Modeling User Interest Shift Using a Bayesian Approach

Wai Lam
Department of Systems Engineering & Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong. E-mail: wlam@se.cuhk.edu.hk

Javed Mostafa
School of Library and Information Science, Indiana University, Bloomington, & Department of Computer and Information Science, Purdue School of Science, Indianapolis, IN. E-mail: jm@indiana.edu

We investigate the modeling of changes in user interest in information filtering systems. A new technique for tracking user interest shifts based on a Bayesian approach is developed. The interest tracker is integrated into a profile learning module of a filtering system. We present an analytical study to establish the rate of convergence for the profile learning with and without the user interest tracking component. We examine the relationship among degree of shift, cost of detection error, and time needed for detection. To study the effect of different patterns of interest shift on system performance we also conducted several filtering experiments. Generally, the findings show that the Bayesian approach is a feasible and effective technique for modeling user interest shift.

Introduction

A growing number of people have come to depend on online information sources for their professional careers, news, shopping, and even entertainment. The popularity of online sources has been sustained by drastic decreases in computer hardware and software costs, as well as convenient access to the World Wide Web. With the rapid increase in number of sources and volume of on-line information a critical demand now exists for personalized information delivery (Foltz & Dumais, 1992; Maes, 1995). Filtering systems attempt to fulfill the personalization demand by refining the information flow based on long-term and stable information needs of users (Belkin & Croft, 1992). The interest of users are represented in filtering systems as internal structures known as interest profiles.

Maintaining fidelity in the interest profile in relation to the actual needs of the user is a key problem in filtering systems. This problem is made challenging due to the following complex constraints of the environment:

1. After exposure to different types of information, a user's interest may expand to include new areas or contract to become more specific. Depending on the user, these changes may happen rapidly or take place gradually.
2. A user's interest in the content-domain is directly influenced by his or her situation in the world. With drastic changes in situations, existing interest may degrade or new interest may develop.
3. The domain upon which filtering is performed may also change in unpredictable ways, depending on new discoveries, findings, or inventions. Such changes may induce the user to shift his or her interest.

The notion of long-term and stable information need, if taken too literally, can make the system excessively "brittle" in relation to the above changes. This, in turn, can cause diminished system performance. A user's interest, therefore, requires ongoing tracking, and to maintain compatibility with the actual interest the profile may require occasional revision. A key question then is: how should such revisions occur? In our previous work, we have developed a document filtering system for profile learning (Mostafa, Mukhopadhyay, Lam, & Palakal, 1997). In this article we present a user interest tracking model on top of our previous profile learning approach. It employs a Bayesian method for analyzing relevance feedback from users. The Bayesian tracker is responsible for detecting interest changes and revising interest profiles accordingly. We conduct analysis to clarify the role of the Bayesian approach in the overall filtering process and explain the range of its behavior. In evaluating the utility of this approach, another important question arises: how do different patterns of interest shift (degree and scope) influence system performance? We take our analysis further and conduct several filtering experiments in an attempt to answer this question as well.

Received November 25, 1998; revised April 6, 2000; accepted July 10, 2000.

 ● Published online 1 February 2001

In the next subsection a brief survey of related literature is presented. In the Document Filtering System section, an overview of the filtering process, as modeled and implemented in this research, is presented. Interest profile learning, tracking, and revision are covered in the User Profile Learning section. An analytical study is presented in the Analytical Study section to identify the cost associated with reconvergence to an optimal profile with or without the tracking scheme. The Simulation Studies section presents results of experiments to identify the relationship among degree of shifts, cost and detection time. In the Filtering Experiments subsection, the practical utility of the interest tracking system is evaluated based on a series of filtering experiments. The article ends with a discussion of the major findings and prospective future research directions.

Survey of Related Research

Generally, the literature covering significant aspects of filtering is quite large and diverse. Here, we highlight a few of the major techniques and research findings that are closely related to the main theme of the article.

After users are presented with a retrieved set of documents, certain systems permit revision of the original query based on user's identification of particular documents as relevant. This technique is called query refinement by relevance feedback (Salton & McGill, 1983). Goker and McCluskey (1991) have extended the technique to generate a more durable representation, called "user-context," based on terms from all relevance feedback episodes that may occur in different sessions. The user-context is similar to a simple interest profile, containing stemmed terms, frequency, and weight. Expert evaluation of user-contexts and comparison with an alternative method for query refinement showed that the context approach holds promise in improving document retrieval. In a research conducted by Foltz and Dumais (1992), profiles were created using key words directly gathered from users and indirectly from highly rated documents (relevance feedback). Then, to conduct filtering, profiles were compared with documents in two ways: match between comprehensive weighted term vectors, and match between reduced dimension vectors based on latent semantic indexing (LSI). It was found that the profiles containing keywords gathered through relevance feedback and with dimensionality reduced using LSI produced the best results.

The annual TREC (Text REtrieval Conference) initiative, sponsored by the National Institute for Standards and Technology (NIST), attempts to gather consistent and more generalizable evidence of retrieval performance based on a standard document collection. Because TREC-4, a new track known as the filtering track has been introduced (Hull & Robertson, 1999). A research group can participate in three subtasks under the filtering track: adaptive, batch, and routing. The user profiles in all the subtasks are created based on information need descriptions called topics. The TREC documents have been preassessed to establish their relevance in relation to each topic. Given a subset of the collection, called the training document set,[1] in the batch and the routing tasks the profile is usually generated off-line based on a suitable learning procedure. A second document subset, called the testing set, is used for evaluation. In the batch evaluation task the profiles are applied to arrive at a binary choice, relevant or nonrelevant, for each test document, and in the routing task the profiles are used to rank test documents according to their relative relevance. The adaptive task differs from the other two tasks in that no training documents are provided. The adaptive system is supposed to start with initial test topics as profiles and determine for each incoming document from the test set if the document is relevant or nonrelevant. If the document is predicted as a relevant one, the system can receive the preassessed true relevance judgment for that document. The true relevance judgment can then be used to adaptively fine tune the profile. A utility based measure known as linear utility function (LF) is the metric used for measuring the performance. LF assigns a cost or value to each document measured in terms of the following factors: (1) relevant or nonrelevant, and (2) retrieved or not retrieved. To normalize scores across topics a scaling function is applied on each topic's LF score before averaging (Hull & Robertson, 1999). In the recent TREC-8, 14 research groups participated in the filtering track. In the adaptive filtering task, no group performed better than the baseline, but the University of Iowa research group, using a dynamic clustering approach, achieved the best LF1 result (threshold for retrieval is 0.4). Based on the more liberal LF2 measure (threshold for retrieval is 0.25), five groups performed above baseline, and the University of Twente, using a probabilistic retrieval model, produced the best result.[2]

Pazzani and Billsus (1997) described a method of profile acquisition that is based on user's identification of Web documents as "hot" or "cold." Using a set of 128 features (words) extracted from a training sample, the profiles were implemented as classifiers that could establish for new documents their membership in the "hot" or "cold" class. A comparison of several different classifiers, including the ID3 decision-tree, nearest neighbor, and neural networks, showed that two particular classifiers, the Bayesian and the Rocchio relevance feedback methods, outperformed the others. In their research, the feature weights used in the Bayesian method were binary values (0 or 1), whereas the Rocchio method used term-frequencies calibrated by the *tf.idf* approach (Salton & McGill, 1983). Jennings and Higuchi (1992) developed the profile based exclusively on a multilayer neural network method. In the neural network each node represented a particular word and the node weight was determined based on documents read or re-

[1] Relevance judgment associated with each document in the training set can be used by the system to generate the initial profile.

