An Adaptive Multi-level Information Filtering System

S. Mukhopadhyay*, J. Mostafa+, M. Palakal*, W. Lam*, L. Xue*, and A. Hudli*

*Computer and Information Science
Purdue University School of Science at Indianapolis
723 W. Michigan St. SL280, Indianapolis, IN 46202
Phone: (317) 274 - 9732. E-mail: dai@iupucs.iupui.edu
+School of Library and Information Science
Indiana University, Bloomington, IN 47402

Abstract

To conduct efficient information filtering, uncertainties occurring at multiple levels must be managed. Uncertainties can occur due to changing document space as well as stochasticity and non-stationarity of the user. A new filtering model based on established techniques in information retrieval and artificial intelligence is proposed. These techniques include document representation by vector-space model, document classification by unsupervised learning, and user modeling by reinforcement learning. A filtering system, named SIFTER, has been developed based on the model. The system can filter information based on content and user's specific interests. The user's interest is automatically learned with little user intervention in the form of optional relevance feedbacks for documents. SIFTER was used to filter typical long and short documents transferred via e-mail in the Internet. The experimental results demonstrate that the system performs very well in filtering documents in a realistic problem setting.

Introduction

Internet, the global computer network, is one of the largest sources of online information in the world. Presently, approximately 10 Terabytes of data is transferred per month over the NSFNET, an Internet backbone (Berners-Lee *et al* 1994). Providing effective access to such a large volume of data presents some extremely difficult challenges. At a macro-level (overall network level), there are problems associated with heterogeneity of sources of data and the demands of a very large distributed user base. However, at micro-level (individual user level) there are also certain fundamental problems that need attention. Users bring their unique information needs to the Internet. To satisfy these needs they need selection systems that can be easily manipulated to match their personal interests. Belkin and Croft (Belkin & Croft 1992) compared two major methods for information selection: filtering and retrieval. Filtering refers to information selection based on long-term information need whereas retrieval refers to information selection based on short-term information need (Belkin & Croft 1992). For effective filtering to occur, accurate information about both the document space and the user must be maintained.

One important application domain that demands efficient information filtering is Internet e-mail communication which is now used for more than personal communications. Client interfaces to popular Internet information services such as USENET, WWW (World-Wide-Web) and WAIS (Wide Area Information Server) support e-mail function for transferring documents. The highly popular distribution list services (also known as Listservs or mailing lists) utilize e-mail to transfer information. Assuming users wish to maintain regular communication with colleagues, occasionally transfer documents using e-mails, and wish to keep up-to-date with discussions in distribution lists, a steady inflow of e-mails can quickly reach a large volume. Managing such a large volume of information can be extremely time-consuming. A good solution to this problem is a filtering system that can order the e-mails according to their relevance to the user, with minimal user intervention. To perform such ordering, the filtering system must capture and maintain a user interest profile (*i.e.*, a model of the user's relevance for various topical areas), and order e-mails according to the profile.

There are, however, certain basic problems that an information filtering system has to deal with. The dynamic nature of the filtering environment can make it extremely difficult to gather and maintain information about the document space and users. New topics may be introduced in the document space or user's interest related to topics may change with changes in personal situations. These changes, viewed from the filtering system's perspective, are sources of uncertainties. Managing these uncertainties require a high level of adaptivity on the system's part which can be achieved by applying learning techniques.

Problem Description

The general problem of information filtering may be posed as determining a map from a space of documents to the space of user relevance values. Denoting the space of documents as $\mathcal{D}$, the objective is a to find 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.

