A Multilevel Approach to Intelligent Information Filtering: Model, System, and Evaluation

J. MOSTAFA
Indiana University
S. MUKHOPADHYAY
Purdue University
W. LAM
The Chinese University of Hong Kong
and
M. PALAKAL
Purdue University

In information-filtering environments, uncertainties associated with changing interests of the user and the dynamic document stream must be handled efficiently. In this article, a filtering model is proposed that decomposes the overall task into subsystem functionalities and highlights the need for multiple adaptation techniques to cope with uncertainties. A filtering system, SIFTER, has been implemented based on the model, using established techniques in information retrieval and artificial intelligence. These techniques include document representation by a vector-space model, document classification by unsupervised learning, and user modeling by reinforcement learning. The system can filter information based on content and a user's specific interests. The user's interests are automatically learned with only limited user intervention in the form of optional relevance feedback for documents. We also describe experimental studies conducted with SIFTER to filter computer and information science documents collected from the Internet and commercial database services. The experimental results demonstrate that the system performs very well in filtering documents in a realistic problem setting.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*clustering*; *selection process*; I.2.6 [**Artificial Intelligence**]: Learning; I.7.3 [**Text Processing**]: Index Generation

S. Mukhopadhyay was partially supported by NSF CAREER grant ECS-9623971 during the course of the research reported in this article.
Authors' addresses: J. Mostafa, School of Library and Information Science, Indiana University, Bloomington, IN 47405-1801; email: jm@juliet.ucs.indiana.edu; S. Mukhopadhyay and M. Palakal, Computer and Information Science, Purdue University School of Science at Indianapolis, Indianapolis, IN 46202; W. Lam, Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong.

General Terms: Algorithms, Experimentation, Theory

Additional Key Words and Phrases: Automated document representation, information filtering, user modeling

1. INTRODUCTION

Information-filtering (IF) systems have recently gained popularity, mainly as part of various information services based on the Internet [Edwards et al. 1996; Oard 1996]. These systems are similar to conventional information retrieval (IR) systems in that they aid in selecting documents that satisfy users' information needs. However, certain fundamental differences do exist between IF and IR systems, making IF systems interesting and an independent object of analysis [Belkin and Croft 1992]. IR systems are usually designed to facilitate rapid retrieval of information units for relatively short-term needs of a diverse population of users. In contrast, IF systems are commonly personalized to support long-term information needs of a particular user or a group of users with similar needs. They accomplish the goal of personalization by directly or indirectly acquiring information from the user. In IF systems, these long-term information needs are represented as interest profiles (Lewis [1992a] refers to them as standing queries), which are subsequently used for matching or ranking purposes. The interest profiles are maintained beyond a single session and may be modified based on users' feedback. Another important difference has to do with the document source. IR systems usually operate on a relatively static set of documents, whereas IF systems are usually concerned with identifying relevant documents from a continuously changing document stream.

To operate efficiently, IF systems must acquire and maintain accurate knowledge regarding documents as well as users. The dynamic nature of users' interests and the document stream makes the maintenance of such knowledge quite complex. Acquiring correct user interest profiles is difficult, since users may be unsure of their interests and may not wish to invest a great deal of effort in creating such a profile. Acquiring information regarding documents is also difficult, because of the size of the document stream and the computational demands associated with parsing voluminous texts. At any time, new topics may be introduced in the document stream, or user's interests related to topics may change. Furthermore, sufficiently representative documents may not be available to facilitate a priori analysis or training. Research on filtering, so far, has not clarified to a significant extent how these particular problems associated with users and documents may influence the overall filtering process.

In this article, we present both an analytical and an empirical examination of the basic problems in filtering. In our investigation here of the demands placed on IF systems, we identify the relevant functions and express them at a suitable abstraction level. This abstraction (we refer to it as the *model*) is then implemented as a system using well-known techniques from information science and machine learning. Following this, the performance of the resulting system is subjected to rigorous experimental analysis to clarify the influence of major constituent functions on the overall filtering process. The primary objective of an IF system is to perform a mapping from a space of documents to a space of user relevance values. This mapping, in turn, can be decomposed into a multilevel process, where the intermediate functions involve the subproblems of representation, classification, and profile management. To ensure effective service, we further assume that these functions must be realized under two strict constraints. First, user intervention in the operation of the system must be minimized. That is, the system should rely on automated techniques as much as possible for acquiring information about documents and users. Second, when faced with changes in documents or users' information needs, the system must adjust quickly with little or no degradation in performance.

In the rest of this section, we discuss in more detail the challenges associated with performing effective filtering while minimizing user intervention and system degradation. We then identify some of the basic problems associated with filtering and delineate our approach for addressing them. We conclude the section by surveying related research. In Section 2, we present our model for information filtering. A description of an implementation of the model, named SIFTER (Smart Information Filtering Technology for Electronic Resources), is provided in Section 3. Results of experimental analysis conducted on SIFTER (and indirectly on the underlying model used) are presented in Section 4. In Section 5, we discuss possible future extensions of SIFTER. Finally, we present our conclusions in Section 6.

1.1 Problem Description

Uncertainties in the filtering environment—especially the dynamic nature of users' interests and the document stream—make it extremely difficult to gather and maintain accurate information necessary for filtering. Rapid or gradual changes introduced in the environment, viewed from the perspective of the filtering system, are sources of uncertainty. To manage such uncertainties requires a high level of adaptivity on the system's part. This adaptivity can be achieved by applying various machine-learning techniques. The overall problem of IF may then be broadly posed as learning a map from a space of documents to the space of real-valued user relevance factors. More precisely, denoting the space of documents as $\mathcal{D}$, the objective is to learn a map $f : \mathcal{D} \rightarrow \mathbb{R}$ such that $f(d)$ corresponds to the relevance of a document d. Given that such a map is known for all points in $\mathcal{D}$, a finite set of documents can always be rank-ordered and presented in a prioritized fashion to the user.

In an IF system, f is not known a priori and has to be estimated on-line based on queries and user feedback. This could, in principle, be accomplished by setting up some form of a parameterized map approximator (such as artificial neural networks) and updating the parameters based on the feedback. Such a direct on-line learning of the map f, however, is computationally intensive and requires a large number of user feedbacks, considering the high dimensionality of any reasonable representation of the documents. To provide a practically feasible solution to the filtering problem, we decompose the latter into two levels. The higher level represents a classification mapping f_1 from the document space to a finite number of classes $\{C_1, \ldots, C_m\}$ (i.e., $f_1 : \mathscr{D} \rightarrow \{C_1, \ldots, C_m\}$). This mapping is learned in an off-line setting, based on a representative database of documents, either by using *prior* information concerning the classes and examples or by automatically discovering abstractions using a clustering technique. Hence, this higher level partitions the document space into m equivalent classes over which user relevance is estimated. The lower level subsequently estimates the mapping f_2 describing user relevance for the different classes (i.e., $f_2 : \{C_1, \ldots, C_m\} \rightarrow \mathbb{R}$). Since f_2, unlike f and f_1, deals with a finite input set of relatively few classes, the on-line learning of f_2 is not unrealistically time consuming and burdensome on the user. Thus, the map f is being learned as the composition of f_1 and f_2. The decomposition of f into f_1 and f_2 clearly limits the maximum achievable filtering accuracy, since a class may not correspond to a constant user interest. However, in our experience, the resulting inaccuracy is more than adequately compensated for by the substantial reduction in learning complexity. If greater accuracy is desired, it can be achieved as a two-stage process. In the first stage, a two-level map (i.e., f_1 and f_2) is learned as stated before. Subsequently, a more general single-level learning scheme can be initialized on the basis of learned f_1 and f_2. From then onward, the general map can be used for ranking purposes and can be updated on the basis of user feedback.

Decomposition of f only aids in reducing the learning complexity; it does not eliminate it. The on-line learning problem is made even more difficult due to the following factors:

(1) *Difficulty of Representation*: In general, it is not possible to represent $\mathscr{D}$ exactly by a finite-dimensional space that corresponds to some features of the documents (e.g., the relative frequencies of some predefined keywords). Hence, any finite-dimensional representation space $\mathscr{D}'$ is merely an approximation to $\mathscr{D}$, and there is always a loss of information in the process. The area of document representation and indexing [Salton and McGill 1983] is devoted to discovering methods for finite-dimensional representations that minimize the information loss in some sense. In a dynamic environment, to make the problem more difficult, the most preferable representation scheme is also a function of time. The choice of the representation scheme directly affects the realization of function f_1.

(2) *Stochasticity of Feedback*: The user relevance feedback may at certain times appear to be random to the filtering system. This can occur due to several reasons. First, the particular user interacting with the system

may have uncertain needs or may not be very discriminating in expressing his or her needs. Second, depending on the f_1 chosen, the target classes may not correspond to the way a user would normally group documents. This may lead to the generation of different user relevance feedback values for documents belonging to the same class. The third and final factor relates to the difficulty described in (1). On certain occasions, user feedback may be motivated by particular features (e.g., keywords) in documents that are actually not part of the underlying representation scheme. Feedback generated based on such "missing features" would appear as random, because the system would be unable to determine what caused such feedback.

(3) *Changing Interests of the User*: Due to personal or professional reasons, a user's interests may shift or change. These changes may happen in a relatively short duration of time or over a long period. We refer to all such situations as the nonstationary user case. The shifts can affect the user's interests partially or fully. Whatever the scope of such shifts, the interest profile must be updated accordingly. The map f_2 is directly affected by this problem.

As mentioned earlier, due to the inherent complexity, filtering based on a direct learning approach is very difficult to accomplish in an efficient fashion. Decomposition allows us to isolate more specific problems, and we solve them by relying on existing and newly developed approaches. The main contributions of this article can now be summarized as follows:

—We present a general model of filtering. As a way to reduce complexity, the architecture of the model incorporates multilevel functional decomposition and supports generality through modularity. It admits application of virtually any preferred techniques for basic tasks involving representation, classification, and profile management.

—The idea of learning is made central to the filtering process. We show how learning techniques can support the high degree of adaptivity required while minimizing user intervention. We apply learning techniques for acquiring information about both documents and users. To support adaptation to changes in the document stream, an unsupervised cluster discovery method is used. A reinforcement learning algorithm with very low overhead is used for user interest profile acquisition.

—We demonstrate how representation can be conducted on a dynamic stream of text. The method provides a high degree of control in determining what content to capture and what to ignore. The classification process is also designed to be flexible. The set of classes (i.e., the target of f_1) can easily be changed by invocation of a relearning process. Both of these features allow convenient tuning of the filter to minimize user intervention.

—We describe a method to handle profile degradation due to shifts in user interests. Graceful handling of interest shifts without requiring additional data from the user is supported by the method. It is capable of detecting multiple interest shifts in the same user and can take appropriate actions to minimize possible negative effects on the function f_2.

Preliminary simulation experiments involving an implementation of the general model have been reported in earlier sources [Lam et al. 1996; Mukhopadhyay et al. 1996]. These experiments simulated the operation of filtering LISTSERV emails. In this article, we describe the integration of all functionalities into a complete working system, conduct studies involving human users in a real-world filtering application, and systematically analyze the influence of various user- and system-related parameters on the filtering performance.