[2] Here, the adaptive filtering results are only reported, because this task is most similar to the filtering method applied in this article. A comprehensive treatment of the results from the TREC-8 filtering track can be found in Hull and Robertson (1999).

jected. Nodes were linked to each other according to word cooccurrence found in the documents read. The application domain was Usenet News, and news documents were ranked using the sum of weights of active nodes after a matching process. Jennings and Higuchi (1992) argued that the nonlinear approach to profile representation offered a more nuanced and accurate way to capture user interest. The evaluation results of document presentation over a 2-week period showed that the system can successfully adapt to certain interests, although there were significant initial delays in acquiring useful profiles. Profile acquisition is difficult when user participation cannot be guaranteed or it is perceived by users as burdensome. In such cases, other sources of data must be relied upon to acquire the profiles. Certain researchers have observed that in large networked environments (such as the WWW) it is easy to identify multiple users or user groups that share the same interest. Hence, by correlating a minimal amount of user interest with the interest of other users, a more comprehensive or expansive profile can be generated. This means of profile acquisition is applied by a particular type of systems known as collaborative filters. In pure collaborative filters, there is little or no content analysis conducted (e.g., no text representation) by the computer (Balabanovic & Shoham, 1997); the idea is to collect rating data based on user input and aggregate the rating data to identify clusters of items similarly rated by individuals. The independence from content, allows the pure collaborative filter to deal with diverse contents, such as image, video, music, etc. Balabanovic and Shoham (1997), however, contend that a mixed mode of filtering, one that combines collaborative rating with ongoing relevance feedback collected directly from users is necessary for overcoming delays in delivering new information due to rating latency from group members.

Over a period of time a domain may go through radical changes due to a wide variety of factors, ultimately affecting the content of the documents in the stream. In the literature, the general phenomenon of changes in the content of information stream flowing into a filter or a classifier is referred to as a concept drift. Widmar and Kubat (1996) studied concept drift based on several different variants of their FLORA (FLOating Rough Approximation) system. In FLORA, the profile is represented as three sets of concept descriptors, with each descriptor represented as logical conjunction of attribute/value pairs. The ADES (Accepted DEscriptors) contains positive concepts, the NDES contains negative concepts, and the PDES contains potentially positive concepts that are overly general and some negative concepts. In the training mode, FLORA maintains a window of size *n* over the incoming labeled examples that it uses to update the content of the three descriptor sets (the profile). Widmar and Kubat presented results of several experiments where they systematically varied the capability of the original FLORA system. With FLORA2 they showed that dynamic adjustment of the window size is superior than a fixed window, and with FLORA3, they demonstrated that maintaining past concept descriptors and reusing them during concept drifts involving recurring concepts may be effective. A final version called FLORA4 that had the features of earlier versions and in addition monitored predictive performance of individual descriptors in the profile was also evaluated, and it was found that it was capable of distinguishing between occasional noise in the incoming stream and actual concept drifts. Klikenberg and Renz (1998) also investigated the phenomenon of concept drift. In their work, they compared influence of fixed window size with dynamic window size on eight different classification algorithms. The classifiers analyzed were: Rocchio, Naive Bayesian, PrTFIDF, K-nearest neighbor, Winnow, Support Vector Machine, CN2, and C4.5. Instead of determining relevance of documents one at a time, their classifiers established relevance of documents for 20 batches, with each batch containing 130 documents. Three measures were used, accuracy, precision, and recall, and they found on all the measures, the adaptive-window adjustment approach achieved slightly higher average performance over fixed-window approach. Klinkenberg and Renz also investigated sudden change in interest (in contrast to gradual)—a phenomenon they referred to as concept shift. For all classifiers, the average performance of all measures were again slightly higher for adaptive-window approach compared to fixed-window approach. A closer analysis of both concept shift and drift, with the CN2 algorithm, established that the adaptive-window approach can quickly detect changes in the information stream, and it generally provides more stable performance than the other approaches.

A related problem area concerned with identification of topical changes in a continuous stream of information is called Topic Detection and Tracking (TDT). Research on TDT concentrates on the domain of broadcast news that may be communicated in multiple languages and distributed in various formats (Fiscus, Doddington, Garofolo, & Martin, 1999). A varied set of problems are studied in TDT, but two important ones are topic detection and tracking. In topic detection the goal is to arrive at a binary decision on a story, new or old, in a continuous stream of stories. Topic tracking involves detecting all related stories to a particular topic in a continuous stream of stories. The performance is characterized in terms of the probability of miss and false alarm errors. These error probabilities are then combined into a single detection cost by assigning costs to miss and false alarm errors. Carbonell, Yang, Lafferty, Brown, Pierce, & Liu (1999) employed a single-pass incremental clustering approach for the detection task and k-NN approach for the tracking task. Papka, Allan, and Lavrenko (1999) investigated both topic detection and tracking. In their topic detection approach, they treated stories as queries. Queries were represented using a variant of the *tf.idf* formula (Salton & McGill, 1983), combined with relevance feedback as implemented in the Inquery system. They provided an algorithm for maintaining queries in memory, matching queries for detecting new topics, and revising the query memory according to the detection results. They also incorporated temporal information about stories in the similarity

evaluation measure. To conduct topic tracking, they used *tf.idf* for story representation. The task involved acquiring a query from few initial stories and matching the query to subsequent stories to determine their similarity to the internal story representation. It was found that as few as four initial training stories can provide relatively good performance on detecting subsequent related stories.

To sum up, we can state that certain interesting and useful findings were uncovered among the techniques reviewed here. Relevance feedback seems to be a promising technique in acquiring profiles, applied by a wide variety of systems. Reducing system latency in acquiring the profile is an important target for filtering systems. The research evidence also showed that performance attained by computationally intensive batch-oriented learning methods and nonlinear methods can be matched by simpler techniques. In this research, our approach integrates reinforcement learning based on relevance feedback with a Bayesian interest tracker to conduct profile acquisition. An objective we treat seriously is the minimization of system latency in acquiring the profile. In many contexts, the user expects on-demand and immediate service and, therefore, delay in acquiring the profile may seriously hinder filtering utility. This constraint would rule out certain batch-oriented techniques, as they require a large amount of training samples and preestablished relevance judgment. In our system, both the reinforcement learner and the Bayesian tracker utilize the same relevance feedback data, and they are designed to function concurrently with the provision of real-time filtering service. Another area we pick up on and emphasize is systematic analysis of user interest shifts. Although, previous studies investigated concept and topic changes in the information stream, we consider these only partially addressing the issue of how changes in the filtering environment can affect the profile. The user profile can be affected through self initiative, interest, or concerns of the user, independent of other changes in the environment. It is a problem area for which we found little prior formal analysis. We believe user's interest shifts should be considered more explicitly and directly in the profile acquisition process. We are interested in pursuing how such shifts can be detected and the profile accordingly adjusted. In this work we present several analytical and simulation studies that consider the influence of such shifts on the overall behavior of the system.

A Document Filtering System

The ultimate objective in filtering is to establish accurate relevance assessment for each document in the collection. Depending on the choice of presentation mode selected by the user, documents can be pruned, grouped, or ranked based on their relevance assessments. We have previously developed a textual document filtering system in an ongoing project SIFTER (Mostafa et al., 1997; Mukhopadhyay, Mostofa, Palakal, Lam, Xue, & Hudli, 1996). It is assumed that a user regularly receives documents (e.g., via e-mail or Web-robots) that are placed in a predesignated document

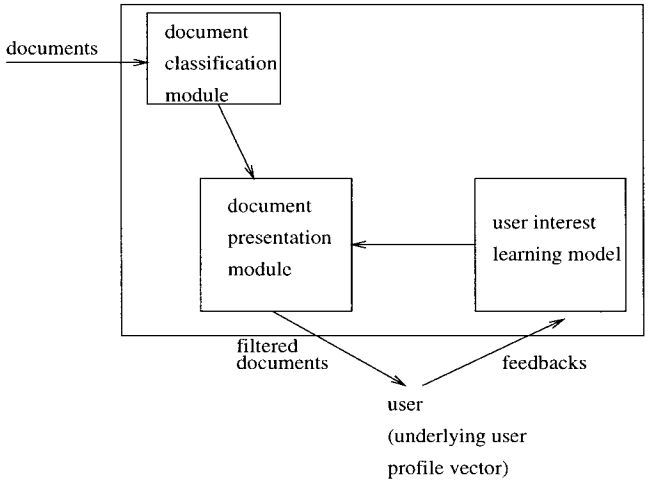

FIG. 1. Our document filtering system.

"bin." When the filtering system is invoked, its task is to compare the internal profile with the contents of the bin, and present to the user a ranked list of the new documents. We refer to the system as passive, because the system does not autonomously or actively seek out documents from external sources, it merely sorts documents placed in its bin. The internal profile does not contain information about individual documents. Rather, it contains information about a fixed set of topical areas we call classes. More specifically, the profile constitutes a user's interest in a set of classes.