The problem of learning, in this context, corresponds to the case when f is not known *a priori* and has to be estimated on-line from user feedback. This could, in principle, be accomplished by setting up some form of a parametrized map approximator (such as, artificial neural networks) and updating the parameters based on the user feedback. One of the principal arguments used in the paper is that such a direct on-line learning of the map f is computationally extremely intensive and will require a very large number of user feedbacks, considering the high dimensionality of any reasonable representation of the documents. In order to provide a practically feasible solution to the filtering problem, we decompose the problem into two levels. The higher level represents a classification mapping f_1 from the document space to a finite number of categories $\{C_1, \ldots, C_m\}$ (*i.e.*, $f_1 : \mathcal{D} \rightarrow \{C_1, \ldots, C_m\}$). This mapping is learned in an off-line setting, based on a database of documents, either using *prior* information concerning the categories and examples, or by automatically discovering the abstractions using a clustering technique. Once the mapping f_1 is learned, the resulting clusters are not modified during user interaction. The higher level thus partitions the document space into m equivalent classes over which the user relevances are estimated. The lower level consequently estimates the mapping f_2 describing the user relevances for the different categories (*i.e.*, $f_2 : \{C_1, \ldots, C_m\} \rightarrow \mathbb{R}$). Since f_2, unlike f and f_1, deal with a countable input set of relatively few points, the on-line learning problem of f_2 is not unrealistically time-consuming. The decomposition of f into f_1 and f_2 clearly limits the maximum achievable accuracy of the filtering system, since a category may not correspond to a constant user interest. However, the authors believe that the resulting inaccuracy is more than adequately compensated by the substantial reduction in learning complexity. If a 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 could be initialized on the basis of learned f_1 and f_2. From then onwards, the general map can be used for ranking purposes and can be updated on the basis of user feedback.

The on-line learning problem, in both the direct case and the two-level case, is further made difficult by the following factors:

(i) Difficulty in Representation: In general, it is not possible to represent $\mathcal{D}$ exactly by a finite-dimensional space that corresponds to some features of the documents. Hence, any finite dimensional representation space X is merely an approximation to $\mathcal{D}$, and there is always a loss of information in the process. The entire area of document representation and indexing (Salton and McGill, 1983) is devoted to discovering methods for such finite-dimensional representation that minimizes the information loss in some sense. In a dynamic environment, to make the problem more difficult, the most preferable representation scheme may also be a function of time.

(ii) Stochasticity of User Relevance Feedbacks: The user feedback may appear to be random to the filtering system due to the presence of features or keywords in the document that are not incorporated in the document representation scheme (*i.e.*, due to missing features). Further, in the two-level approach, the feedback may appear random to the lower level learning agent due to the partitioning of the document space. The user may not have the same interest level for all documents belonging to a particular class. Finally, there may arguably be inherent randomness in the users, particularly the uncertain ones. In all such cases the maps f and f_2 appear as stochastic ones to the filtering system. The binary (relevant/non-relevant) nature of the user feedback introduces an additional difficulty for the learning system since it has to determine the real-valued underlying (average) level of interest from the discrete realizations of the corresponding random variables.

The contributions of this paper can now be summarized as follows.

- A two-level model for information filtering is developed that is applicable to diverse types of textual information. We believe the generality of the solution will be important for filtering information in a distributed environment (*e.g.*, the Internet) where it is very difficult to justify any assumptions concerning the structure and content of the documents.

- We propose a method, based on reinforcement learning algorithms studied in the AI community, for modeling and generating profiles of a user's interest in the various categories of information. The categorization itself is accomplished by means of a vector-space document representation followed by a clustering algorithm.

- We describe a preliminary working system called SIFTER (Smart Information Filtering Technology for Electronic Resources) for the specific application domain of filtering listserv e-mails on the Internet. We present some experimental results that show promising improvement in the performance of the systems with learning.

Currently, the model relies on direct user intervention to deal with difficulty (i), (*i.e.*, the inadequacy of the representation scheme in a dynamic information environment). The problem of automatic adaptation of the representation scheme is a topic for future work.

Related Work

Information filtering is receiving considerable attention from researchers involved in information retrieval, database management, and, more recently, artificial intelligence. Good summaries of various research projects related to this area can be found in two special issues of ACM Communications (special issue on information filtering 1992 & special issue on intelligent agents 1994). The most promising paradigm proposes that the system should be endowed with sufficient knowledge about the user and various system-oriented components (major functions and the information domain) to act as an active agent performing tasks on user's behalf, collaborating on complex tasks, and generally hiding the complexity of difficult tasks from the user (Maes 1994). This new paradigm is called agent-based interaction. The most difficult challenge in agent-based systems is the accumulation of reliable knowledge for the agent. The problem of developing intelligent agents is very similar to the problem of reducing uncertainties in information filtering systems and several works in this area provided useful insights.