1.2 Related Work

As mentioned in the introduction, IF systems are strongly related to IR systems in their functional goals and in the methods they apply to accomplish those goals. Belkin and Croft [1992] provided an excellent review of IF, comparing it with IR and several other closely related processes (e.g., text routing, text categorization, etc.). We do not intend to repeat such a comparison here. Instead, we review literature that deals more directly with the problems delineated in the last section.

A basic filtering problem is to transform a large volume of information (text) into entities that permit efficient computation without significant loss of content. In our formulation of the problem, this is the objective of function f_1, mapping documents to a more limited space of document classes. This particular task is generally referred to in IR as automated document classification. The first step in this process demands that a representative feature set for each document be identified. Various techniques have been developed for feature selection, ranging from simple procedures that calculate statistical distribution of keywords to more sophisticated techniques relying on analysis based on natural language processing (NLP) algorithms. Lewis [1992a] provided a thorough review of feature selection procedures and the influence of such procedures on document classification. The general and surprising finding of feature selection research in IR is that simple keyword-distribution-based approaches are almost as effective as more sophisticated approaches [Lewis 1992b]. The next step in classification involves assigning documents to one or more groups. In IR, hierarchical cluster generation techniques such as single-link and complete-link methods are commonly applied [Salton 1989]. A particular track of research in IR has concentrated on generating predictive classifier functions based on off-line training conducted on representative document sets. As far back as 1963, Borko and Bernick [1963] described a text classifier based on a simple linear regression model that produced good results. A more recent successful effort that also applied a linear model classifier, based on least-squares fit, was described by Yang and Chute [1994]. The DARPA-sponsored MUC (Message Understanding Conferences) initiative has generated significant research in the area of text routing [Lewis and Tong 1992]. The MUC efforts rely on strong

NLP approaches to develop classifiers, since analysis is necessary at a fine-grain level to assess document content (e.g., identification of terrorist events based on news stories). It has been demonstrated, however, that less complex linear models may be appropriate if the type of information a system must handle is relatively simple (e.g., elements that constitute bibliographic document information). Generally, in most IF systems, the classification process has to be conducted fast, whereas the classifier building process can be delegated to a slower process.

Traditionally, IR placed little attention on the users' role—specifically, identification of users' interests, representation of interests, and application of such representations in interactions [Belkin and Croft 1992]. Myaeng and Korfhage's [1990] work on user profiles is one of the few and important efforts in this area. It attempted to integrate user interest profiles in IR systems and focused on various combinations of queries and profiles in enhancing retrieval. The profiles however had certain limitations. They had to be contributed directly by the user (who may be uncertain or unwilling to take the trouble), and profiles did not change during interaction. To keep up with changes in users' interests automatically, systems can rely on internal knowledge representations or on learning. Rich [1983] demonstrated how in an IR setting, in the absence of direct evidence about information needs, stereotypes can be applied to generate user models representing long-term interests. This is an innovative technique; however, substantive human investment in knowledge engineering would be required to build the user stereotypes. Relevance feedback, a highly constrained and indirect form of evidence, has been successfully used to learn and adapt representations used for the purpose of query reformulation [Frants et al. 1993; Goker and McCluskey 1991]. It should be noted, though, that in many IF systems queries (in an IR sense) are not necessary, and users' interests are more stable than in typical IR situations. These factors must be taken into consideration in devising methods that minimize user involvement in profile management.

A body of IF research exists that directly addresses problems associated with profile acquisition and maintenance, applying mostly AI-based techniques. Malone et al. [1987] described an intelligent message-sharing system called InfoLens in which users can generate profiles using rules. The rules prescribe appropriate actions with tests on content-based factors such as message type, date, and sender. Such explicit user-based knowledge acquisition methods support a high degree of transparency, permitting users to follow an "up-to-the-moment" knowledge state of the system. InfoScope, a system that applies a similar technique, has been developed for filtering Usenet news [Fischer and Stevens 1991]. InfoScope uses heuristic rules associating common patterns of usage (e.g., number of sessions, newsgroups read, frequencies of relevant terms in an article, etc.) to appropriate actions. To refine profiles, users must add or remove terms from the profile and must set appropriate rule-triggering thresholds. The requirement for direct and explicit user input for profile management, in our view, is somewhat demanding, and furthermore, such rule-based approaches may be too "brittle" to support efficient profile adaptation. NewsWeeder [Lang 1995] is another Usenet filtering tool. In this, users' ratings of documents are used as training examples for a machine-learning algorithm that is executed nightly to generate the user interest profiles for the next day. By limiting the user input to only ratings of documents, NewsWeeder is successful in reducing user involvement. However, NewsWeeder's inability to adapt the profile in an on-line fashion limits its utility. SIFT (Stanford Information Filtering Tool) has also been developed to filter Usenet news [Yan and Garcia-Molina 1995]. SIFT requires users to specify keywords to generate the initial profile. Depending on the user's choice, the filter may be represented using the vector-space model or simply as a boolean formula. If a vector-space approach is selected, SIFT can provide some adaptivity in profile refinement. In this mode, SIFT requires users to provide relevance feedback (by pointing out documents of interest), based on which weights in the profile are adjusted accordingly. Finally, NewT (news tailor) [Seth 1994] offers the user the option to select multiple profiles from a set of predefined profiles that cover common topical areas. NewT also applies relevance feedback for profile adaptation. To further reduce user involvement in profile refinement, NewT utilizes a genetic algorithm to evolve profiles toward increased fitness.

In summary, IR provides a solid basis to exploit various document representation techniques, especially for the intermediate IF stage of document classification (i.e., f_1). Relevance feedback and machine-learning-based approaches show promise in handling the subsequent IF operation of user modeling. However, at this point little is known as to how multiple functional components can be integrated satisfactorily in a single IF system, and additional empirical evidence is required to clarify how characteristics associated with users and the document stream may affect filtering performance.

2. FILTERING MODEL

There are three important and independent entities that constitute a filtering environment. These are the document source, the filter, and the user (Figure 1). Documents may exist at various sites and may be received by the user through disparate channels. The task of storing such documents, before filtering, is handled by a component we call *document acquisition and management (DAM).* DAM is a separate component from the filter, and its actual design may vary from one environment to another. For example, at its core, DAM may be a web-crawler utility that retrieves documents from designated sites, a daemon that maintains indexed files, or even a sophisticated DBMS. Whatever the construction of DAM, when invoked, it would produce a stream of documents that flows into the filter.

The filter itself consists primarily of three modules: ($M1$) representer, ($M2$) classifier, and ($M3$) profile manager. In the context of the multilevel decomposition of the map $f : \mathcal{D} \rightarrow \mathbb{R}$ (i.e., $f = f_2 \circ f_1$) discussed in Section 1.1, $M1$ determines the input space for f_1; $M2$ maps the resulting vector

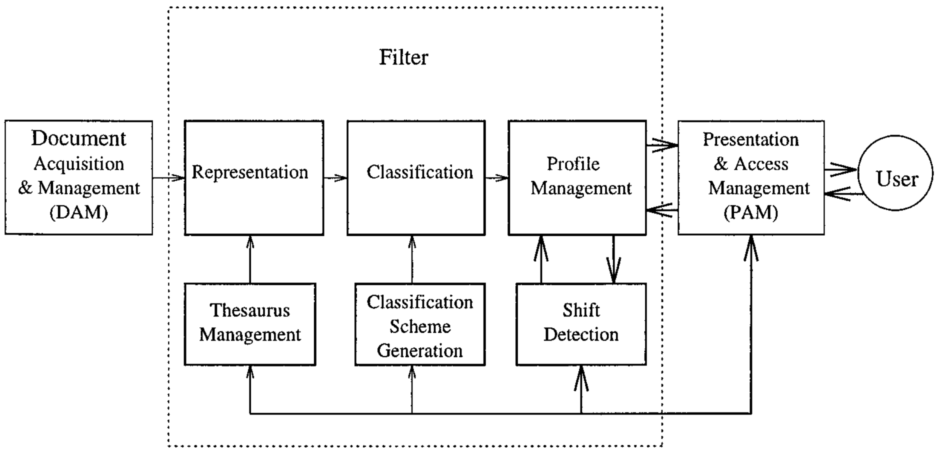

Filter
Document Acquisition & Management (DAM)
Representation
Classification
Profile Management
Presentation & Access Management (PAM)
User
Thesaurus Management
Classification Scheme Generation
Shift Detection

Fig. 1. Model of the filtering process.

representation to the classification space (i.e., the output for f_1); and $M3$ implements the mapping f_2. The functions of these modules are best described in terms of two different modes: filter application and filter tuning. In the filter application mode, upon arrival of new documents the representer module transforms the stream into more efficient representations. This transformation would involve identifying relevant concepts and correctly assessing the discriminatory value of concepts in relation to specific documents. To avoid unnecessary parsing of concepts, a thesaurus management submodule would be used to select concepts for only those domains that are of interest to the user. The function of the classifier module is to identify for each document its corresponding document class or group. The classifier module utilizes a classification scheme, generated by a submodule, as an off-line process. In selecting an appropriate size for the space of classes, a crucial constraint must be followed. The space of classes in the filter must be smaller than the space of input document space. This aspect of the filtering model ensures a significant reduction in computational complexity (see the discussion in Section 1.1). The profile manager module has the dual role of maintaining accurate interest profiles and applying the profiles to assess the relevance of documents. Profile representation constitutes information concerning user preferences for document classes utilized by the filter. Such preference information may be acquired in various ways, but the method requiring the least user effort should be favored (i.e., it should be the default method used by the system). It appears that the best automatic profile acquisition methods are available from the machine-learning literature, relying on relevance feedback from the user. Whatever method is ultimately chosen, users should always have the option to enter or modify values in their profiles directly to ensure transparency of the filter. Once profile representation is achieved, documents are ranked in relation to their membership in classes. It is worth noting here that, due to the strict imposition of a class space, assignment of semantically related documents to different classes may occur. But, as profile learning is always conducted over the set of classes, it would have minimal effect on the overall document ranking. After the profile is learned, the classes that are semantically related are treated approximately equally by the system, for ranking purposes.

At the output end of the filtering model lies the *presentation and access management (PAM) system*. PAM is more tightly coupled with the filter than DAM and would normally be the user interface of the filtering system. To support the filter application mode, PAM can offer various functions, the most important being the actual presentation of documents. PAM must allow the user to select documents for display and to control the way documents are actually displayed (e.g., window size, font size, color, etc.). Another important function of PAM is to collect information for the purpose of profile management. For example, if relevance feedback is chosen, functions should exist to permit users to point out relevant documents.