Upon execution of the system, the general process flow involves: checking the bin for new documents, classifying the documents to their classes, sorting the documents according to the class-interest information in the profile, presenting the documents, and finally collecting relevance feedback on documents. Initially, the system has no information about a user's interest. The profile is learned based on the relevance feedback generated for documents, which in turn, is treated as the relevance feedback for the corresponding classes. Instead of learning the interest directly over documents or document components (e.g., words or phrases), the interest is learned over classes due to the following main reasons. In our view, classes are more stable semantic units than individual words or phrases appearing in documents. We used classes from the Medical Subject Headings[3] list. These classes, therefore, were authoritative representations of subtopics in the domain of our concentration. Another motivating factor for using classes was simplification of profile management. The concise and stable nature of the classes meant the profiles could be limited to a particular size, requiring only occasional changes (although our system does permit revision of classes if necessary). As depicted in Figure 1, the filtering system is mod-

[3] Medical Subject Headings (MeSH) is a classification scheme produced by the National Library of Medicine (NLM) in the United States. We selected a small subset of headings concerning the cell biology area. The headings used in this research are shown in Table 2.

eled as a multilevel process supported by three basic components: (1) document classification, (2) user interest learning, and (3) document presentation. Here, we present a brief overview of these components.

The first module in the system performs the classification task. The input to this process are documents and the output are class labels for each document. We applied a fairly straightforward procedure to achieve this. In this research, the classification is accomplished by calculating the similarity of a document to each class. This is conducted by using the distance measure below that is based on the cosine similarity formula as proposed by Salton and McGill (1983).

$$1 - \sum_{i=1}^{t} v_i z_i / \sqrt{\left(\sum_{i=1}^{t} v_i^2\right)\left(\sum_{i=1}^{t} z_i^2\right)} \quad (1)$$

The computation above involves calculating the similarity between two vectors, where the vector v_i is the document, and the vector z_i is a class. The two vectors were based on t elements, where each element represented a term (a single-word). There were 43 elements in each vector and they were directly determined based on terms found in the 29 Medical Subject Headings selected for this research. Generating the vector representation of documents involved establishing for each document the frequency of the 43 relevant terms (we used a simple word-dictionary for this). Each term-frequency (tf) in the document vector was then multiplied with an inverse-document frequency (idf) calibration measure (Salton & McGill, 1983). The idf calibration measure for all terms in the dictionary were preestablished based on a subset of the document corpus used in this research. The calibration step is useful for "normalizing" the effect of differences in number of documents processed from session to session. For each class a corresponding vector was created a priori, so that in the class vector representation the relevant elements had a value of 1 and the rest of the elements were set to 0.[4]

After the distance values were calculated for a document by matching it with all the classes, the class producing the smallest distance was selected as the target class. The selected class label was then associated with the document record for profile matching purposes.

After identifying the classes for all documents in a session, the subsequent operation involves relevance assessment of each document. This operation is conducted as part of the user modeling component. It required modeling the distribution of user interest over the set of fixed classes. To generate and update this model, a particular scheme called reinforcement learning (Narendra & Thathachar, 1989) was adopted. In implementing this scheme, an estimated relevance assessment vector and a class selection vector (both of size equal to the number of classes, with each element representing a class) were maintained and updated. Relevance feedback on documents was applied to continually update the internal user model. The relevance feedback was also used to detect abrupt changes in interest. A tracking component was developed to perform Bayesian analysis on the pattern of feedback to detect shifts. One assumption of our user modeling is that the number of classes is known in advance. Nevertheless, this number is only required for the internal user modeling and the user does not need to know this information in the whole filtering process. More details on the two significant user modeling subcomponents: profile learning and shift-detection are presented in the next section.

The presentation module is responsible for facilities that allow the user to view the filtered documents in ranked, grouped, or pruned form (ranked mode was selected for this article). This component also permits an experimenter to directly control how relevance feedback is generated. Our implementation supported two major options: an autonomous mode (used in this article) and an interactive mode. The autonomous mode was designed to provide filtering service with minimal user intervention. In the autonomous mode, the experimenter can enter an external static profile (Θ) specifying a particular value 0–1.0 (inclusive) for each class, which thereafter is used to automatically deduce the relevance feedback. This is conducted probabilistically so that documents belonging to classes with high interest values (near 1.0) in Θ would have a higher likelihood to receive positive feedback than documents in classes with low interest values (close to 0). In this scheme, the internal user modeling is still supported, as it was designed to be adaptive to changing filtering demands.

User Profile Learning and Tracking

The User Interest Learning Model

The user interest learning model is developed based on a reinforcement scheme known as the L_{R-I} model. In this scheme, the system is conceived as a learning automaton that is in a feedback configuration with the user (Narendra & Thathachar, 1989). The system assumes a set of m predefined classes for documents. A "true" user interest or relevance is modeled as a vector $\Theta(t) = (\theta_1(t), \ldots, \theta_m(t))$. Each θ_i represents the true degree of relevance of a document in class i. The system models the learning environment in terms of estimated probabilities of relevance and class selection probabilities. The probabilities of relevance, represented as a vector maintained in the system, is the internally estimated user profile: $\hat{\Theta}(t) = (\hat{\theta}_1(t), \ldots, \hat{\theta}_m(t))$. The class selection probabilities are represented as a

[4] The initial set of classes selected is dependent on the scope of the domain coverage desired by the user. In our system, as part of the initialization step, any type or number of classes can be selected (for, e.g., additional MeSH classes from the headings list). The shift detection algorithm does not make any assumption about the initial number or type of classes. However, in our present work, the detection algorithm does assume that the classes remain fixed after system initialization. Please see the Conclusion section for further discussion on this issue.

second vector: $Q(t) = (q_1(t), \ldots, q_m(t))$. In each session, after documents are classified into individual classes, sorting of documents takes place by examining the probability values associated with the classes in both vectors. First, the best class is selected based on the probability values of classes in the class selection vector Q. Then, the other classes are rank-ordered according to their probability values in the estimated user profile vector $\hat{\Theta}$. This ranking of classes is used to determine the order of documents as presented to the user. The relevance feedback provided to a document is a binary value (1: a positive reaction feedback and 0: a negative reaction feedback). The feedback for a document is assumed to be a feedback for the corresponding document class. Unlike other filtering systems, our class-based framework is a simplification for the purpose of developing a model and easy handling of changes in user interests over time.

The learning takes place based solely on the relevance feedback. In the interactive mode this feedback is directly acquired from the user. Typically, a set of documents are presented to the user in every filtering session. The user may provide relevance feedback for certain documents in the session.

In the autonomous mode the feedback is derived probabilistically according to the externally provided profile (Θ). Regardless of the origin, each feedback collected is used to update the internally maintained estimated profile $\hat{\Theta}$. Suppose a feedback r_c is received for the class c. r_c is 0 or 1, denoting a negative or positive feedback, respectively. The element $\hat{\theta}_c$ in this vector is updated in the following manner:

$$\hat{\theta}_c(t+1) = (\hat{\theta}_c(t)t + r_c)/(t+1)$$

In other words, the $\hat{\theta}_c$ is actually the running average of feedback received for the class c. If this feedback is positive, then the class selection vector Q is updated in the following way:

$$q_i(t+1) = q_i(t) + \lambda(1 - q_i(t)) \quad \text{if } i = c,$$

$$= q_i(t) - \lambda q_i(t) \quad \text{if } i \neq c \tag{2}$$

where λ is a suitably chosen learning rate and $0 < \lambda < 1$. If no positive feedback is received, then the vector Q remains unchanged. At the beginning of filtering sessions, the class selection vector is initialized to contain equal probability for all classes, i.e., q_i, $(i = 1, \ldots, n)$ are set to be $1/n$.

Two vectors, $\hat{\Theta}$ and Q, are maintained instead of just using $\hat{\Theta}$ in our filtering model. This model possesses a nice theoretical foundation drawn from the L_{R-I} model in reinforcement learning. It can be guaranteed that the class selection vector Q converges to an unit vector where the most favored class attains the highest value. By giving a chance to documents from all the classes to be displayed at the top, at least a few times during the convergence process, the class selection probability vector permits additional refinement and eventually more accurate reflection of the user interest in the estimated profile $\hat{\Theta}$.