Patti Maes (Maes 1994) describes three major approaches currently applied to supply and update agent knowledge. In one approach a particular type of agent, called "semi autonomous agents" is built by using information about interest areas directly provided by the user. The Information Lens system, used for e-mail filtering, is an example of such a system (Malone *et al* 1987). It requires the user to build simple rules describing their interest area. This particular approach relies on explicit user-supplied knowledge. Hence, it permits the users to comprehend and follow agent's "up-to-the-moment" knowledge state. However, this approach suffer from several disadvantages. Users may not be sure of their information need. Also they may not have the competence nor the motivation to create and update the agent knowledge. A second approach utilizes knowledge-engineering to develop knowledge-bases covering two specific areas: detailed knowledge of the application environment and knowledge of how users typically perform tasks (Maes 1994). Using the knowledge-base, the agent can automatically recognize user's plans and contribute toward efficient task execution. The knowledge-based approach has been used in developing intelligent help systems to support interaction and learning in complex domains (Selker 1994).

This approach, however, suffers from extensive initial overhead of knowledge-base creation. An additional problem with this approach is that the knowledge-base cannot be changed easily – it cannot be customized to match user's changing preferences or their expertise associated with tasks.

Two recent works have applied machine learning to overcome some of the disadvantages of the other methods in acquiring agent knowledge. Maxims (Metral 1993) is an e-mail filtering agent that relies on memory-based learning to generate agent knowledge. Maxims can be useful for filtering simple and brief e-mails. The main weakness of the Maxims system is that it is not content-sensitive – it only uses header information (To:, From:, Subject:, etc.) of e-mails and ignores the body of e-mail text. Another system named NewT (news tailor) utilizes relevance feedback and genetic algorithms as means to acquire agent knowledge (Seth 1994). NewT is content-sensitive and it demonstrated that agent knowledge can be acquired with minimal user intervention. However, the weakness of this system is that it requires substantial amount of time to map documents to user relevance values, and as a result filtering cannot be carried out in real-time.

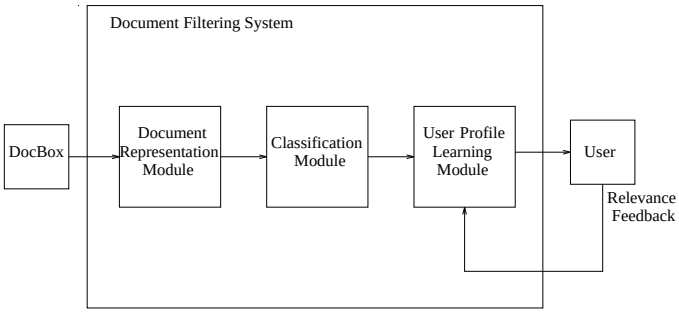

Figure 1: Functional architecture of the overall filtering system

In this section, we have discussed several approaches to acquiring knowledge about the user and the application environment to support agent-based information filtering. Some of the weaknesses of these approaches include: (1) too much dependence on the user for agent-knowledge creation and update, (2) high up-front cost of knowledge engineering, (3) inflexibility of using static knowledge to match changing user interest, (4) content-insensitivity and (5) inordinate amount of time for mapping between documents and user relevance values.

In our system, we have attempted to overcome these limitations. The salient features of our system include the following: it uses machine learning to quickly create and update user interest profiles, it can quickly map a document to user relevance values, it can flexibly deal with changing user interest, and it requires minimal user intervention.

Overview of the Filtering System

An essential feature of our system is the integration of several modules, each serving a definite purpose and each using a different learning technique. In this section we provide a brief overview of the system and the relationships of its components. In subsequent sections, we describe each of the components in more detail. The architecture of the over-

all system is shown schematically in Figure 1. The input to the system consists of the incoming documents denoted by the DocBox in Figure 1. The filtering system presents these in a sorted fashion to the user. In order to perform its sorting tasks efficiently, the filtering system makes use of several modules, and operates in a feedback configuration with the user.

The problem of information filtering was decomposed into learning two maps f_1 and f_2 in section 1; while f_1 was determined in an off-line manner, f_2 was learned through interaction with the user. The two modules in the model dealing with the problem of learning f_1 are the document representation module and the classifier module. The document representation module determines a finite-dimensional vector description of a document (the input space for f_1), and the classifier module finds the category to which a document belongs to (the output for f_1) . The user profile learning module is responsible for the on-line learning of f_2 (*i.e.*, the map from the categories to the relevance values).

Components of the SIFTER System

In this section, we describe the methods used to implement the three modules described above in the present SIFTER system.

Document Representation Module

The main purpose of this component is to convert documents arriving in the DocBox into structures that are representative of original documents and are easily parsable by the component in the next level. By representativeness we mean maintaining the essential content of the documents not their format.