In modeling the filter, we also identified ways of tuning the filter so as to customize and improve its performance. Various types of tuning operations can be performed to influence the behavior of the three modules that constitute the filter. The frequency of such tuning would vary depending on the proximity of the particular module to the user (i.e., from the PAM end of the model). Hence, the profile manager is subjected to frequent tuning. An important type of tuning that applies to the profile manager module is avoidance of profile degradation when a user's interests change due to some external circumstances. Because such a case can have an immediate effect on the filter's performance, it should preferably be handled automatically. This would require continuous monitoring of users' feedback and predicting shifts as quickly as possible. We show this tuning operation as a submodule of the profile manager module. The structure, size, and content of the classification scheme can also have a significant influence on the filter's behavior. Such a scheme is usually generated using a training document set (a large and representative document set). However, the content of the document stream may change sufficiently over time to demand regeneration of the classification scheme. This type of tuning would be necessary less frequently and can be conducted by a submodule of the classifier (using the last n documents as the new training set). Finally, the structure and content of the thesaurus may directly affect document representation and consequently the rest of the filtering processes. When a domain or a field experiences significant change (which usually happens very slowly), tuning operations would be needed to update the thesaurus to keep up with such changes. We show these operations as a submodule of the representer module.

3. SIFTER: AN IMPLEMENTATION OF THE FILTERING MODEL

As a way to empirically investigate the utility of the model, we implemented a filtering system named SIFTER (written in C and TCL/TK for a Unix environment) that incorporates the major components described in the last section. We now describe these components in detail. We begin with the filter part of SIFTER, focusing mainly on the three constituent modules.

3.1 Document Representation Using a Vector-Space Model

The first component of the filter (i.e., the document representation module) needs to convert documents into structures that can efficiently be parsed without the loss of vital content. We chose the vector-space model [Salton 1989] for document representation, because it has been widely tested and is general enough to support other computational requirements of the filtering environment. This, in turn, relies on a thesaurus management submodule. At the core of the latter is a set of technical terms or concepts culled from authoritative sources representing a given area (presently, we are using the ACM *Computing Reviews* Classification Scheme and the American Society of Information Science Thesaurus for filtering documents). The thesaurus management submodule is also used for the pruning of common functional terms, as well as several term normalization tasks, including synonymy control and singular-plural form control.

We apply the popular tf*idf (term frequency times inverse document frequency) technique to establish the particular degree of importance for each concept in a document. To apply this technique, a table is generated off-line containing total frequencies of all terms in the thesaurus, using a sufficiently representative collection of documents as a base (2000 documents randomly sampled from the source). A separate and representative document base is especially useful in filtering because the incoming document stream may occasionally contain only a few documents. In the on-line filter application mode, another table is generated containing the frequencies of all unique terms found in newly arrived documents. Then, based on the values in the two tables, the following equation is used to derive appropriate weights for terms in each document:

$$W_{ik} = T_{ik} \times \log(N/n_k),$$

where T_{ik} is the number of occurrences of term T_k in document i; $I_k = \log(N/n_k)$ is the inverse document frequency of term T_k in the document base; N is the total number of documents in the document base; and n_k is the number of documents in the base that contain the given term T_k.

3.2 Document Classification Module

The classification module consists mainly of two processing stages: an unsupervised cluster learning stage and a vector classification stage. During the learning stage, an initial cluster hypothesis $[C^1, \ldots, C^k]$ is generated from a representative sample of document vectors $[S^1, \ldots, S^N]$. Each cluster C^i is then represented by its centroid, Z^i. During the classification stage, an incoming document vector V^i is classified into a particular class C^k using the learned centroids from stage 1. The learning of cluster centroids is done in an off-line batch mode while classification is carried out continuously as documents arrive.

A simple, heuristic, unsupervised clustering algorithm, called the *Maximin-Distance* algorithm [Tou and Gonzalez 1974], is currently used to determine the centroids over the document vector space. In this algorithm, centroids are generated in an iterative manner. At each stage, a point (document) in the data set is selected that has the largest distance from the existing centroids. The distance of any document from the existing centroids, in turn, is the minimum of its distances over all the centroids. The selected point is then added as a new centroid only if its distance from the existing centroids is an appreciable fraction of the previous maximum distances. The threshold value used for this fraction determines the granularity and the number of the clusters. During the on-line operation, the classification module merely classifies an incoming document vector V^i as belonging to the class whose centroid has the minimum distance to the document vector. The resulting class information corresponding to each vector is then passed on to the user profile learning module.

The measure used for computing the distance between two document vectors is the cosine similarity measure [Salton 1989]. Given two nonnull document vectors $X = [x_1, \ldots, x_t]^T$ and $Y = [y_1, \ldots, y_t]^T$, such a similarity measure represents the cosine of the angle between them and is calculated as

$$\frac{\sum_{i=1}^{t} x_i y_i}{\sqrt{(\sum_{i=1}^{t} x_i^2)(\sum_{i=1}^{t} y_i^2)}}.$$

The distance is then computed as 1 minus the similarity. When one or both of the vectors are identically zero (meaning that the corresponding documents cannot be represented using the given thesaurus), no distance computations are necessary, since all such vectors are assigned to a special class called "others" by convention.

3.3 User Profile Learning Module

The function of the user profile learning module is to determine the user's preference for the different classes of information C^i $(i = 1, \ldots, k)$ and based on their classes as well as the estimated user preferences for the classes to prioritize the incoming documents. To accomplish this task, the learning agent maintains and updates a simplified model of the user, based on the relevance feedback. The algorithm currently used to learn the user model is based on a reinforcement learning algorithm studied in the artificial intelligence and mathematical psychology communities [Narendra and Thathachar 1989].

Let d_i denote the underlying (unknown) expected user preference (relevance) for the class C^i. The learning agent maintains and updates two vectors of dimensions equal to the number of classes. The first is the *estimated relevance probability vector*, with elements $\hat{d}_i$ $(i = 1, \ldots, n)$, which is an estimate of d_i. The second is an *action probability vector* $p = [p_i]$, such that p_i represents the probability of the class C^i being selected by the filter as the most relevant class. Both p and $\hat{d}$ vectors are continuously updated during the learning process, on the basis of user relevance feedback.

The learning agent at every iteration (i.e., presentation of documents to the user) sorts the incoming documents by first sampling the p vector to select the class to be presented at the top. The rest of the classes are sorted according to the corresponding $\hat{d}$ values. The elements of the p vector are all initialized to a value of $1/k$, where k is the number of classes. Hence, at the beginning, all classes are probabilistically given the chance to be ranked at the top and, hence, to receive the user's attention and relevance feedback. This allows the user model, in the form of the $\hat{d}$ vector, to be learned sufficiently accurately. As the learning progresses, one element of the p vector (corresponding to the most relevant class) approaches the value of unity, while the rest of the elements tend to zero. At the same time, the elements of the $\hat{d}$ vector approach those of d, the underlying user relevance vector. Hence, after a sufficiently long learning time, the ranking of the documents is strictly performed according to user relevances for the corresponding classes.

The learning algorithm (i.e., the algorithm for updating $p(k)$ and $\hat{d}(k)$) is described briefly as follows [Thathachar and Sastry 1985]. $\hat{d}_i(k)$ $(i = 1, \ldots, n)$ at any instant is the running average of the relevance values given by the user for documents belonging to class i. Denoting the current maximum element of the $\hat{d}$ vector as having the index l, a unit vector $E(k)$ is created of dimension n whose lth element is one and whose all other elements are zero. Then $p_i(k)$ $(i = 1, \ldots, n)$ is updated as

$$p_i(k + 1) = p_i(k) + \eta(E_i(k) - p_i(k)),$$

where $0 < \eta < 1$ is a suitably chosen step size. Thus, the p vector is moved by a small distance toward the optimal unit vector. The convergence properties of this algorithm have been well investigated [Thathachar and Sastry 1985], and it is known that, with a sufficiently small step size η, both the p vector and the $\hat{d}$ vector show the desired convergence behavior mentioned earlier.

Two points are worth repeating. First, the $\hat{d}$ vector after convergence represents a simplified model of the user's interests. In practice, the user is always given an option to specify his or her interests for each of the classes. In such a case, the elements of the $\hat{d}$ vector are initialized with the user-provided values, and the learning process can be viewed merely as augmenting the profile provided by the user. Second, profile learning, conducted over the set of classes C, can minimize the effect of assigning semantically related documents to different classes. Assume that documents of interest to the user are assigned to two different classes. Over a period of time (after the profile is learned), classes that are semantically related would be treated approximately equally for ranking purposes.

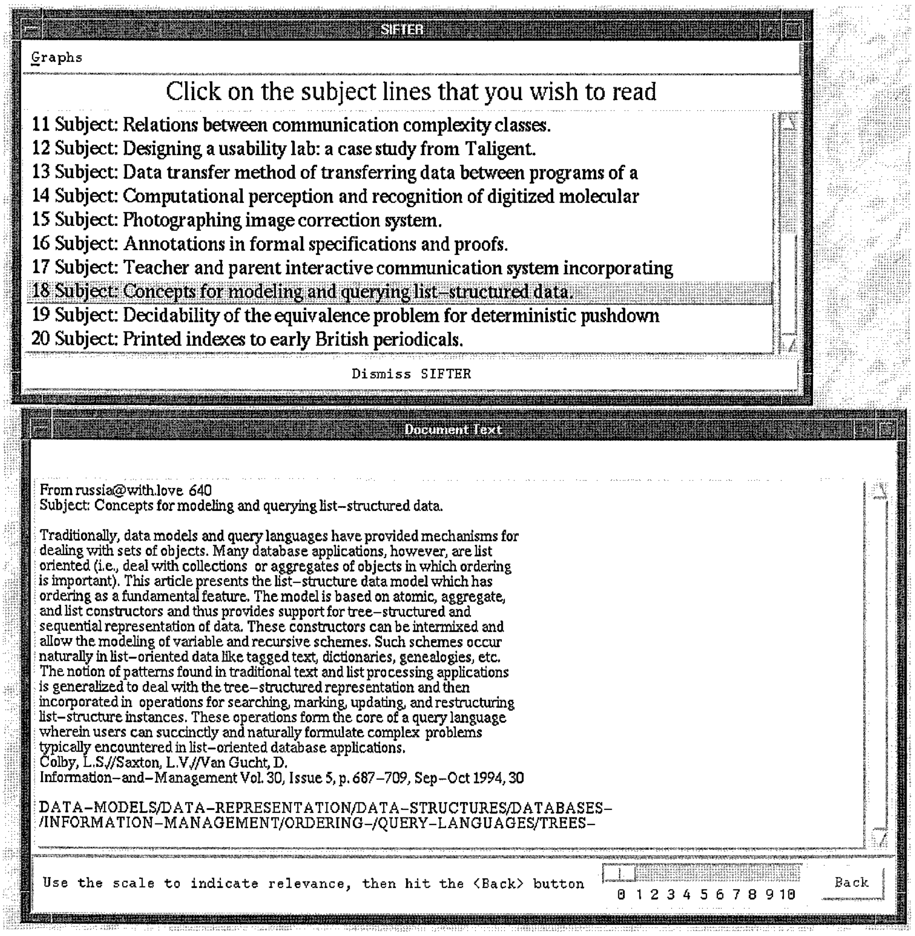
Therefore, the user is going to see the related documents together, at the top. Here, from the perspective of the user, it really does not matter that documents come from two different classes.