The User Interest Tracking Model

As mentioned above, there is a probability of relevance associated with each class in the estimated user profile. These probabilities of relevance capture the user interest for the corresponding classes. The user interest learning model presented in the previous section (L_{R-I}) applies well if the user interest is stationary. The user interest, however, may shift due to the varied reasons outlined in the introductory section. As the L_{R-I} is conceived and implemented, it has the capacity to relearn the shifted interest profile and recover from an inappropriate converged state. In practice, however, such relearning takes numerous iterations and feedback cycles, and in the meantime, the performance degrades drastically. Therefore, another strategy is needed to detect and recover from shifts quickly. We present here a user interest tracking model designed to support the necessary functions.

The tracking is performed on each class separately. For each class, there exists a history of user relevance feedback data. The tracking scheme employs a Bayesian framework to detect a shift in the probability of relevance based on the feedback data. We first describe a model for which a single shift has occurred at some point of time with a known prior probability distribution of the time of shift. The objective is to detect it quickly. When multiple shifts occur, the time interval between two successive shifts is assumed to be sufficiently large. This allows each shift detection problem to be treated independently, and the time window over which a shift can occur to be idealized as an infinite horizon window.

Let $r_1, r_2, \cdots$ be a sequence of the feedback data given by the user for a particular class. Each r_i is either 1 or 0, governed by the underlying probability of relevance. Hence, r_i can be regarded as an independent Bernoulli random variable where the probability of relevance $r^{(i)}$ is the probability of "success" (i.e., $P(r_i = 1) = r^{(i)}$). Under this framework, this problem can be viewed as detecting a shift in the success probability in a sequence of Bernoulli trials. We propose a Bayesian approach for this problem by computing the posterior probability that a shift has occurred given the feedback data. An upward shift in probability of success from Bernoulli trials has been studied in (Zacks & Barzily, 1981). Our shift detection approach makes extensive use of the results from their work and extends it to handle a downward shift as well.

Suppose a shift occurs at time instance h. Let θ and β be the probability of relevance before and after an occurrence of a shift respectively. We treat h, θ, and β as random variables. Although we model the shift as a sharp one, this technique is equally applicable to a gradual shift.

One choice of the prior distribution for h is a geometric prior distribution $p_s(h)$, which is defined as:

$$p_s(h) = (1 - s_0)s_1(1 - s_1)^{h-1}$$

We define $p_s(0) = s_0$. s_0 and s_1 are the parameters of the geometric distribution and $0 < s_0, s_1 < 1$. This geometric distribution has the attractive property that the prior probability of a shift tails off as $h \to \infty$. It implies that the effect of the prior distribution on the decision rule becomes negligible for large h.

The parameters θ and β have a prior distribution $p_d(\theta, \beta)$ over the simplex $0 < \beta - \theta < 1$ for an upward shift and $0 < \theta - \beta < 1$ for a downward shift. Without any prior knowledge concerning the values of the probabilities of relevance before and after the shift, $p_d(.,.)$ is assumed to be uniform over the domain of support.

Consider a sequence of feedback data $R_n = (r_1, r_2, \ldots, r_n)$, the likelihood function of (h, θ, β) is:

$$L(h, \theta, \beta|R_n) = \begin{cases} \theta^{T_h}(1-\theta)^{h-T_h}\beta^{T_n-T_h}(1-\beta)^{n-h-(T_n-T_h)} & \text{if } h < n \\ \theta^{T_n}(1-\theta)^{n-T_n} & \text{if } h \geq n \end{cases} \quad (3)$$

where $T_j = \sum_{i=1}^{j} \beta_i$ and $T_0 = 0$. Our objective is to compute the posterior probability of "a shift has occurred" given the sequence of feedback data R_n. Denoting such probability as $S(R_n)$ (i.e., $S(R_n) \equiv P((h < n)|R_n)$), we let $S^{(u)}(R_n)$ and $S^{(d)}(R_n)$ be the posterior probability of an occurrence of an upward and a downward shift respectively given R_n. Using Bayes theorem, we can calculate $S^{(u)}(R_n)$ as follows:

$$1 - \left[\left(\sum_{j=n}^{\infty} p_s(j) \right) \int_0^1 \int_\beta^1 L(n, \theta, \beta|R_n) p_d(\theta, \beta)\, d\theta\, d\beta \right] / \mathcal{N} \quad (4)$$

where $\mathcal{N}$ is a normalization factor given by:

$$\mathcal{N} = \sum_{j=0}^{\infty} \left[p_s(j) \int_0^1 \int_\beta^1 L(j, \theta, \beta|R_n) p_d(\theta, \beta)\, d\theta\, d\beta \right] \quad (5)$$

The details of the derivation are given in the Appendix.

As proved in Zacks and Barzily (1981), the sequence $S^{(u)}(R_n)$ is a submartingale when the prior distribution $p_s(h)$ is assumed to be geometric. This implies that as $n \to \infty$, the posterior probability of the shift approaches 1, and the shift will be eventually detected. The rate of convergence analysis presented in the Rate of Convergence Analysis section also assumes that a single shift has occurred, and that it is correctly detected (i.e., n, the window of observation is sufficiently large).

For each class c, we compute $S_c^{(u)}(R_n)$ and $S_c^{(d)}(R_n)$ representing the posterior probability of an upward and a downward shift, respectively, given the sequence of feedback data R_n. In practice, because of the finite choice of n, there will always be a nonzero probability of making a wrong detection. This leads to the use of decision functions with associated costs in practice as explained in the following section.

Integrating the Tracking Model with the Learning Model

We introduce two cost quantities, namely the cost of ignoring a shift (missed detection) and the cost of declaring an unnecessary shift (false alarm). A decision function is then formulated based on the shift probability and these two cost quantities. This decision function is used for determining if the user interest learning model needs to be informed so that appropriate reinitialization can take place.

The posterior probability (i.e., $S_c^{(u)}(R_n)$ or $S_c^{(d)}(R_n)$) is the dominant parameter for this decision function. Basically, if $S_c^{(u)}(R_n)$ is high, it is likely that we should declare an upward shift for the class c. Similarly, if $S_c^{(d)}(R_n)$ is high, it is likely that we should declare a downward shift for the class c. Let k_1 be the cost of ignoring a shift that has already occurred, and k_2 be the cost of declaring an unnecessary shift. Then, we define a cost ratio k computed as k_2/k_1, which can be interpreted as the relative cost of declaring a shift. Two decision functions, namely $F_c^{(u)}(R_n)$ and $F_c^{(d)}(R_n)$ representing the decision of declaring an upward and a downward shift respectively for each class c are defined as:

$$F_c^{(u)}(R_n) = [S_c^{(u)}(R_n)]^k$$

$$F_c^{(d)}(R_n) = [S_c^{(d)}(R_n)]^k \quad (6)$$

If $k > 1$, the decision function F_c grows in a slower fashion than S_c. On the other hand, if $k < 1$, F_c grows in a faster fashion than S_c. As k gets larger, the model becomes cautious about declaring shifts. As k gets smaller, the model becomes very sensitive to potential shifts. The choice of k can be determined by the desired detection sensitivity in an application. Finally, we set a threshold $F_{\text{threshold}}$. If the decision function F_i exceeds $F_{\text{threshold}}$, then the user interest learning model examines the kind of shift and adjusts the class selection vector accordingly.

When an upward shift for a class c is declared, it will further check whether c is the most probable class (i.e., q_c currently attains the highest class selection probability). If c is the most probable class, no adjustment for the class selection vector is needed. Otherwise, it looks for the most probable class l and distributes the sum of q_c and q_l equally among these two classes as follows:

$$q_c(t) = q_l(t) = (q_c(t) + q_l(t))/2$$

Suppose a downward shift for a class c is declared. If c is not the most probable class, no adjustment is needed. Otherwise, the user interest learning model will react as follows: first, we look for the next best class. When the learning model has converged to the optimal class c, the class selection probabilities of all other classes will be close to 0. Thus, the class selection probabilities are not a good choice for finding the next best class. Instead of using these probabilities, we use the estimate of the user profile vector $\hat{\Theta}$. Specifically, let the class, excluding c, with the highest estimate of the probability of relevance be e (i.e., $\hat{\theta}_e = \max_{j \neq c}\{\hat{\theta}_j\}$). It distributes the sum of q_c and q_e equally among these two classes. It can be formally stated as follows:

$$q_c(t) = q_e(t) = (q_c(t) + q_e(t))/2$$

Analytical Study

Rate of Convergence Analysis

Under our proposed filtering framework, there exists a stable filtering performance for a particular user profile. We investigate convergence analysis, which is essentially a study of how fast the system achieves the stable filtering behavior. Consider a scenario when a single shift has occurred in the underlying user profile. We assume that the shift is always correctly detected.

