types of experience. *Journal of American Society for Information Science, 32*, 23–32.

Haverkamp, D. S., & Gauch, S. (1998). Intelligent information agents: Review and challenges for distributed information sources. *Journal of American Society for Information Science, 49*(4), 304–311.

Joachims, T. (1997). A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *International conference on machine learning (icml).* Nashville, Tennessee, USA.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *International conference on machine learning (icml).* Bled, Slovenia.

Lewis, D. D. (1992). Text representation for intelligent text retrieval: A classification-oriented view. In P. S. Jacobs (Ed.), *Text-based intelligent systems.* Lawrence Erlbaum.

Lieberman, H. (1995). Letizia: An agent that assists web browsing. In *International joint conference on artificial intelligence.*

Marchiori, M. (1997). The quest for correct information on the web: hyper search engines. *Computer Networks and ISDN Systems*, *29*, 1225–1235.

Monson, L. (1997). Classifying text with id3 and c4.5. *Dr.Dobb's Journal*, 117–119.

Porter, M. F. (1980). An Algorithm for Suffix Stripping. *Program*, *14*(3), 130–137.

Salton, G. (Ed.). (1989). *Automatic text processing: the transformation, analysis, and retrieval of information by computer.* Addison Wesley.

Walker, N., & Truman, G. (1997). Neural networks for data mining electronic text collections. *SPIE Proceedings Series*, *3077*, 299–306.

Xu, J., & Croft, B. (1998). Corpus-based stemming using cooccurrence of word variants. *ACM Transactions on Information Systems*, *16*(1), 61–81.