3.4 User Interest Shift Detection Module

Although in theory the learning algorithm described in Section 3.3 has the ability to come out of the converged state and relearn in the presence of user interest shifts, in practice, such relearning requires a very long time. During this intermediate period, the filtering performance is poor, since the ranking of the documents is carried out by means of an invalid user model. To cope with such nonstationary users, we propose using a shift detection module that tracks any changes in users' interests and suitably reinitializes the user profile learning module. The tracking is performed on each class separately, on the basis of a history of relevance feedback data, using a Bayesian framework. The specific algorithm used to perform this analysis is based heavily on Zacks and Barzily [1981]. Intuitively, the algorithm analyzes a window of feedback collected for each class in order to decide whether the user's interest in the given class has genuinely changed or whether the variation (if any) in the feedback is due only to inherent randomness. In this section, we provide a brief outline of the method. For further details concerning both the shift detection algorithm and reinitialization of the profile-learning module in response to a detected shift, the reader is referred to Lam et al. [1996].

Let β_1, β_2, . . . , β_n be the sequence of the relevance feedback data collected for a particular class. Since each β_i is either 1 or 0, governed by the underlying relevance probability of the corresponding class, β_i can be regarded as an independent Bernoulli random variable, where the relevance probability is the probability of "success" of the corresponding relevance feedback β_i. Now, the problem can be viewed as detecting a shift in the success probability in a sequence of Bernoulli trials. This, in turn, is achieved by computing the posterior probability that a shift has occurred given the relevance feedback data, as discussed in Zacks and Barzily [1981].

Given B_n (i.e., the history of the relevance feedback over the last n presentations of documents) for a given class, we denote by $h^{(u)}(B_n)$ and $h^{(d)}(B_n)$ the posterior probability of an occurrence of an upward shift (i.e., an increase in the user's interest in the given class) and a downward shift (i.e., a decrease in the user's interest), respectively, in the corresponding class of information. The actual computation of these quantities is considerably simplified by making several assumptions (e.g., a geometric prior distribution) that permit the use of certain mathematical identities. These computed posterior probabilities are used in a decision function at every iteration to determine whether a shift has occurred. Once a shift is detected in this fashion, the tracking system informs the profile-learning module, and an appropriate reinitialization of the latter's states takes place [Lam et al. 1996].



Click on the subject lines that you wish to read

11 Subject: Relations between communication complexity classes.
12 Subject: Designing a usability lab: a case study from Taligent.
13 Subject: Data transfer method of transferring data between programs of a
14 Subject: Computational perception and recognition of digitized molecular
15 Subject: Photographing image correction system.
16 Subject: Annotations in formal specifications and proofs.
17 Subject: Teacher and parent interactive communication system incorporating
18 Subject: Concepts for modeling and querying list–structured data.
19 Subject: Decidability of the equivalence problem for deterministic pushdown
20 Subject: Printed indexes to early British periodicals.

From russia@with.love 640
Subject: Concepts for modeling and querying list–structured data.

Traditionally, data models and query languages have provided mechanisms for dealing with sets of objects. Many database applications, however, are list oriented (i.e., deal with collections or aggregates of objects in which ordering is important). This article presents the list–structure data model which has ordering as a fundamental feature. The model is based on atomic, aggregate, and list constructors and thus provides support for tree–structured and sequential representation of data. These constructors can be intermixed and allow the modeling of variable and recursive schemes. Such schemes occur naturally in list–oriented data like tagged text, dictionaries, genealogies, etc. The notion of patterns found in traditional text and list processing applications is generalized to deal with the tree–structured representation and then incorporated in operations for searching, marking, updating, and restructuring list–structure instances. These operations form the core of a query language wherein users can succinctly and naturally formulate complex problems typically encountered in list–oriented database applications.
Colby, L.S.//Saxton, L.V.//Van Gucht, D.
Information–and–Management Vol. 30, Issue 5, p. 687–709, Sep–Oct 1994, 30

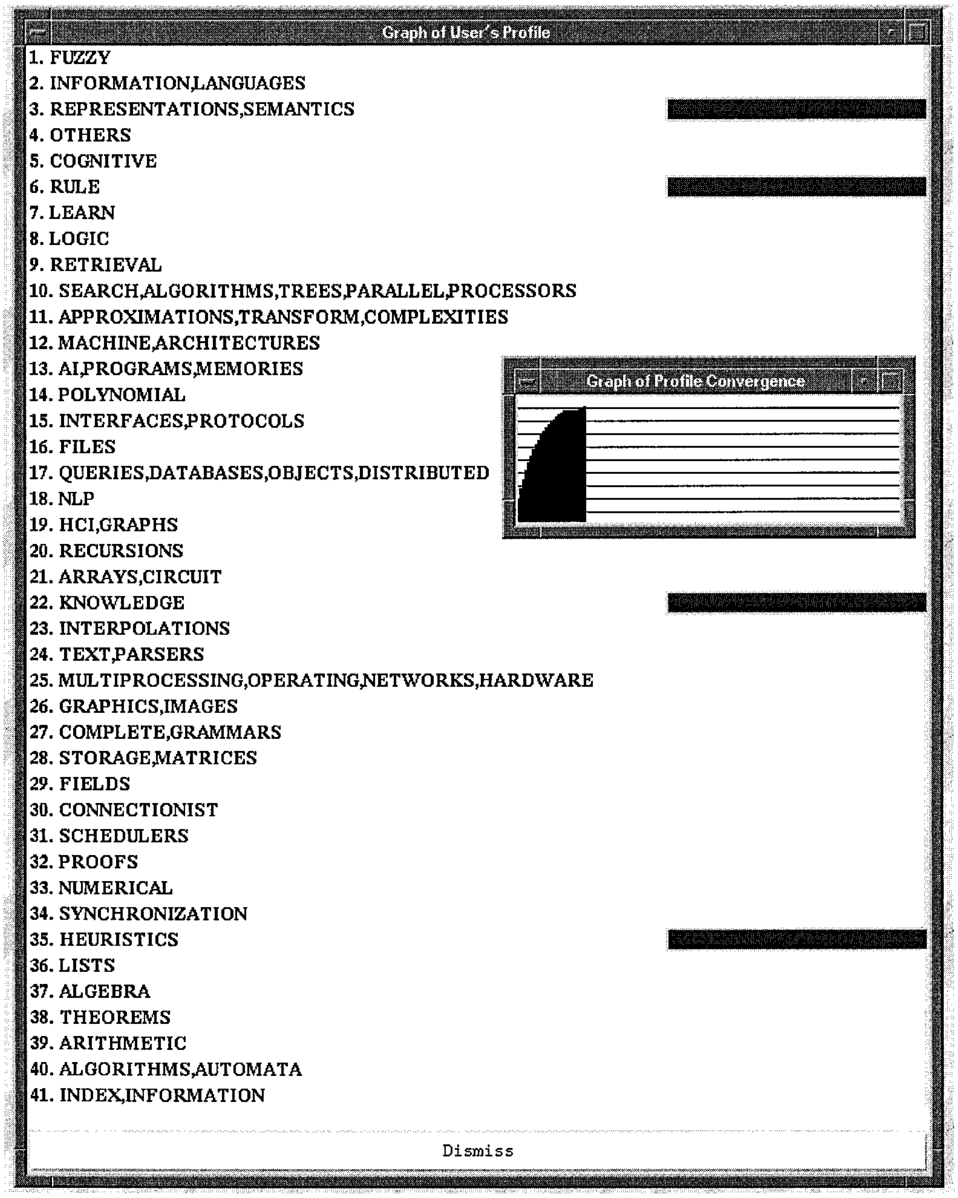
DATA–MODELS/DATA–REPRESENTATION/DATA–STRUCTURES/DATABASES–
/INFORMATION–MANAGEMENT/ORDERING–/QUERY–LANGUAGES/TREES–

Use the scale to indicate relevance, then hit the <Back> button

Fig. 2. Two main windows of SIFTER's GUI. Top window shows a ranked list of documents, and bottom window shows the body of a document with a scale for relevance feedback.

3.5 Graphical User Interface

Users interact with SIFTER by using a graphical user interface (GUI). The GUI is equivalent to the PAM component of our filtering model and performs three major functions: (1) it displays all new documents in a ranked order, (2) it allows users to select individual documents and read them, and (3) it collects user feedback and passes the feedback for user profile learning. When SIFTER is invoked, a ranked list of recently arrived document titles is displayed in a scrollable window (Figure 2). By clicking on a document title, the user can view the content of the corresponding document in a separate window. Providing relevance feedback is an optional feature. When the window with the body of a document is initially displayed, the user sees a "feedback button" at the bottom. If this button is activated, then the window is extended, displaying a scale for entering



1. FUZZY
2. INFORMATION,LANGUAGES
3. REPRESENTATIONS,SEMANTICS
4. OTHERS
5. COGNITIVE
6. RULE
7. LEARN
8. LOGIC
9. RETRIEVAL
10. SEARCH,ALGORITHMS,TREES,PARALLEL,PROCESSORS
11. APPROXIMATIONS,TRANSFORM,COMPLEXITIES
12. MACHINE,ARCHITECTURES
13. AI,PROGRAMS,MEMORIES
14. POLYNOMIAL
15. INTERFACES,PROTOCOLS
16. FILES
17. QUERIES,DATABASES,OBJECTS,DISTRIBUTED
18. NLP
19. HCI,GRAPHS
20. RECURSIONS
21. ARRAYS,CIRCUIT
22. KNOWLEDGE
23. INTERPOLATIONS
24. TEXT,PARSERS
25. MULTIPROCESSING,OPERATING,NETWORKS,HARDWARE
26. GRAPHICS,IMAGES
27. COMPLETE,GRAMMARS
28. STORAGE,MATRICES
29. FIELDS
30. CONNECTIONIST
31. SCHEDULERS
32. PROOFS
33. NUMERICAL
34. SYNCHRONIZATION
35. HEURISTICS
36. LISTS
37. ALGEBRA
38. THEOREMS
39. ARITHMETIC
40. ALGORITHMS,AUTOMATA
41. INDEX,INFORMATION

Fig. 3. Two SIFTER GUI windows showing the learning state of the system.

feedback. Otherwise, the user can close the window, without providing any feedback.

SIFTER uses the feedback values to determine the degree of interest in a subset of classes. SIFTER's GUI provides an optional display of the class space (Figure 3). In this window, SIFTER's assessment of the degree of interest for each class is plotted as a bar graph. Another optional window shows the graph of profile convergence. Although the convergence graph is a simple one, it is capable of communicating several key facts to the user. For example, when the highest value in the graph is reached (topmost horizontal line), this would indicate that the profile has been fully captured. On the other hand, when the low value is shown, this would indicate that the profile is incomplete and that more learning is to be expected. Additionally, oscillation patterns in the graph would indicate that more consistent feedback is necessary. We built these two displays to allow the user to monitor the up-to-the-moment "learning state" of SIFTER. Both of these windows can be displayed at any time, using the "Graphs" option from the main SIFTER window.