Document Representation using Vector-Space Model There are several methods for converting textual documents to representative structures. Most of these methods have their origins in classical IR (Information Retrieval) research. Salton in (Salton & McGill 1983) has described a widely used methodology for generating representative structures. This method known as vector-space model accepts as input a set of textual documents and produces as output a set of vectors (representing individual documents) whose locations in the document space is fixed based on multiple weights (dimensions) assigned to them. Salton described several methods for generating weights for the vectors. The simplest technique (therefore easiest to automate) is to generate frequencies for individual terms (synonyms are treated as the same term) appearing in documents and use those frequencies as weights. This is the method used in the current implementation of SIFTER. As there are an extremely high number of words in the English language not all terms can be used as candidates for weight generation. Therefore, certain terms must be pruned from the candidate pool.

Term Pruning: Certain short function words such as *a, and, or,* and *the* carry little meaning. Therefore, to reduce the computational complexity of the document representation scheme, they can be eliminated without losing representativeness. Elimination of function words can easily be automated by using a list of these terms as the negation set (known as "stop list"). Beyond the short function words, depending on the domain, certain other terms may be identified that have low discrimination values. These terms tend to be very broad in scope (for e.g., the term "computer" in the domain computer science). To eliminate these terms the vector generation system must impose a domain boundary on the documents. The vector generation system can impose a domain boundary by using an online thesaurus that includes only those terms considered most relevant (highest discrimination value) in a domain.

Weight Value Generation: Once all documents are pruned so that only the relevant terms are retained, they are ready for conversion to weight vectors. For a document set, a table is generated that contains the total frequencies of all unique terms. Next, for each document in the set, another table is generated that contains the frequencies of terms occurring in individual documents. Then to generate weights for each vector based on the term frequencies the following is used

$$W_{ik} = T_{ik} \times I_k$$

where T_{ik} is the number of occurrences of term T_k in document i and I_k is the inverse document frequency of the term T_k in the collection of documents. A commonly used measure for the inverse document frequency is

$$I_k = \log(N/n_k)$$

where N is the total number of documents in the set, of which n_k documents contain a given term T_k. The inverse document frequency measure helps to reduce the weights of those terms that appear more frequently relative to others.

Classification Module

The scope of the vector classification module is to associate each incoming document vector V^i to a particular category class C^k. The categories generated by the classification module are different from the topical divisions that can be obtained from the listservs. The topical divisions inherent in the listservs do not provide the necessary discriminatory capabilities required for good classification. This module consists of two processing stages: an unsupervised learning stage and a vector classification stage. During the learning stage, an initial cluster hypothesis $[C^1, \cdots, C^n]$ is generated for a given set of document vectors $[V^1, \ldots, V^N]$. During

[1]Throughout the paper, the terms 'category' and 'class' are used interchangeably in the context of documents

the second stage (*i.e.*, the on-line classification stage), an incoming document vector V^i is classified into a particular class C^k using the learned cluster centroids from stage 1.

Unsupervised Learning of Clusters: A sample set of document vectors are used in order to obtain an initial hypotheses about various categories of the documents. The initial hypotheses generation is based on a minimum distance classification technique similar to the $Maximin - Distance$ clustering algorithm (Tou & Gonzalez 1974).

Each vector V^i that corresponds to a document has two types of inherent meaning to it: (i) each document vector V^i is composed of a set of $topics$ and, (ii) each topic is composed of a set of $terms$. That is, the set of terms used to represent a document can be decomposed into a set of topics with each topic containing several terms. The topics and terms represent a two-level hierarchical thesaurus which is based on vocabularies standardized by professional organizations such as the ACM. The categories, on the other hand, are discovered automatically by the classification algorithm based on a representative sample of documents and do not correspond one-to-one with the topics. The terms within a topic are closely related (but not synonyms, *e.g.*, 'programming' and 'languages'). Hence, the distance between two vectors are computed on the basis of topical information rather than the individual terms. We represent each document vector as, $V^j = [V^j_{pq}]$, $p = 1, \cdots, T$ and $q = 1, \cdots, K_p$, where V^j_{pq} represents the numerical weight assigned to the q^{th} term in p^{th} topic of document j. In order to take topical information into consideration during distance computation between two document vectors V^i and V^j, the significance of a topic p is first computed for both the documents: $W^i_p = \sum_{l=1}^{K_p} V^i_{pl}$ and $W^j_p = \sum_{l=1}^{K_p} V^j_{pl}$. The above equations reduce the space of vectors V^i and V^j to the space of vectors W^i and W^j. The actual distance can be now computed as,

$$\|V^i - V^j\| = \|W^i - W^j\| = \sqrt{\sum_{p=1}^{T} (W^i_p - W^j_p)^2}$$

The qualitative effect of considering distances between the topics rather than the terms is to encourage the clustering algorithm to form clusters around each of the topics. However, the actual number of clusters formed may be significantly more than the number of topics, since clusters may be formed overlapping two or more topics.