Let $V(t)$ be the average relevance for a given class selection vector defined as:

$$\sum_{i=1}^{m} \theta_i q_i(t) \tag{7}$$

Let $E[V(t)]$ be the expectation of $V(t)$ and θ_l be the optimal probability of relevance for the user (i.e., $\theta_l = \max_i(\theta_i)$). A measure of the instantaneous rate of convergence at feedback cycle t is given by:

$$\hat{\rho}(t) = \frac{\theta_l - E[V(t+1)]}{\theta_l - E[V(t)]} \tag{8}$$

$\hat{\rho}(t)$ represents how close $E[V(t+1)]$ is to $E[C^*]$ in comparison with $E[V(t)]$.

Because the user interest learning model is based on a L_{R-I} scheme, which is an absolutely expedient scheme, $E[V(t)]$ is a monotonically increasing function of t (i.e., $E[V(t+1)] > E[V(t)]$) (Narendra & Thathachar, 1989). Therefore, $\hat{\rho}(t) \leq 1$ for all t. Equation (8) can be expressed as follows:

$$\hat{\rho}(t) = 1 - \frac{E[V(t+1) - V(t)]}{\theta_l - E[V(t)]} \tag{9}$$

$E[V(t+1) - V(t)]$ is the expected increase in the average relevance at the time instance t. Now, we consider a learning scheme described in Equation (2). The expected increase in the average relevance is given by:

$$E[V(t+1) - V(t)] = \frac{\lambda}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} (\theta_i - \theta_j)^2 E[q_i(t) q_j(t)] \tag{10}$$

$E[V(t)]$ is given by:

$$E[V(t)] = \sum_{i=1}^{m} \theta_i E[q_i(t)] \tag{11}$$

The expressions in Equation (10) and Equation (11) involve $E[q_i(t)]$, which cannot be computed because the distribution of $Q(t)$ is unknown. Nevertheless, we can make use of the value at time t to serve as an estimate of the expectation value (i.e., we use $q_i(t)$ for an estimate of $E[q_i(t)]$). As a result, $\rho(t)$ can be expressed as:

$$\rho(t) = 1 - \frac{\lambda/2 \sum_{i=1}^{m} \sum_{j=1}^{m} (\theta_i - \theta_j)^2 q_i(t) q_j(t)}{\theta_l - \sum_{i=1}^{m} \theta_i q_i(t)} \tag{12}$$

Equation (12) plays a central role in analyzing the rate of convergence of the learning model incorporating the user interest tracking scheme. As indicated in this equation, the rate of convergence depends on both the current class selection vector Q and the underlying user profile vector Θ.

Convergence Analysis Without Shift Detection

Consider the situation where the learning model has converged to the optimal class l. We expect q_l to be very close to 1, whereas all other probability of relevance are close to 0. Now suppose there is a change in the user profile. Specifically, the new probability of relevance of l drops to a rather low value. The class with the next highest probability of relevance in the old user profile becomes the new optimal class l'. Let $\Theta' = \{\theta'_1, \theta'_2, \ldots, \theta'_r\}$ be the probability of relevance vector of the new profile. Hence, $\theta_i = \theta'_i \; \forall i \neq l$. Let t_c be the time instance where this profile change takes place. Using Equation (12), we can calculate $\rho(t_c)$ the rate of convergence at t_c as follows:

$$\rho(t_c) = 1 - \frac{\lambda/2 \sum_{i=1}^{m} \sum_{j=1}^{m} (\theta'_i - \theta'_j)^2 q_i(t_c) q_j(t_c)}{\theta'_{l'} - \sum_{i=1}^{m} \theta'_i q_i(t_c)} \tag{13}$$

Also, we calculate $\rho(\infty)$, the rate of convergence when the learning model converges to the optimal class as follows:

$$\rho(\infty) = 1 - \frac{\lambda \sum_{i=1}^{m} (\theta'_{l'} - \theta'_i)^2}{\sum_{i=1}^{m} (\theta'_{l'} - \theta'_i)}$$

If no user interest tracking scheme is used, $\rho(t_c)$ and $\rho(\infty)$ give the bounds of the convergence rate. To simplify the analysis, we can develop the average rate of convergence ρ_{nodetect} using the bounds. For instance, ρ_{nodetect} can be calculated using the weighted sum like $(w_1\rho(t_c) + w_2\rho(\infty))/(w_1 + w_2)$. Let b_{nodetect} be the relevance distance between the current average relevance and the optimal one at time t_c. b_{nodetect} can be calculated by:

$$b_{\text{nodetect}} = \theta'_{l'} - \sum_{i=1}^{m} \theta'_i q_i(t_c) \tag{14}$$

Let b_{desired} be the desired relevance distance that we want to achieve. It can be shown that the time t_{nodetect} (number of feedback received) needed for b_{nodetect} to decrease to b_{desired} is:

$$t_{\text{nodetect}} = \log\left(\frac{b_{\text{desired}}}{b_{\text{nodetect}}}\right)/\log \rho_{\text{nodetect}} \tag{15}$$

Note that both the numerator and denominator in Equation (15) are negative.

Convergence Analysis With Shift Detection

We compare the time needed if the learning model incorporates the user interest tracking scheme. Let t_d be the time instance (number of feedbacks received) after t_c for which the appropriate shift is detected and declared so that the learning model reinitializes its class selection probability vector to $Q' = \{q'_1, q'_2, \ldots, q'_m\}$. Let b_{detect} be the relevance distance between the average relevance at time $t_c + t_d$ and the optimal one. b_{detect} can be calculated by:

$$b_{\text{detect}} = \theta'_{l'} - \sum_{i=1}^{m} \theta'_i q'_i \tag{16}$$

Similarly, the average rate of convergence ρ_{detect} is given by $(w_1\rho(t_c + t_d) + w_2\rho(\infty))/(w_1 + w_2)$. The rate of convergence at this time instance $t_c + t_d$ is:

$$\rho(t_c + t_d) = 1 - \frac{\lambda/2 \sum_{i=1}^{m} \sum_{j=1}^{m} (\theta'_i - \theta'_j)^2 q'_i q'_j}{\theta'_{l'} - \sum_{i=1}^{m} \theta'_i q'_i} \tag{17}$$

Hence, the time t_{detect} needed to reach the same performance level b_{desired} is:

$$t_{\text{detect}} = t_d + \log\left(\frac{b_{\text{desired}}}{b_{\text{detect}}}\right)/\log \rho_{\text{detect}} \tag{18}$$

As a result, the gain in the convergence time γ can be characterized using Equation (15) and Equation (18) as $t_{\text{nodetect}}/t_{\text{detect}}$. If $\gamma > 1$, the learning model incorporated with the user interest tracking scheme is superior to the one without the tracking algorithm. If $\gamma < 1$, the user interest tracking algorithm penalizes the performance of the learning model.

To analyze the gain γ, consider Equation (14). Because q_l is close to 1, whereas all other probabilities of relevance are very close to 0, b_{nodetect} is approximately equal to $\theta'_{l'} - \theta'_l$. On the other hand, from Equation (16), b_{detect} is roughly equal to $\theta'_{l'} - 0.5(\theta'_l + \theta'_{l'}) = 0.5(\theta'_{l'} - \theta'_l)$ (due to the reinitializing scheme discussed in the User Interest Tracking Model section). Obviously, we have $b_{\text{detect}} < b_{\text{nodetect}}$. Moreover, the larger is the value of b_{nodetect}, the larger the difference between b_{nodetect} and b_{detect}. Due to the same reason, $\rho(t_c)$ in Equation (13) is closer to 1 compared with $\rho(t_c + t_d)$ in Equation (17). Therefore, t_{nodetect} in Equation (15) will be larger than the expression on the right side of the sum in Equation (18).