The present version of the GUI is designed to function in the X Window environment. It was created using TCL/TK and a set of C routines.

4. EXPERIMENTAL RESULTS

The filtering performance of SIFTER was analyzed with two specific goals: (1) to establish if satisfactory filtering performance can be achieved from a system (SIFTER) based on our model and (2) to clarify the influence of user-oriented and system-oriented variables on filtering performance. For the document source, we used 6000 records created for journal articles from the Information Science and Abstracts (ISA) database and computer science technical reports collected from various sites on the Internet. Each document record included a complete abstract, title, author, and subject keywords. About one-third of the documents were used for training purposes, and the other two-thirds were used for testing. A document acquisition module (DAM) was developed that, upon invocation, can generate a stream containing a preset number of new documents. The output of the DAM was used to test SIFTER.

For each experiment, the filtering performance was evaluated using two related criteria, normalized recall and normalized precision, described by the following equations [Salton and McGill 1983]:

$$Recall_{norm} = 1 - \frac{\Sigma_{i=1}^{REL} RANK_i - \Sigma_{i=1}^{REL} i}{REL(N - REL)}, \tag{1}$$

$$Precision_{norm} = 1 - \frac{\Sigma_{i=1}^{REL} \log RANK_i - \Sigma_{i=1}^{REL} \log i}{\log(N!/(N - REL)!REL!)}. \tag{2}$$

In these equations, N represents the total documents in the collection, and REL represents the total number of relevant documents. Normalized precision and recall were selected as the performance criteria because they are independent of a retrieval threshold used to segregate documents into retrieved and nonretrieved sets [Salton and McGill 1983]. In SIFTER, all documents that arrive are presented to the user in a ranked fashion. Hence, the separation of documents into retrieved and nonretrieved sets is artificial.

For assessment purposes (i.e., to compute normalized precision and recall), several additional features were built into SIFTER. At the end of each session, users were prompted to select the documents they found to be relevant. A logging facility was also included in SIFTER to save all input data from a session, including information on documents that the user found to be relevant. The log file helped us to avoid repeating sessions with users for assessing performance under different conditions (e.g., learning and no learning). Also, SIFTER was designed to support a simulation mode, in which many different parameters (both user- and system-oriented) can be conveniently set and their effects on filtering can be quickly analyzed.

4.1 Baseline Experiments

To establish realistic performance levels to be expected from SIFTER, a set of experiments was conducted with six actual users. In these experiments (referred to as baseline experiments), the objective was to determine how normalized precision and recall are affected when different users utilize SIFTER with and without learning. It is natural to expect the performance to improve with learning. However, the extent of the improvement is also important in justifying the additional computational overhead due to the incorporation of learning. Three members of our research group and three Ph.D. graduate students in computer and information sciences participated in this study. Each user received a brief tutorial (five minutes) on major SIFTER features. Users were then requested to run SIFTER for 40 sessions. For each session, SIFTER presented users with 20 new documents.

In Figures 4–7, we present detailed results from a single session with user 1 (a Ph.D. student). These results are used to explain the important features in the resulting plots and how these are used for analysis purposes. One of the main features in Figures 4 and 5 is that normalized precision and recall are highly oscillatory. The primary reasons for such oscillation are the initial latency of learning the user profile as well as variability of the document stream from session to session. For SIFTER to learn the relevance of a particular class, a number of relevance feedback iterations are required on documents belonging to that class. However, in the early period of use, the likelihood of encountering documents from previously unseen classes is very high. This produces the initial fluctuations from session to session. Another factor has to do with the variability of the document stream in terms of the number of relevant documents. If there are no relevant documents in the stream, a case that is not uncommon in filtering, both normalized precision and recall (as given by Eqs. (1) and (2)) are undefined. Hence, these equations were slightly modified so that the highest defined value would be 0.95, and the lowest would be 0. The complete absence of relevant documents (a case for which filtering is not meaningful) was made to generate a value of 1.00. Hence, the tallest spikes in Figures 4 and 5 actually represent cases where there were no relevant documents in the stream.

Even in the raw oscillatory data (Figures 4 and 5), the significant general improvement of performance with learning is apparent. However, to obtain

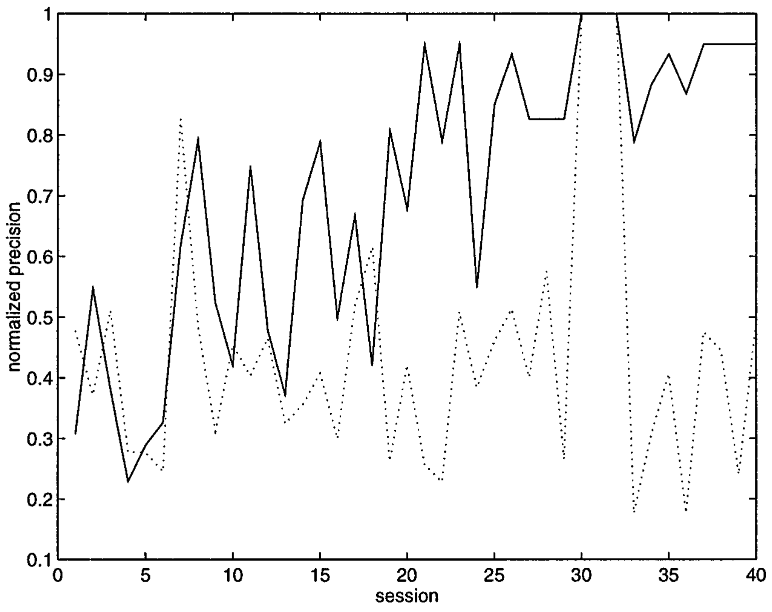


Fig. 4. Raw data showing normalized precision for user 1. Solid line: with learning; dotted line: without learning.

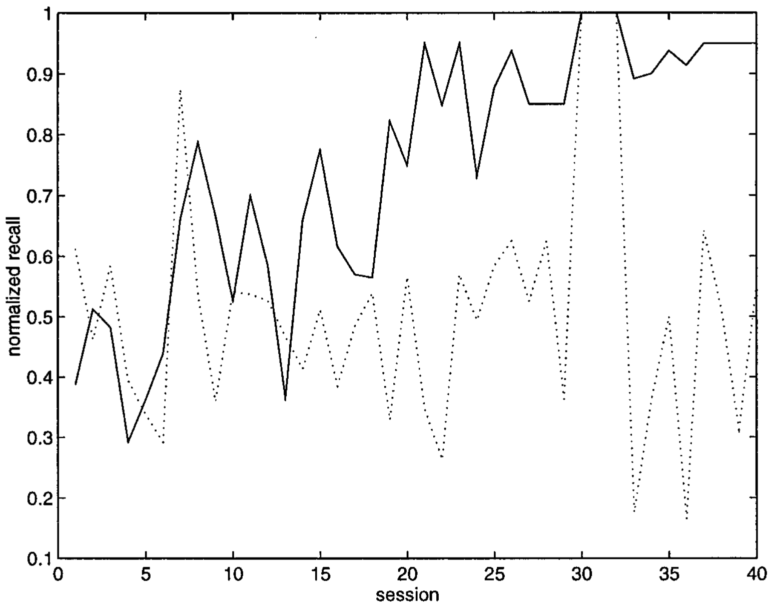


Fig. 5. Raw data showing normalized recall for user 1. Solid line: with learning; dotted line: without learning.

a clearer picture the raw precision and recall data were smoothed by averaging over every five consecutive sessions, after the removal of data for sessions in which no relevant documents were present. Figures 6 and 7

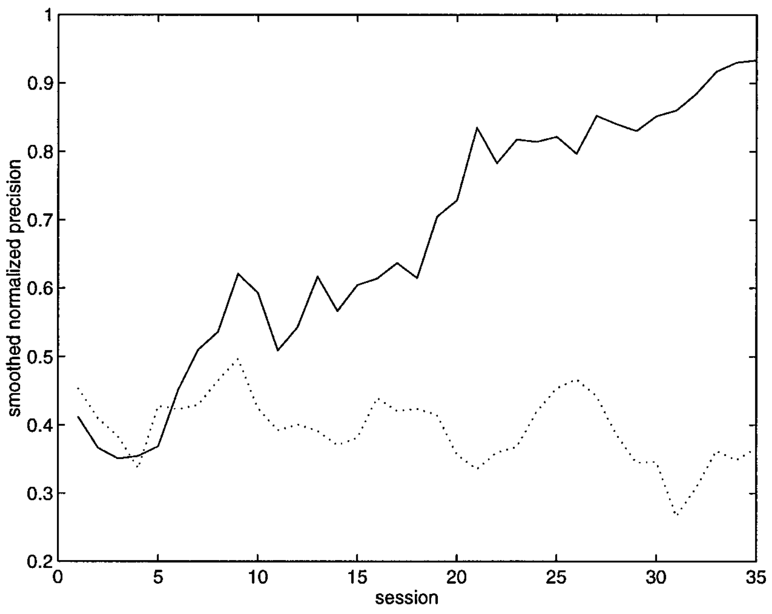


Fig. 6. Smoothed data showing normalized precision for user 1. Solid line: with learning; dotted line: without learning.

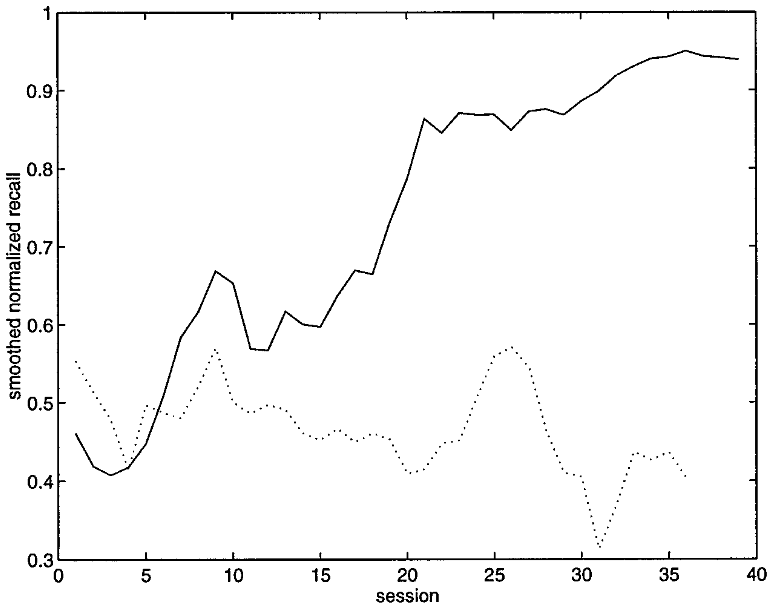


Fig. 7. Smoothed data showing normalized recall for user 1. Solid line: with learning; dotted line: without learning.

show such smoothed precision and recall performances for user 1. The general improving trend and the high terminal performance are evident from these figures. To reduce redundancy and improve clarity, henceforth