User Profile Learning Module

The user profile learning module consists of a learning agent that interacts directly with the user. This subsystem sorts the incoming documents according to its belief of the preferences of a particular user. To accomplish this task, the learning agent maintains and updates a simplified model of the user. The user is modeled by means of an *estimated relevance vector*, which is used during the learning phase to sort the incoming information. On the basis of limited user feedback (the *user's* concept of relevance), the learning agent updates its estimates so as to improve its performance even while interacting with a less than certain user.

The problem, as stated, fits very well with the *reinforcement learning* paradigm studied extensively in the Artificial Intelligence and Mathematical Psychology communities for the purpose of modeling learning in animal behaviors. One such learning model using reinforcement learning is the learning automaton (Narendra & Thathachar 1989). In its simplest form, such a model consists of an automaton with a finite action set operating in feedback configuration with an environment (teacher). The automaton chooses an action from its action set according to a probability distribution over the action set (which is initially equal for all the actions). As a result of performing the selected action, it receives a random feedback (termed reinforcement) from the environment. The reinforcement may be binary (reward/punishment, success/failure) or may take a discrete or continuous set of values in a range. The environment can be mathematically modeled as a conditional probability distribution over the reinforcement set given an action chosen by the automaton. The environment distributions are assumed to be unknown to the learning agent, and the only information it receives concerning the environment is through the reinforcements (which are samples from the environmental reinforcement probability distributions). The problem is to determine how the automaton should update its action probabilities on the basis of stochastic reinforcement feedback from the environment so as to improve its expected reward in some sense. The two aspects of reinforcement learning approach that makes it particularly suitable in the context of information filtering are the real-time learn-while-perform aspect and its capability to perform in a random environment.

We first state the assumptions that are made concerning the user and the incoming documents.

The User: The user is modeled by a set of unknown relevance probabilities defined over the set of categories. The user relevance feedback is assumed to be binary (0: not relevant, 1: relevant). Denoting the categories of documents by $C^1, \ldots, C^n$, d_i is used to denote the probability that the user will provide a relevance value of 1 ('interesting') to a document belonging to the class C^i.

Arrival of documents: It is assumed that as the number of arriving documents tends to infinity, all categories will occur infinitely often. They may arrive with different frequencies but will not influence the eventual outcome of learning.

The Learning Algorithm:

For our current work, we chose a particular algorithm, termed pursuit learning algo-

rithm (Mukhopadhyay & Thathachar 1989), for the following two reasons. Firstly, in empirical studies this algorithm appeared to have the attractive property of fast convergence. Secondly, the filtering problem requires that the categories (actions of the learning automaton) be rank-ordered, while most learning algorithms merely determine the optimal action. Since, as described later in this section, the pursuit learning algorithm maintains and updates an estimated relevance vector (unlike many other learning algorithms), the latter can be used to rank-order the different categories of information.

The learning agent keeps two vectors of dimensions equal to the number of categories. 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 learning agent is assumed to have no *a priori* knowledge of the user preferences. Hence, all the elements of $\hat{d}$ vector are initialized to zero. In addition to the $\hat{d}$ vector, the learning agent also keeps an action probability vector $p = [p_i]$, such that p_i represents the probability that the category C^i is the first category of documents to present to the user. In the absence of any *a priori* knowledge, all elements of p vector are initially made equal to each other, *i.e.*, p_i, ($i = 1, \ldots n$) are initialized to $\frac{1}{n}$. Both p and $\hat{d}$ vectors are updated during the learning process on the basis of user relevance feedback.

The learning agent at every iteration sorts the documents in the DocBox according to the following scheme. An action (category) is selected at instant k by sampling the probability distribution p. For an example, consider a p-vector equal to $[0.2, 0.4, 0.1, 0.3]$. A pseudo-random number (PRN) with uniform distribution is generated between 0 and 1. Then category 1 will be selected if the PRN lies between 0 and 0.2, category 2 will be selected if the PRN is between 0.2 and 0.6, category 3 is selected if the PRN $\in [0.6, 0.7]$, and category 4 will be selected in all other cases. The probabilistic method of selecting a category, which is the one to be presented to the user at the top of the sorted list, initially provides the learning scheme the flexibility to explore all categories. As the learning progresses, the p vector will converge to a unit vector, and hence a single category will emerge as the most relevant one.