The remaining component for t_{detect} to be analyzed in Equation (18) is t_d (i.e., the time required to declare a shift occurrence). We are interested in calculating the expected value of t_d (i.e., $E[t_d]$). Following the notation used in the User Interest Tracking Model section, we consider the situation where a downward shift in probability of relevance from θ to β has occurred at time h. Let $\mathcal{R}$ denote the set containing R_n which enables the shift declaration at the exact time t_d after the shift occurrence for a class c. Precisely, the following conditions hold for $\mathcal{R}$:

$$\forall R_n \in \mathcal{R}[\ \forall n_0 < n;\ F_c^{(d)}(R_{n_0}) \le F_{\text{threshold}} \text{ and } F_c^{(d)}(R_n) > F_{\text{threshold}}]$$

where R_{n_0} denotes the partial sequence from the first feedback up to the n_0th feedback in R_n. $F_c^{(d)}$ is the decision function described in Equation (7). The probability of shift declaration occurred at t_d after shift occurrence is given by:

$$\sum_{R_n \in \mathcal{R}} L(h, \theta, \beta | R_n)$$

TABLE 1. The shift detection time of different degree of shifts, $k = 5$, $\theta = 0.9$.

		Trial number										
		1	2	3	4	5	6	7	8	9	10	Average
β	0.1	5	8	9	6	5	5	6	8	4	4	6.0
	0.3	6	9	17	13	8	18	18	16	14	7	12.6
	0.5	10	16	26	30	19	29	20	27	15	19	21.1

where L is the likelihood function given in Equation (3). Hence, the expected value $E[t_d]$ is:

$$\sum_{t_d=1}^{\infty} \left(t_d \sum_{R_n \in \mathcal{R}} L(h, \theta, \beta | R_n) \right) \tag{19}$$

From the simulation study, we found that the average value of t_d lies between 4 and 9 for an abrupt shift (e.g., $\theta = 0.9$, $\beta = 0.1$). It lies between 20 and 45 for a moderate shift (e.g., $\theta = 0.9$, $\beta = 0.5$). Therefore, t_d is negligible when compared with t_{nodetect}, which is in the order of hundreds or even thousands. As a result, we expect t_{detect} is much smaller than t_{nodetect}, especially when there is a drastic change in the optimal class in the user profile.

The effect of a wrong decision on the part of the tracking algorithm will be to deteriorate the performance of the scheme. For example, a shift decision when none has occurred (false alarm) can unnecessarily reinitialize the learning model, and cause relearning on the latter's part. However, with proper choices of the parameters of the tracking algorithm (i.e., k_1 and k_2 described in the Integrating the Tracking Model section, or s_0 and s_1 described in the User Interest section), the probability of false alarms and missed detections can be made as small as desired.

Simulation Studies and Experiments

Simulation Studies

To validate our basic detection model, we have conducted some simulation studies to verify the effect of the degree of shift (e.g., abrupt shift and moderate shift) and the cost ratio parameter k on the number of feedback received (shift detection time) t_d. Because our detection model considers each class independently, we used a single class in the studies. The feedbacks were given according to the probability of relevance value of this class. The on-line shift detection module was invoked at each iteration.

The probability of relevance for a single class at the beginning was set to 0.9 (i.e., $\theta = 0.9$). At iteration 60, the probability of relevance was switched to a new probability of relevance β. The time for detecting a shift was recorded. Table 1 shows the shift detection time of each trial at $k = 5$ under different degree of shift. The first row represents an abrupt shift. The average shift detection time was 6. As we move down the table, the degree of shift becomes moderate. We can observe that the average shift detection time becomes larger if the degree of shift gets moderate.

We also varied the cost ratio parameter k. For each combination of β and k, we conducted 10 trials.

Figure 2 depicts the average shift detection time of various degree of shift and cost parameter k. The initial probability of relevance was 0.9 and the shift occurred at time 60. The plot also demonstrates that the average shift detection time increases as the degree of shift becomes moderate. Moreover, for a given β, the shift detection time becomes longer as we increase the cost ratio parameter k.

Filtering Experiments

We have conducted a series of experiments to evaluate our user interest tracking model and verify our analytical study. The document collection was created using records from the MEDLINE database. Bibliographic records, containing title, author, and abstract, of 6,000 documents from 29 cell biology classes (Table 2) were used.

To evaluate our model under different user behaviors, we conducted simulation of our filtering model on different user profiles. Filtering was conducted for multiple sessions both with and without the user interest tracking model. There were 15 documents in a session and feedbacks of the first S documents were given in each session. To evaluate the filtering performance, we use the *average filtering precision*, AFP, which is defined as:

$$\text{AFP} = \sum_{i=1}^{M} R_i/(M \times S)$$

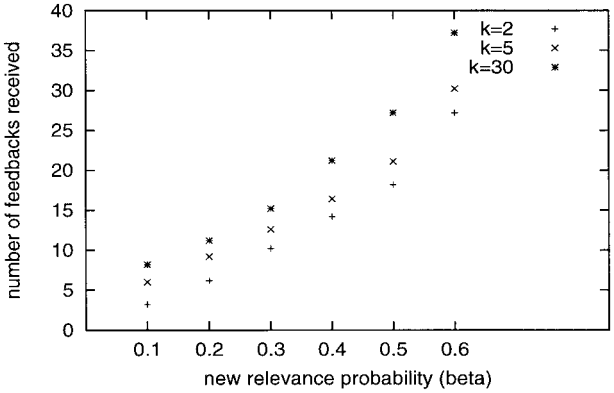

FIG. 2. The average shift detection time of various degree of shift and cost ratio parameter k.

TABLE 2. The classes used in the experiments.

Class label	Class name	Class label	Class name
1	Cell adhesion	16	Nerve
2	Antibody	17	Healing
3	Endocytosis	18	Phagocyte
4	Immunocompromised	19	Aggregation
5	Immune tolerance	20	Communication
6	Complement activation	21	Cytotoxicity immunologic
7	Evolution	22	Immune tolerance
8	Autoimmunity	23	Lymphocyte
9	Digest	24	Transport
10	Cytotoxicity	25	Cell survival
11	Bone	26	Clone
12	Cell death	27	Radiation
13	Chemotaxis	28	Formation
14	Cellular immunity	29	Cell movement
15	Regeneration		

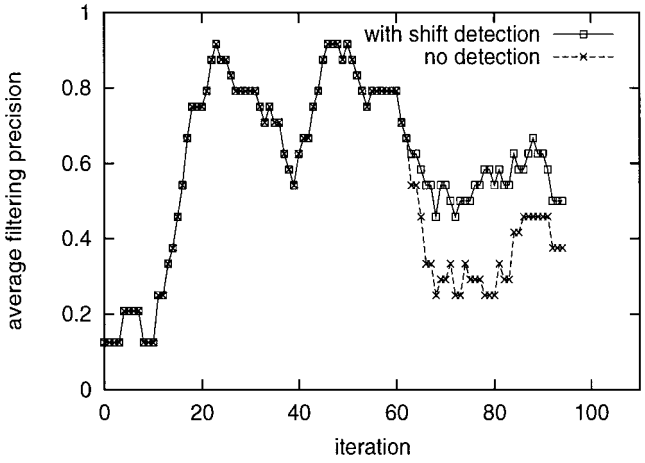

FIG. 4. The average filtering relevance of the experiment for user 1.

where M is the total number of sessions; R_i is the number of relevant documents in the first S documents within the ith session.

The underlying user profile vector for the first user is: 0.9 0.0 0.2 0.0 0.0 0.8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0. At the beginning, the user is very interested in the first class (i.e., Cell Adhesion) and the sixth class (i.e., Complement Activation). At iteration 60, the relevance value of the first class decreases from 0.9 to 0.1. It represents the fact that the user loses interests in Cell Adhesion. Therefore, the only topic in which the user remains interested is Complement Activation.

The result shows that the class selection vector Q almost converges to the first class, with q_1 being 0.999 and all other elements in Q being close to 0. Figure 3 depicts the root-mean-square (r.m.s.) error of the estimated user profile vector (i.e., the distance between the vectors Θ and $\hat{\Theta}$). The r.m.s. error became stable after around iteration 36. At iteration 60, due to the user interest shift, the r.m.s. error surged. With the user tracking model, the downward relevance shift of the first class was detected at iteration 62. Upon this successful shift detection, our model will revise the class selection vector Q so that q_1 and q_6 become 0.499, while all other elements in Q remains unchanged. The r.m.s. error reduced drastically. On the other hand, the system without the tracking model maintained a high r.m.s. error. Figure 4 shows the average filtering precision of each iteration. The filtering system with the user tracking model was able to converge to the sixth class more quickly with AFP value of 0.667 at around iteration 88. It has an average AFP of 0.60 because the user interest shift. On the other hand, the system without the user tracking model converged to the sixth class slower with AFP value of 0.458 at iteration 88. It has an average AFP of 0.33 because the user interest shift. It demonstrates a filtering performance improvement of over 82%.