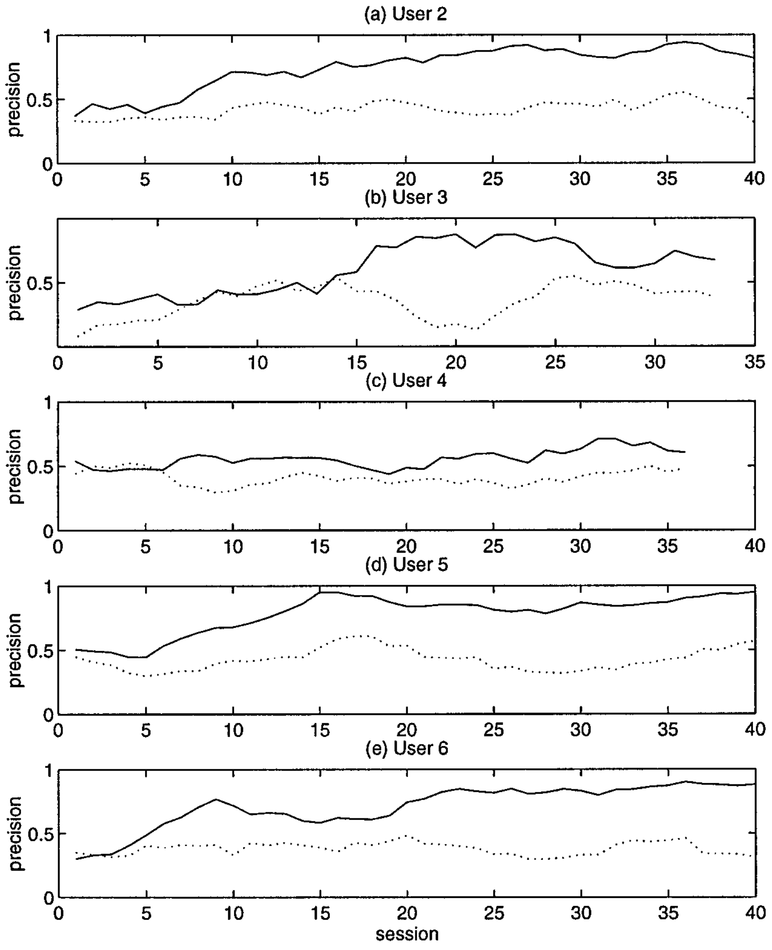


Fig. 8. Smoothed data showing normalized precision for users 2–6. Solid line: with learning; dotted line: without learning.

only smoothed results will be presented. Furthermore, since normalized recall and precision plots follow similar trends (see Figures 6 and 7), only the latter will be presented as a performance measure for filtering.

Results from the experiments involving the five other users are presented in a compact form in Figure 8. The major outcome to note is that in all cases improved performance was achieved when the system learned a user profile and applied it to filter the incoming data. In particular, the system was found to provide a very high degree of filtering performance in four out of six cases, reaching as high a value as 0.9 or better. In almost all cases, about 15–20 sessions were required before performance stabilized to a steady increase. This latency can be attributed to the fact that a sufficient number of feedback iterations are necessary before the system can fully acquire the user profile.

The variability of the filtering performance achieved with different users deserves some comment. A closer look at the log files revealed that the degree of interest in the different classes varied from user to user. Certain users were found to have very definite interests, as reflected by the relevance factors 0.9–1.0 for the relevant classes and 0.0–0.1 for the remaining classes. We refer to such users as discriminatory users. For other users, the interest levels hovered around 0.4–0.6 for many classes, meaning that they were uncertain and somewhat nondiscriminatory in their preferences. It is intuitively clear that the effect of filtering is expected to be more pronounced for discriminatory users than for nondiscriminatory users. This was corroborated by the experimental results. The relationship between user discrimination and filtering performance is further explored in Section 4.2.

4.2 Simulation Experiments

The previous set of experiments provided a good understanding of the operation of the filtering system with real users. Several additional experiments were necessary, however, to clarify how users' roles (e.g., varying discrimination levels) and system-oriented parameters (e.g., the number of document classes) influence filtering performance. Conducting many experiments with actual users is expensive and, in terms of time requirements, can be impractical. Simulations can aid in investigating the influence of certain parameters that cannot be easily or practically tested. Hence, a simulation mode was included in SIFTER. In this mode, a user is represented as a profile with appropriate relevance values for the various classes. Our approach to simulation experiments, using realistic user profiles and documents, is highly similar to numerous studies conducted in IR that do not directly deal with users. However, to have further confidence in simulation results, we performed a correlation study between the results obtained in simulations and experiments with real users.

4.2.1 *Correlation with Baseline Data.* To start with, several users were asked to provide written descriptions of the areas they were interested in. These descriptions were not used in the baseline experiments but were used to create a simulation of the users. For example, to produce the simulated outcomes shown in Figures 9 and 10, a user model d (see the discussion in Section 3.3) was created based on the actual topical areas user 1 was interested in. Two separate experiments were then conducted with the same document set, one with the real user and the other with the simulated one. In the latter case, relevance values were probabilistically assigned to individual documents, based on the elements of the simulation model d. As can be seen from Figures 9 and 10, a high degree of correlation exists between results with the actual and the simulated users. The Pearson's correlation coefficient was 0.89 for normalized precision and 0.83 for normalized recall. Analysis based on information collected from other users also produced strong correlations.

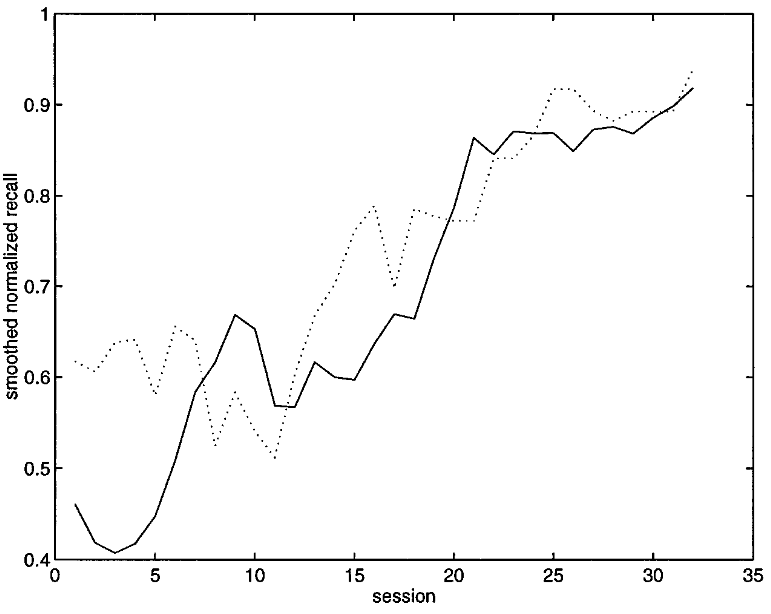


Fig. 9. Smoothed normalized recall data for user 1. Solid line: actual; dotted line: simulated.

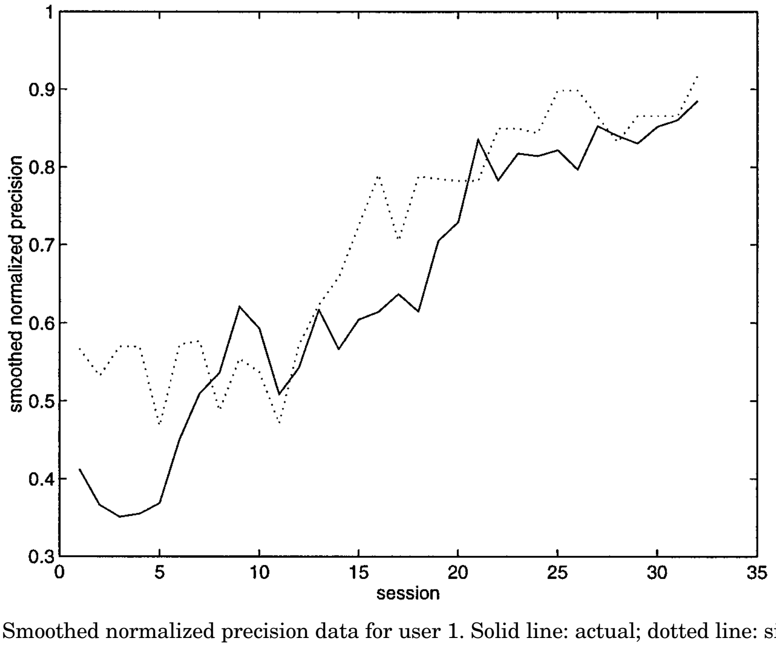


Fig. 10. Smoothed normalized precision data for user 1. Solid line: actual; dotted line: simulated.

4.2.2 *User-Oriented Experiments.* To analyze how users can directly influence filtering, several experiments were conducted where the dependent variables were user-oriented. Here, we present the results from three such experiments.

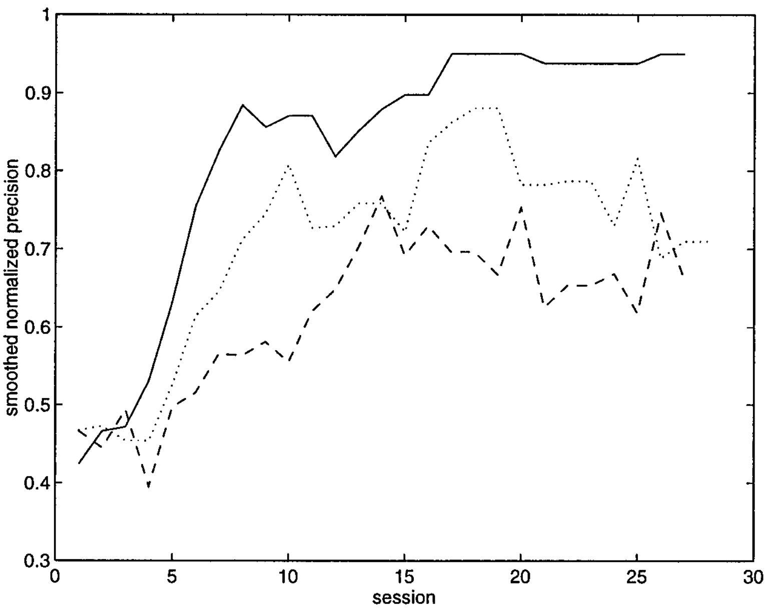


Fig. 11. Normalized precision data for three categories of users. Solid line: discriminatory; dotted line: moderately discriminatory; dashed line: nondiscriminatory.