Let $\alpha(k)$ be the category selected using this procedure. A vector $q(k)$ with n elements is computed such that

$$\begin{aligned} q_i(k) &= 1 + \hat{d}_i(k) && \text{if } \alpha(k) = C^i \\ &= \hat{d}_i(k) && \text{otherwise} \end{aligned}$$

where $\hat{d}_i(k)$ is the i^{th} element of $\hat{d}$ vector at instant k. Since $\hat{d}_i(k)$ is bounded by 1, the maximum element of $q(k)$ always corresponds to the category chosen as $\alpha(k)$. The remaining elements of $q(k)$ have a relative ordering the same as those of $\hat{d}_i(k)$.

Let $M_1(k), \ldots, M_m(k)$ denote the documents in the DocBox at instant k. Then they are sorted according to their categories and the values of the corresponding elements in $q(k)$ vector. That is, all the documents belonging to the category which corresponds to the largest element of $q(k)$ are placed at the top, followed by documents belonging to the category corresponding to the second highest element of $q(k)$, and so on.

The learning algorithm (*i.e.*, the algorithm for updating $p(k)$ and $\hat{d}(k)$) is described below. $\hat{d}_i(k)$ ($i = 1, \ldots, n$) at any instant is computed as the running average of the relevance values given by the user for category i. Denoting the current maximum element of $\hat{d}$ vector as having the index l, a unit vector $E(k)$ is created of dimension n whose l^{th} element is 1, and whose all other elements are zero. Then $p(k)$ is updated as

$$p(k+1) = p(k) + \eta(E(k) - p(k))$$

where $0 < \eta < 1$ is the learning step-size that is chosen empirically through trial-and-error. The motivation behind the pursuit learning algorithm is clear. An optimal unit vector is computed at every iteration for the p vector based on the current estimates of the relevance probability vector (*i.e.*, $\hat{d}$ vector), and the p vector is moved by a small distance towards the optimal unit vector.

Graphical User Interface and SIFTER Engine

Users interact with SIFTER by using a graphical user interface (GUI). The GUI performs three major functions: (1) it gathers all recent documents and displays them in a ranked list (based on the ranking provided by the user profile learning module), (2) it allows users to select individual documents and read them, and (3) it collects user feedback and passes the feedback to the user profile learning module. When a ranked list of e-mails are shown in a GUI window, users can scroll the window and select any document using the mouse. The body of the e-mail selected by the user is then shown in a second window. At the bottom of the second window, the user is prompted to indicate the relevance of the e-mail by entering a 0 for non-relevant and 1 for relevant. The user can close the window without entering a relevance value. The present version of the GUI is designed to function in the X Window environment. Preliminary use analysis of the SIFTER GUI has shown that the major functions are easy to understand and use.

The control flow of the entire system is coordinated by a filter engine which makes system calls to the individual modules. The engine is activated by the user whenever he/she wishes to read filtered mails. When activated, the engine calls the document representation module. The latter reads the new mails collected by the operating system mail daemon. Weight vectors are generated for the new mails and stored in a file. The control then returns to the engine which invokes the classification module. The classifier reads the weight vectors and outputs their classes into another file.

The engine subsequently activates the profile learning module that ranks the e-mails according to their classes and the current user profile. The control then passes to the GUI which presents the ranked e-mails to the user as described earlier. If any user relevance feedback has been collected by the GUI, these are stored in a file. The engine then invokes the user profile learning module again to update the user model based on the relevance feedbacks. This constitutes one complete cycle of the engine. The user can initiate a new cycle or terminate the engine through the GUI. The various modules and the engine were implemented using C in a Unix environment.

Experimental Results

The SIFTER system was experimentally tested on the problem of filtering of e-mails. A set of thirty keywords were chosen to represent e-mails pertaining to the overall domain of computer science. The present operation of the system represents a scenario in which a user has subscribed to and is receiving mails from several Internet LISTSERVers related to different areas in computer science. The classification module was trained in an unsupervised fashion, using a small database of representative mails, to classify the incoming mails to one of twelve possible classes (*e.g*, 'programming', 'operating system', 'distributed systems','database', 'information retrieval','learning', 'indexing', 'file systems', 'concurrency control', and 'networks'). In addition, there were two additional classes called 'digest' (to represent mails which cover many areas) and 'others' (to represent mails which do not contain any of the keywords selected). While the classes are not exhaustive, our intention is merely to demonstrate the feasibility of the overall approach in a nontrivial application domain.