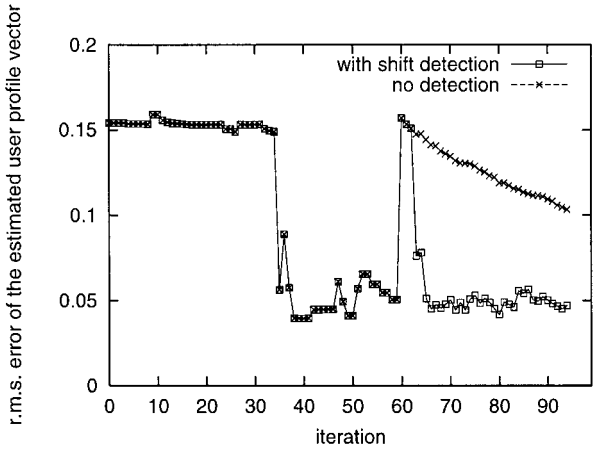

FIG. 3. The root-mean-square error of the estimated user profile vector for user 1.

The underlying user profile for the second user is: 0.9 0.0 0.2 0.0 0.0 0.4 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0. This profile is similar to the initial profile of the first user except the degree of interest for the sixth class.

At the beginning, the user is very interested in the first class (i.e., Cell Adhesion) and only is somewhat interested in the sixth class (i.e., Complement Activation). At iteration 50, the user switches the interest from the first class to the sixth class. In particular, the relevance value for the first class decreases from 0.9 to 0.1, and the relevance value for the sixth class increases from 0.4 to 1.0. The result shows that the class selection vector Q almost converges to the first class, with q_1 being 0.996 and all other elements in Q being close to 0. Figure 5 depicts the root-mean-square (r.m.s.) error of the estimated user profile vector. The r.m.s. error became stable after around iteration 35. At iteration 50, due to the user interest shift, the r.m.s. error surged. With the user tracking model, the downward relevance shift of the first class was detected at iteration 55. Upon this detection, the class selection vector Q was updated. The r.m.s. error reduced drastically. On the other hand, the system without

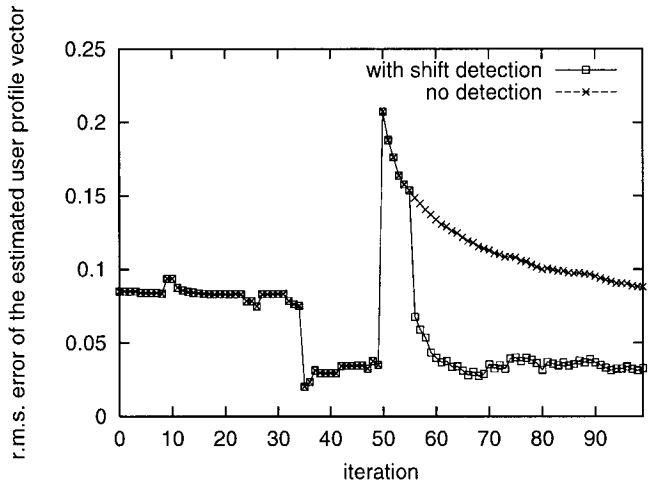

FIG. 5. The root-mean-square error of the estimated user profile vector for user 2.

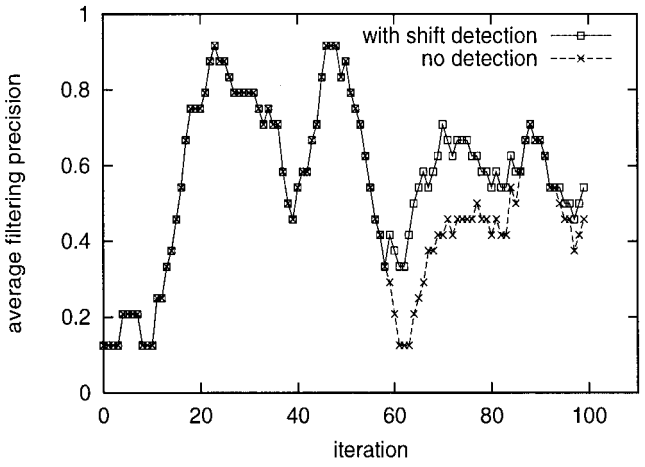

FIG. 6. The average relevance of the experiment for user 2.

the tracking model maintained a high r.m.s. error. Figure 6 shows the average filtering precision of each iteration. The filtering system with the user tracking model was able to converge to the sixth class more quickly with AFP value of 0.667 at iteration 71. It has an average AFP of 0.58, because the user interest shift. On the other hand, the system without the user tracking model converged to the sixth class slower with AFP value of 0.458 at iteration 71. It has an average AFP of 0.37, because the user interest shift. It demonstrates a filtering performance improvement of roughly 57%.

The underlying user profile for the third user is: 0.45 0.0 0.0 0.0 0.0 0.05 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0. At the beginning, the user is somewhat interested in the first class (i.e., Cell Adhesion) and has little interest in the sixth class (i.e., Complement Activation). At iteration 120, the user becomes interested in the sixth class in which he/she was not interested. In particular, the relevance value for the sixth class increases from 0.05 to 1.0. The result shows that the class selection vector Q almost converges to the first class, with q_1 being almost 1.0, and all other elements in Q being close to 0. Figure 7 depicts the root mean square (r.m.s.) error of the estimated user profile vector. The r.m.s. error became stable after around iteration 100. At iteration 120, due to the user interest shift, the r.m.s. error surged. With the user tracking model, the upward relevance shift of the sixth class was detected at iteration 125. Upon this detection, the class selection vector Q was updated. The r.m.s. error reduced drastically. On the other hand, the system without the tracking model maintained a high r.m.s. error. Figure 8 shows the average filtering precision of each iteration. The filtering system with the user tracking model was able to converge to the sixth class more quickly with AFP value of 0.833 at iteration 160. It has an average AFP of 0.61 since the user interest shift. On the other hand, the system without the user tracking model converged to the sixth class slower with AFP value of 0.542 at iteration 160. It has an average AFP of 0.49, because the user interest shifts. It demonstrates a filtering performance improvement of roughly 25%.

Conclusions

We address the issue of tracking user interest profile by developing a shift detection model based on a Bayesian framework. Bayesian modeling provides a rigorous means to achieve the objective. The detection model is integrated into a text document filtering system, which employs a reinforcement learning for automatic user profile acquisition. An analytical study on the rate of convergence is presented to gain insight on the performance of our proposed Bayesian shift detection model. We have conducted simulation studies to investigate the influence of different patterns of interest shift on system performance. The results demonstrate that the user interest shift modeling is able to track the changes of user interests on-line. This capability

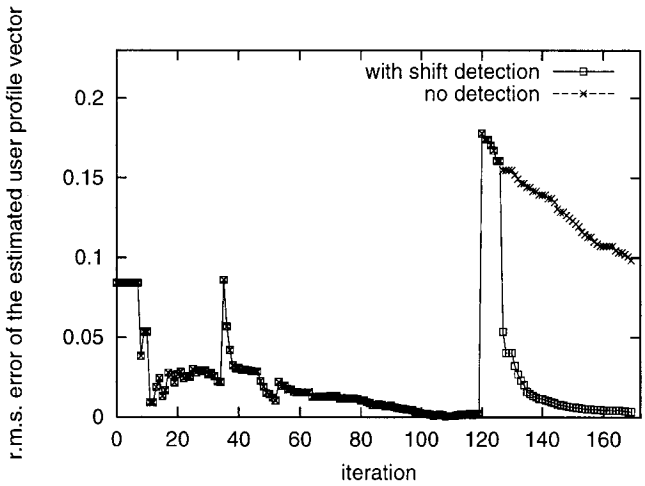

FIG. 7. The root-mean-square error of the estimated user profile vector for user 3.