Varying Levels of Discrimination. Using *d* as the simulated user model in SIFTER, three different users were created: discriminatory, moderately discriminatory, and nondiscriminatory. The same four classes were selected (i.e., "representation and semantics," "rule," "knowledge," and "heuristics"; see Figure 3) for all three users, while the relevance factors given to individual classes were varied to indicate different levels of discrimination. In the discriminatory user model, all four classes received a weight of 1.0. In the moderately discriminatory user model, the first two classes were given a value of 1.0, and the last two classes were given a value of 0.7. In the nondiscriminatory user model, the values assigned to the four classes were 0.65, 0.6, 0.7, and 0.6 respectively. For all three users, the remaining classes had zero relevance factors. In Figure 11, we present a comparison of the filtering outcome of the three user models, with the profile-learning mode turned on. As can be expected, the discriminatory user model produced the best performance among the three models. Another related and more interesting observed outcome was that, after the latency period, the discriminatory user model stabilized at a high level of performance, while the other two models continued to display oscillatory behavior. A possible explanation for such oscillatory behavior is the difficulty in ranking documents accurately when differences in interest levels in classes are not large. Although, theoretically, SIFTER's profile acquisition procedure can learn to distinguish even small differences among interests, in practice, such learning may require a large number of feedback iterations. The results achieved with the discriminatory user model closely matched the best outcomes of the baseline experiments: a steady rise to the maximum performance level followed by little or no oscillatory behavior. Similarly, the results achieved with the other two models were similar to the worse cases found in the baseline experiments: an increase in performance after the latency period followed by frequent oscillations.

Amount of User Feedback. Requiring the user to provide feedback for a large fraction of documents implies an increased burden on the user in terms of the number of documents to be read. At the same time, increased feedback results in better performance and lower latency time. In the experiments reported in Figure 11, the worst performance was found in the nondiscriminatory user case. It is for this type of user that the improvement in performance with increased user feedback is expected to be significant. To validate such a claim, the nondiscriminatory user model (as in Figure 11) was used in two different modes: standard feedback and increased feedback. In the former case, 7 documents out of 20 were given feedback in each session. For the increased-feedback case, all 20 documents were given user feedback. Due to space constraints, the results of this experiment are not presented graphically. However, to summarize, three significant improvements were observed in the outcome for the increased-feedback model. First, there was a steady increase in normalized precision to 0.9. Second, this increase was reached in less than 10 sessions (the latency period), and therefore, the latency period was successfully shortened by increasing the amount of user feedback. Finally, beyond the latency period (i.e., after 10 sessions), the outcome never fell below 0.85 and showed only minor oscillatory behavior. In contrast, the standard-feedback model resulted in a latency period of about 15 sessions, with the highest precision value of about 0.77, and with continued oscillatory behavior.

Nonstationary Users and Shift Detection. Several experiments were also conducted to analyze the operation of the shift detection module, that is, the rapid adaptation and recovery in response to significant changes in users' areas of interest. In the following, we present the results of one such experiment. For other experiments involving user interest shifts in a different filtering scenario (i.e., filtering LISTSERV emails), the reader is referred to Lam et al. [1996]. Keeping all other parameters constant, all experiments were conducted in two modes: one with the shift detection module and one without. As changes in interest are more likely to occur in long-term use of the system, experiments were conducted over 250 sessions.

In the experiment being reported, a moderately discriminatory user profile (i.e., d) was used. It was assumed the user held high interest in information retrieval and low to moderate interest in other classes. So, in d, classes 2 and 9 (refer to Figure 3) were set to 0.95 and 0.9 respectively, and all other classes were initialized to values ranging from 0.1 to 0.5. After session 150, to simulate a shift in user interest to the area of distributed operating systems, the value for class 17 was increased from 0.5 to 1.0, and the value for class 25 was increased from 0.4 to 1.0. The interest values for classes 2 and 9 were assumed to drop to 0.5.

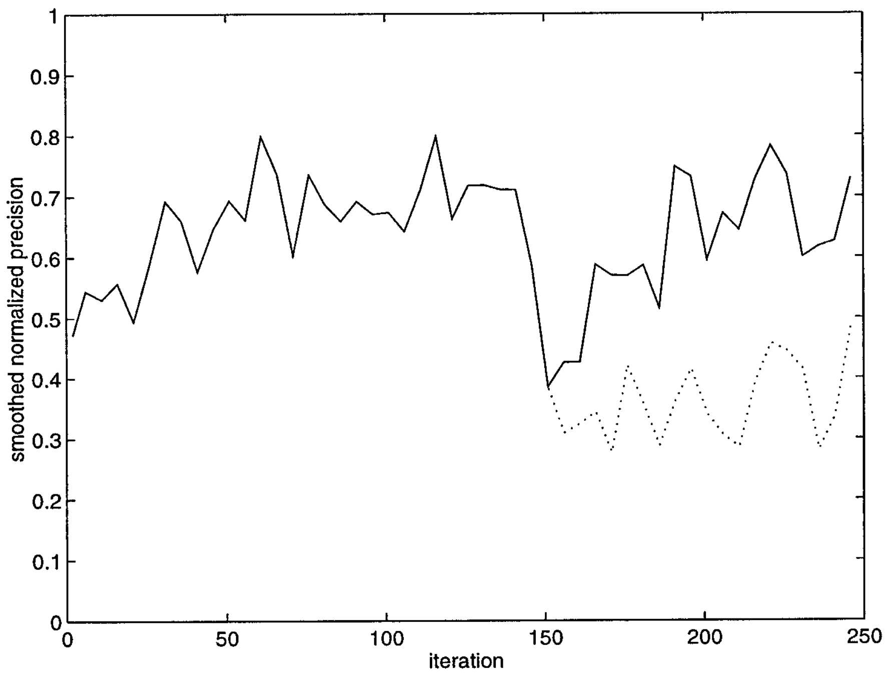


Fig. 12. Normalized precision for a nonstationary user. Solid line: with user interest shift detection; dotted line: without user interest shift detection.

As seen from Figure 12, the normalized precision of both trials reached above 0.7 just before the shift occurred (iteration 150). Just after the shift, the precision of the system without the shift detection module began to drop. At iteration 250, the precision was less than 0.5. The reason for such poor performance is that the system failed to recognize the loss of user interest in classes 2 and 9 (which used to be highly relevant classes). Hence, it continued to present documents from these two classes at the top. In contrast, the system equipped with the shift detection module showed a much better filtering performance under the same user scenario. All shifts involving classes 2, 9, 17, and 25 were successfully detected. In particular, the downward shift for class 2 was detected based on just four relevance feedback iterations after the shift (at iteration 151). The shifts corresponding to the other classes were all detected by iteration 196. The upward shifts required a longer detection time due to the fact that the degree of these shifts was not as large as the downward ones in this experiment. As a result of these successful shift detections, the system was able to discover the new user profiles. This, in turn, resulted in a higher normalized precision value of 0.75 at iteration 250. The precision improvement of the shift detection at iteration 250 was about 50% and showed an inclination for further improvement.

4.2.3 *System-Oriented Experiments.* To analyze how various system parameters can influence filtering performance, two additional sets of simulation experiments were conducted with SIFTER. The first involved the size of the document stream (i.e., the number of documents presented to the user in each session), while the second was concerned with the number of document classes.

The Size of the Document Stream. Three document stream sizes (10, 20, and 40) were chosen for experimental study. For each of these sizes, several sessions were run maintaining the same moderately discriminatory user model described earlier and a constant number of documents receiving feedback (7). Once again, we omit graphical presentation of the results due to space constraints and instead describe the outcome qualitatively. It was found that, during the latency period (the first 15 iterations), sessions with a stream size of 10 produced the best average normalized precision (0.81), and sessions with a stream size of 40 produced the worst average normalized precision (0.68). The sessions with 20 documents produced an average normalized precision of 0.64. This is not unexpected, since in the early period the profile contains very little information about the relevance of the classes, and increasing the number of documents per session only increases the likelihood of encountering documents from a wider variety of classes. In the postlatency period, however, the 40-document sessions generated the highest average precision value of 0.92; the 20-document sessions generated an average precision value of 0.76; and the 10-document sessions generated an average precision value of 0.81. With an increasing number of sessions, the number of different classes encountered increased, and this in turn contributed to the improvement in the user profile in the 40-document sessions. Another interesting result we noted was that the 40-document sessions in the postlatency period produced more consistent results than the sessions with fewer documents. The standard deviation of performance in the 40-document sessions in the postlatency period was 0.05, whereas in the 20-document sessions it was 0.24, and in the 10-document sessions it was 0.19. This suggests that the user profile learned in 40-document sessions was very robust.

The Number of Document Classes. In the filtering model proposed in this article, the user profile is learned over a document classification space and not directly over documents. Hence, the size of the classification space (i.e., the number of document classes) can have a significant influence on filtering. A small space would mean fewer classes and broader scope per class, implying a weak relationship between class centroids and member documents. A large number of classes would correspond to smaller classes, implying a strong and perhaps more accurate relationship between the class centroids and member documents. The final set of experiments concerned an investigation of the relationship between the size of classification space and filtering performance. For these experiments, we selected three different classification space sizes (22, 41, 62) and utilized SIFTER's relevance feedback mode. We again chose the moderately discriminatory user model described earlier (Figure 11). The same user model was used for

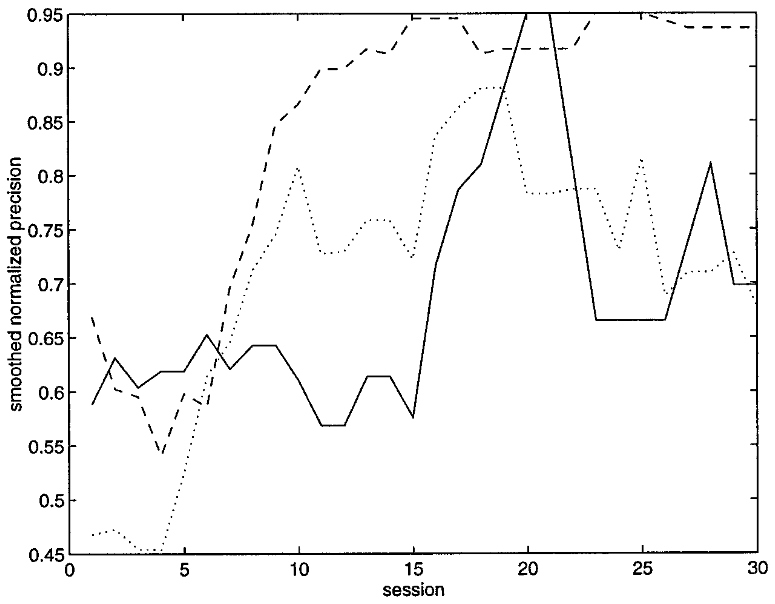


Fig. 13. Smoothed normalized precision data. Solid line: with 22 classes; dotted line: with 41 classes; dashed line: with 62 classes.

all sizes except in the size 22 case, where the class "heuristics" was dropped as it was not in the classification space.

For all sessions, the document stream size used was 20, and the number of feedbacks per session was fixed at 7. It was found that increasing the number of classes (i.e., the size of the classification space) resulted in a significantly improved filtering performance (Figure 13). Discounting the influence of latency (the first 15 sessions), we found that size 22 produced an average precision of 0.75; size 41 produced an average precision of 0.77; and size 62 produced an average precision of 0.94. In the postlatency period, the standard deviation of the best performance (size 62) was 0.04, whereas the standard deviation of the worst performance (size 22) was 0.26. The experimental results showed that with an increased number of classes the classification resolution is improved, and this in turn leads to better filtering performance. However, a point of diminishing returns is soon reached. For example, in this experiment, no significant improvement in performance was observed when the number of classes was increased beyond 62. Hence, considering that a large number of classes leads to inherent computational and learning complexities, an appropriate number of classes needs to be determined based on the trade-off between performance and complexity.