The system presents mails to the user, sorted according to its current belief concerning the user interests, every time ten e-mails have been received, and updates the user profile on the basis of relevance feedback from the user. It is assumed that the user provides feedback only to the first three e-mails in the sorted list. The system performance was measured in terms of *average relevance*, which was computed as the fraction of times a positive relevance was received from the user to the total number of user feedbacks. The objective of the learning agent is to maximize the value of the average relevance, given its past experience with the user.

Several experiments were carried out to estimate the efficacy of the filtering system. We present only two results obtained with two different kinds of user: a discriminatory user, and an intermediate average user. In each case, we compare the performance of the system with learning (*i.e.*, with sorting of mail according to estimated user profile), and without learning (*i.e.*, when mails are presented in the same order as they were received). In all cases, the total relevance is initialized to zero. Hence, the initial rising part in the graphs for the 'no-learning' case does not correspond to any learning behavior, but merely to the time interval required by the average reward to reach the steady-state value.

Discriminatory User: The improvement in the system performance was most prominent in the case when the user relevance factors for the different classes were quite disparate. This situation represents a user who has very definite preferences concerning the different classes. Such a user was simulated with relevance factors equal to 0.9 for four out of the twelve classes (the user had strong preferences for e-mails classified to the areas 'programming', 'operating system', 'learning' and 'information retrieval'). The values for the other classes were set to 0.1 (thus, the user was quite disinterested about the mails belonging to the other 8 classes).

The evolution in the average relevance value obtained by the system in the 'learning' and 'no learning' cases is shown in Figure 2, where each iteration corresponds to a presentation of 10 mails. Thus the total time span shown in the figure corresponds roughly to about 2,000 mails. Assuming that a user receives about 50 mails a day (*i.e.*, 5 presentations), the time interval in real time corresponds to about 40 days. However, as can be seen from Figure 2, the average relevance with learning reaches a high value (about 0.7) within a span of less than 50 iterations (10 days), and then continues to improve only gradually. The maximum value of the average relevance attained was about 0.8 which was roughly twice the value received when no learning took place. No attempts were made to fine tune the performance of the system by adjusting design parameters such as the step size in the learning algorithm.

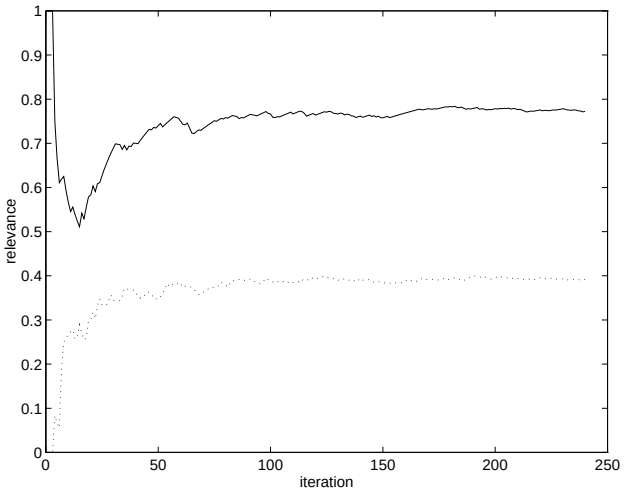

Figure 2: The average relevance received by the e-mail filtering system for a discriminating user with and without learning; solid line: with learning, dotted line: without learning

Average User: In this case, one of the coauthors of this pa-

per was asked to enter his preferences for all the categories. This particular user entered high values (0.9 and 0.7 respectively) for the categories 'information retrieval' and 'learning', and low values (0.1, 0.1, 0.2, 0.2, 0.2) for the classes 'others', 'digest', 'programming', 'operating system', and 'concurrency control', respectively. The remaining classes received values between 0.4 and 0.6, implying that the user was uncertain about the relevance of them.