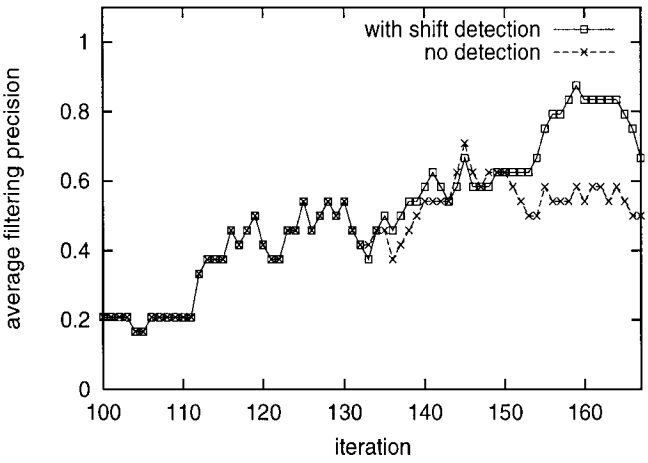

FIG. 8. The average relevance of the experiment for user 3.

can improve the filtering performance by effectively maintaining the fidelity of the user profile.

As part of future extension to the system and further analysis of shift detection, we wish to relax the assumption that the classes remain fixed over the lifetime of the system. A possible approach would be to identify classes for which the relevance feedback is almost stochastic, and split them into classes with narrower scope until feedback stabilizes. Additionally, classes for which feedback remains negative over a long period of time may eventually be dropped. To accommodate class space contraction or expansion, the interest values would need to be redistributed accordingly, and the algorithms for reinforcement learning and shift detection would have to be appropriately modified to handle such revisions of the interest values.

Finally, we intend to enhance the current approach by considering different types of relationships among concepts. The relationships such as hierarchy and subsumption among concepts are useful in learning a more accurate user profile. Another future direction is to investigate the application of our Bayesian tracking model with a heterogeneous and large group of users. This research could reveal among other things how backgrounds of users influence feedback patterns, and how that may in turn influence shift detection.

Acknowledgments

We would like to thank the members of the SIFTER group for fruitful discussion and input on the user interest learning model used in this article. Wai Lam was partially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region (Project No. CUHK4385/99E). Javed Mostafa was funded by a visiting scholar grant from the Systems Engineering & Engineering Management Department of the Chinese University of Hong Kong and partially supported by the NSF Digital Libraries Phase II grant IIS-9817572. We are also grateful to the anonymous reviewers for their constructive comments.

Appendix

To compute $S^{(u)}(R_n)$, we first calculate the posterior probability of no upward shift has occurred given R_n (i.e., $P((h \geq n)|R_n)$ using Bayes theorem as follows:

$$\left[\left(\sum_{j=n}^{\infty} p_s(j)\right)\int_0^1\int_\theta^1 L(n,\,\theta,\,\beta|R_n)p_d(\theta,\,\beta)\,d\beta\,dd\right]/\mathcal{N} \quad (20)$$

where $\mathcal{N}$ is a normalization factor given by:

$$\mathcal{N} = \sum_{j=0}^{\infty}\left[p_s(j)\int_0^1\int_\theta^1 L(j,\,\theta,\,\beta|R_n)p_d(\theta,\,\beta)\,d\beta\,d\theta\right] \quad (21)$$

As a result, $S^{(u)}(R_n)$ can be readily obtained from Equation (20) as follows:

$$1 - \left[\left(\sum_{j=n}^{\infty} p_s(j)\right)\int_0^1\int_\theta^1 L(n,\,\theta,\,\beta|R_n)p_d(\theta,\,\beta)\,d\beta\,d\theta\right]/\mathcal{N} \quad (22)$$

To calculate the posterior probability of a downward shift $S^{(d)}(R_n)$, we follow a similar technique and modify Equation (20) and Equation (21) to perform the integration over the range $0 < \beta \leq \theta < 1$ instead of using the range $0 < \theta \leq \beta < 1$. Therefore, $S^{(d)}(R_n)$ can be computed as follows:

$$1 - \left[\left(\sum_{j=n}^{\infty} p_s(j)\right)\int_0^1\int_\beta^1 L(n,\,\theta,\,\beta|R_n)p_d(\theta,\,\beta)\,d\theta\,d\beta\right]/\mathcal{N} \quad (23)$$

where $\mathcal{N}$ is given by:

$$\mathcal{N} = \sum_{j=0}^{\infty}\left[p_s(j)\int_0^1\int_\beta^1 L(j,\,\theta,\,\beta|R_n)p_d(\theta,\,\beta)\,d\theta\,d\beta\right] \quad (24)$$

To evaluate the integration in the above formulae, we make use of the following relationship regarding the beta function:

$$\int_0^1\int_{a_1}^1 a_2^{t_1}(1-a_2)^{t_2}\,da_2\,da_1 = \int_0^1 a_2^{t_1+1}(1-a_2)^{t_2}\,da_2 = \mathrm{Beta}(t_1+2,\,t_2+1)$$

where Beta($\cdot,\cdot$) denotes the beta function. As a result, both $S^{(u)}(R_n)$ and $S^{(d)}(R_n)$ can be obtained by a sequence of beta function evaluations.

References

Balabanovic, M., & Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. Communications of the ACM, 40(3), 66–72.

Belkin, N.J., & Croft, W.B. (1992). Information filtering and information retrieval: Two sides of the same coin. Communications of the ACM, 35(12), 29–38.

Carbonell, J., Yang, Y., Lafferty, J., Brown, R., Pierce, T., & Liu, X. (1999). CMU Report on TDT-2: Segmentation, detection and tracking. Proceedings of the DARPA Broadcast News Workshop, Virginia, (pp. 117–120). San Mateo, CA: Morgan Kaufmann.

Fiscus, J., Doddington, G., Garofolo, J., & Martin, A. (1999). NIST's 1998 topic detection and tracking evaluation (TDT2). Proceedings of the DARPA broadcast news workshop, Virginia (pp. 19–24). San Mateo, CA: Morgan Kaufmann.

Foltz, P.W., & Dumais, S.T. (1992). Personalized information delivery: An analysis of information filtering methods. Communications of the ACM, 35(12), 51–60.

Goker, A., & McCluskey, T.L. (1991). Towards an adaptive information retrieval system. In Z.W. Ras & M. Zemankova (Eds.), Methodologies for intelligent systems: Proceedings of the 6th international symposium of ISMIS, Charlotte, NC, (pp. 348–357). New York: Springer-Verlag.

Hull, D., & Robertson, S. (1999). The TREC-8 filtering track final report. In E.M. Voorhees & D.K. Harman (Eds.), Proceedings of the eighth text retrieval conference, Gaithersburg, MD, (pp. 1–14). Gaithersburg, MD: Dept. of Commerce, National Institute of Standards and Technology.

Jennings, A., & Higuchi, H. (1992). A Personal news service based on a user model neural network. IEICE Transactions on Information Systems, 75, 198–209.

Klinkenberg, R., & Renz, I. (1998). Adaptive information filtering: Learning in the presence of concept drifts. AAAI Workshop on Learning for Text Categorization (pp. 1–8).

Maes, P. (1995). Intelligent software. Scientific American, September, 84–86.

Mostafa, J., Mukhopadhyay, S., Lam, W., & Palakal, M. (1997). A Multilevel approach to intelligent information filtering: Model, system, and evaluation. ACM Transactions on Information Systems, 15(4), 368–399.

Mukhopadhyay, S., Mostafa, J., Palakal, M., Lam, W., Xue, L., & Hudli, A. (1996). An adaptive multilevel information filtering system. Proceedings of the 5th international conference on user modeling, Kailua-Kona, HI (pp. 21–28).

Narendra, K.S., & Thathachar, M.A.L. (Eds.). (1989). Learning automata—An introduction. Englewood Cliffs, NJ: Prentice Hall.

Papka, R., Allan, J., & Lavrenko, V. (1999). UMASS approaches to detection and tracking at TDT2. Proceedings of the DARPA Broadcast News Workshop, Virginia (pp. 111–116). San Mateo, CA: Morgan Kaufmann.

Pazzani, M., & Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. Machine Learning, 27, 313–331.

Salton, G., & McGill, M.J. (1983). Introduction to modern information retrieval. New York: McGraw-Hill.

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. Machine Learning, 23, 69–101.

Zacks, S., & Barzily, Z. (1981). Bayes procedures for detecting a shift in the probability of success in a series of Bernoulli trials. Journal of Statistical Planning and Inference, 5, 107–119.