5. FUTURE EXTENSIONS OF SIFTER

In this section, we point out some of the open issues and problems that are currently being investigated. These relate to the flexibility of the system with respect to dynamic and changing document streams, the ability of the filtering system to cater to an individual user's notion of document classes, and problems involving collaboration between multiple filters serving different users. The successful resolution of these issues will improve the overall filtering performance and reduce the a priori effort needed to customize the system in a given context.

(1) *Thesaurus Adaptation*: As mentioned earlier, we used the ACM *Computing Reviews* Classifications Scheme and the ASIS Thesaurus to choose what we believe to be an adequate thesaurus for the domain of computer and information science. When such standardized sources are not available or are not believed to be adequate, there is a need to incorporate new terms automatically. Standard statistical approaches exist for automated thesaurus generation [Salton and McGill 1983]. Several other approaches based on machine learning and NLP techniques have also been reported in the literature for automatic term discovery [Futrelle et al. 1994; Guntzer et al. 1988] and refinement [Liddy et al. 1994]. Of course, users should also have the option to introduce new terms to suit their individual needs. Whenever there is a change in the thesaurus, there is also a need to update the centroids of the document clusters and/or reclustering.

(2) *Classification Scheme Adaptation*: Since the document classes are generated using an unsupervised clustering approach, they may not correspond to a particular user's notion of such classes. Hence, an unnecessarily large number of clusters may be needed to satisfy the assumption that user relevance is more or less constant across documents in a given cluster. Work is currently in progress to develop a variable resolution class adaptation scheme that starts with a small number of clusters and that splits large clusters depending on the degree of uncertainty of user feedback over each cluster.

(3) *Collaborative Filtering*: Preliminary experiments, involving multiple filtering systems serving multiple users, have strongly indicated that inadequacy of the initial thesaurus can be effectively overcome through consultation with other filters over the network. A simple scenario illustrating this situation is one in which a filter specializing in computer science documents collaborates with another in the medical domain to successfully represent and classify documents pertaining to medical computing. Thesaurus adaptation is only one area where such collaboration can be desirable. Huhns et al. [1987] described a distributed document environment where individual user models are used as filters to control the overall flow of information and to improve retrieval. Laskari et al. [1994] discussed the possibility of reducing the learning effort by transferring knowledge from an expert filter (one that has been running for a long time) to a novice (one that is newly initiated).

6. CONCLUSIONS

Information filtering, that is, ranking and presenting incoming documents according to a particular user's interests, is a user-oriented service whose importance can only increase dramatically as more and more users begin utilizing the vast information resources available via electronic media. To provide effective user-oriented filtering services, uncertainties associated with the representation and categorization of documents as well as user's interests have to be dealt with, and this requires integration of techniques from different areas such as information science and machine learning. In this article, we first discussed a general model for an information-filtering system that describes the broad functionalities desired of various subsystems. The model is general enough to permit the use of any preferred method for implementing the functionalities in a given context. A particular implementation of the system, called SIFTER, is also described, and is used to filter computer and information science technical documents. The present working version of SIFTER involves application and integration of several well-known techniques for document representation, clustering, user profile learning, and detection of changes in users' interests. Studies involving both real users and simulated users have been presented to illustrate the performance of SIFTER (as measured by well-established criteria such as normalized precision and recall) and the effects of various design parameters on the performance. Future extensions, including online adaptation of thesauri, adaptation of the classification tree, and collaboration between multiple filtering systems have also been discussed briefly. The general modular nature of the overall model permits the seamless integration of such functionalities in the future when deemed necessary.

Our current interpretation of the performance of SIFTER is that it performs very well, according to objective criteria such as precision and recall, when the users are reasonably familiar with the domain of information and are motivated to provide relevance feedback at the beginning of the learning process. The users tested had no difficulty in using the system and were able to understand intuitively the reasons for any deviation from their expected performance quite easily. Of course, user studies at a much larger scale are needed to validate any usability claims for SIFTER, particularly for lay users. However, in view of its success in the experiments performed and the urgent need to at least ameliorate the effects of information overload, the authors believe that such filters will be used widely in the coming years.

ACKNOWLEDGMENTS

The authors would like to express their sincere appreciation of the assistance provided by Derek Kern, Tanya Blekher, and Todd Kirk in developing the prototype of SIFTER. The comments and suggestions articulated by Lucy Xue, Andrew Olson, and Anand Hudli in many discussions are also gratefully acknowledged.

REFERENCES

BELKIN, N. J. AND CROFT, W. B. 1992. Information filtering and information retrieval: Two sides of the same coin. *Commun. ACM 35*, 12 (Dec.), 29–38.

BORKO, H. AND BERNICK, M. 1963. Automatic document classification. *J. ACM 10*, 2, 151–163.

EDWARDS, P., BAYER, D., GREEN, C. L., AND PAYNE, T. R. 1996. Experience with learning agents which manage Internet-based information. In *Proceedings of the AAAI Stanford Spring Symposium on Machine Learning in Information Access* (Palo Alto, Calif., Mar.). AAAI Press, Menlo Park, Calif.

FISCHER, G. AND STEVENS, C. 1991. Information access in complex, poorly structured information spaces. In *Proceedings of ACM Special Interest Group on Human Computer Interaction Annual Conference* (New Orleans, La., Apr. 27–May 2). ACM, New York, 63–70.

FRANTS, V. I., KAMENOFF, N. I., AND SHAPIRO, J. 1993. One approach to classification of users and automatic clustering of documents. *Inf. Process. Manage. 29*, 2, 187–195.

FUTRELLE, R. P., XIAOLAN, Z., AND SEKIYA, Y. 1994. Corpus linguistics for establishing the natural language content of digital library documents. In *Digital Libraries*. Lecture Notes in Computer Science, vol. 916. Springer-Verlag, New York, 165–180.

GOKER, A. AND MCCLUSKEY, T. L. 1991. Toward an adaptive information retrieval system. In *Proceedings of 6th International Symposium* (Charlotte, N.C., Oct. 16–19). ISMIS, 348–357.

GUNTZER, U., JUTTNER, G., SEEGMULLER, G., AND SARRE, F. 1988. Automatic thesaurus construction by machine learning from retrieval sessions. In *Proceedings of RIAO: User-Oriented Content-Based Text and Image Handling* (Cambridge, Mass., Mar. 21–24). MIT, Cambridge, Mass., 587–596.

HUHNS, M., MUKHOPADHYAY, U., STEPHENS, L. M., AND BONNELL, R. D. 1987. DAI for document retrieval: The Minds project. In *Distributed Artificial Intelligence*. Pittman, London.

LAM, W., MUKHOPADHYAY, S., MOSTAFA, J., AND PALAKAL, M. 1996. Detection of shifts in user interface for personalized information filtering. In *Proceedings of the 19th International Conference on Research and Development in Information Retrieval* (Zurich, Switzerland, Aug. 18–22).

LANG, K. 1995. NewsWeeder: An adaptive multi-user text filter. Tech. Rep., School of Computer Science, Carnegie Mellon Univ., Pittsburgh, Pa.

LASKARI, Y., METRAL, M., AND MAES, P. 1994. Collaborative interface agents. In *Proceedings of AAAI Conference* (Seattle, Wash., Aug. 1–6). AAAI Press, Menlo Park, Calif.

LEWIS, D. D. 1992a. Representation and learning in information retrieval. Dissertation, Dept. of Computer and Information Science, Univ. of Massachusetts, Amherst, Mass.

LEWIS, D. D. 1992b. Text representation for intelligent text retrieval: A classification-oriented view. In *Text-Based Intelligent Systems: Current Research and Practice in Information Extraction and Retrieval*, P. S. Jacobs, Ed. Lawrence Erlbaum, Hillsdale, N.J., 179–197.

LEWIS, D. D. AND TONG, R. M. 1992. Text filtering in MUC-3 and MUC-4. In the *4th Message Understanding Conference* (*MUC-4*) (McLean, Va., June 16). 51–66.

LIDDY, E. D., PAIK, W., AND YU, E. S. 1994. Text categorization for multiple users based on semantic features from a machine-readable dictionary. *ACM Trans. Inf. Sys. 12*, 3 (July), 278–295.

MALONE, T. W., GRANT, K. R., TURBAK, F. A., BROBST, S. A., AND COHEN, M. D. 1987. Intelligent information sharing systems. *Commun. ACM 30*, 5 (May), 390–402.

MUKHOPADHYAY, S., MOSTAFA, J., PALAKAL, M., LAM, W., XUE, L., AND HUDLI, A. 1996. An adaptive multi-level information filtering system. In *Proceedings of the 5th International Conference on User Modeling* (Kailua-Kona, Hawaii, Jan. 2–5). 21–28.

MYAENG, S. H. AND KORFHAGE, R. R. 1990. Integration of user profiles: Models and experiments in information retrieval. *Inf. Process. Manage. 26*, 6, 719–738.

NARENDRA, K. S. AND THATHACHAR, M. A. L. 1989. *Learning Automata—An Introduction.* Prentice-Hall, Englewood Cliffs, N.J.

OARD, D. 1996. Information filtering resources. Univ. of Maryland, College Park, Md. Available as http://www.ee.umd.edu/medlab/filter.

RICH, E. 1983. Users are individuals: Individualizing user models. *Int. J. Man-Mach. Stud. 18*, 199–214.

SALTON, G. 1989. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, Reading, Mass.

SALTON, G. AND MCGILL, M. J. 1983. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York.

SETH, B. D. 1994. A learning approach to personalized information filtering. M.S. thesis, Electrical Engineering and Computer Science Dept., MIT, Cambridge, Mass.

THATHACHAR, M. A. L. AND SASTRY, P. S. 1985. A new approach to the design of reinforcement schemes for learning automata. *IEEE Trans. Syst. Man Cybern. 15*, 168–175.

TOU, J. T. AND GONZALEZ, R. C. 1974. *Pattern Recognition Principles.* Addison-Wesley, Reading, Mass.

YAN, T. Y. AND GARCIA-MOLINA, H. 1995. Sift—A tool for wide-area information dissemination. In *Proceedings of the* 1995 *USENIX Technical Conference*. USENIX Assoc., Berkeley, Calif., 177–186.

YANG, Y. AND CHUTE, C. 1994. An example-based mapping method for text categorization and retrieval. *ACM Trans. Inf. Syst. 12*, 3 (July), 252–277.

ZACKS, S. AND BARZILY, Z. 1981. Bayes procedures for detecting shift in the probability of success in a series of Bernoulli trials. *J. Stat. Plan. Inference 5*, 117–119.

Received July 1995; revised September 1996 and February 1997; accepted February 1997