The average relevance received by the system with and without learning are shown in Figure 3. While the average relevance without learning saturated at about 0.38, the average relevance with learning increased to about 0.55 within 50 iterations and continued to improve. The final value after 240 iterations was roughly 0.64. A possible explanation for this behavior is that the system quickly learned to discriminate between the classes for which the user entered widely separated values, *e.g.*, 0.9 and 0.1. It needed more time to decide about the relative ranking of the more uncertain categories.

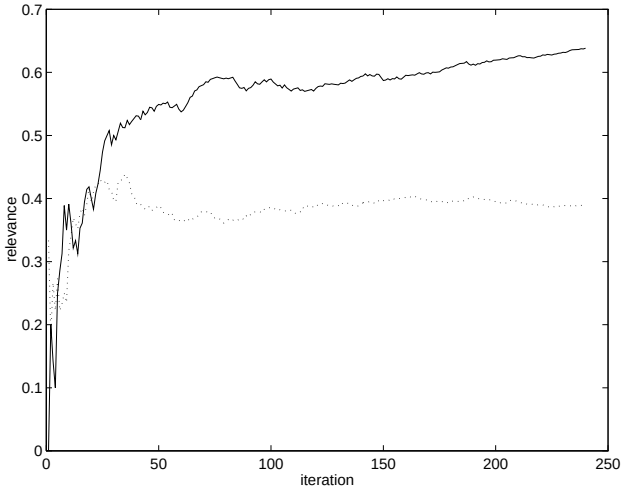

Figure 3: The average relevance received by the e-mail filtering system for an average user with and without learning; solid line: with learning, dotted line: without learning

Conclusions and Future Work

In this paper, we describe a multilevel information filtering system known as SIFTER. Uncertainties arising due to changes in the document stream are handled by the document representation module and the classification module. The document representation module is responsible for limiting the scope of filtering to specific domains. For this it uses an online thesaurus. The classification module helps to reduce the document space size, without loss of vital meaning. Uncertainties in the user are handled by the user profile learning module which uses a reinforcement learning method to automatically determine user interests with minimum user intervention. We have applied this model to filtering Internet listserv messages. The experiments conducted demonstrate very satisfactory results, and the system is indeed able to learn and adapt its filtering action to achieve the goal of delivering relevant information to the user with a high-level of performance.

One direction for our future work involves the automatic adaptation of the document representation scheme, adaptation of the classification tree (*e.g.*, splitting of large classes into smaller subclasses), rapid detection of changes in user interests, and use of 'common sense' knowledge to improve the speed of learning. A second direction will investigate the effect of collaboration between multiple users' filtering system to achieve efficient information sharing in a distributed information environment such as the Internet.

References

Belkin, N. J.; and Croft, W. B. 1992. Information Filtering and Information Retrieval: Two Sides of the Same Coin. *Communications of the ACM* 35(12): 29–38.

Berners-Lee, T.; Cailliau, R.; Luotonen, A.; Nielsen, H. F.; and Secret, A. 1994. The World-Wide Web. *Communications of the ACM* 37(8): 76–82.

Maes, P. 1994. Agents that Reduce Work and Information Overload. *Communications of the ACM* 37(7): 31–40.

Malone, T. W.; Grant, K. R,; Turbak, F. A.; Brobst,S. A.; and Cohen, M. D. 1987. Intelligent Information Sharing Systems. *Communications of the ACM* 30(5): 390–402.

Metral, M. E. 1993. Design of a Generic Learning Interface Agent. B. S. Thesis, EECS dept., MIT.

Mukhopadhyay, S.; and Thathachar, M. A. L. 1989. Associative Learning of Boolean Functions. *IEEE Transactions on Systems, Man, and Cybernetics* 19(5): 1008–1015.

Narendra, K. S.; and Thathachar, M. A. L. 1989. *Learning Automata – An Introduction*. Englewood Cliffs, New Jersey: Prentice-Hall.

Salton, G.; and McGill, M. J. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill.

Selker, T. 1994. Coach: A Teaching Agent that Learns. *Communications of the ACM* 37(7): 92–99.

Seth, B. D. 1994. A learning Approach to Personalized Information Filtering. M. S. Thesis, EECS dept. MIT.

Special issue on Information Filtering. 1992. *Communications of the ACM* 35(12).

Special issue on Intelligent Agents. 1994. *Communications of the ACM* 37(7).

Tou, J. T.; and Gonzalez, R. C. 1974. *Pattern Recognition Principles.* Addision-Wesley.

Acknowledgment

The authors wish to acknowledge Dr. T. Blekher and Dr. A. Olson for many constructive comments during the course of the project.