An Adaptable Connectionist Text-Retrieval System With Relevance Feedback

M. R. Azimi-Sadjadi, *Senior Member, IEEE*, J. Salazar, S. Srinivasan, and S. Sheedvash

***Abstract*—This paper introduces a new connectionist network for certain domain-specific text-retrieval and search applications with expert end users. A new model reference adaptive system is proposed that involves three learning phases. Initial model-reference learning is first performed based upon an ensemble set of input–output of an initial reference model. Model-reference following is needed in dynamic environments where documents are added, deleted, or updated. Relevance feedback learning from multiple expert users then optimally maps the original query using either a score-based or a click-through selection process. The learning can be implemented, in regression or classification modes, using a three-layer network. The first layer is an adaptable layer that performs mapping from query domain to document space. The second and third layers perform document-to-term mapping, search/retrieval, and scoring tasks. The learning algorithms are thoroughly tested on a domain-specific text database that encompasses a wide range of Hewlett Packard (HP) products and for a large number of most commonly used single- and multiterm queries.**

***Index Terms*—Connectionist networks, learning algorithms, query mapping, relevance feedback, text retrieval.**

I. Introduction

THE focus of most general-purpose text-retrieval systems (TRSs) is to apply search and content matching to deal effectively and consistently with an overwhelmingly large volume of information. In these systems, the user typically modifies and enhances the query text in a subjective manner in order to narrow the domain of the search. The search process typically culminates at a list of the documents from which the user identifies, either implicitly or explicitly, the most relevant ones after navigating or browsing through the list in the order of the documents' "retrieval status values" or relative scores. This trial-and-error-based query modification does not allow for the incorporation of the user expertise or feedback to influence the suggested solutions. Moreover, identification of an optimum query that carries the required concept is difficult or sometimes impossible, even for expert users.

TRSs that allow for user contribution typically utilize "relevance feedback" [1] from the users to modify the original query in order to meet the users' requirements and improve the retrieval efficiency. It is expected that the modified query would deliver a more refined list of documents than that delivered by the original query. A mechanism to implement relevance feedback was originally introduced by Rocchio [2]. In this algorithm, the query is selectively modified using the relevant documents to achieve improved retrieval. Ide [3] showed that by incorporating nonrelevant documents as well as the relevant ones, retrieval accuracy could be improved even further. In [4] and [5], query modification using Rocchio's formula and query expansion schemes are used, while others rely on support vector machines (SVMs) [6]–[8] or on boosting algorithms [9].

The learning capabilities of a neural network (NN) provide a framework around which an adaptive TRS could be built [10], [11]. Kwok [12] devised a probabilistic document retrieval system implemented using a feedforward NN. The search results are ranked in the order of conditional probabilities that are estimated based on a sample of relevant documents to the query. In [13] and [14], a backpropagation neural network (BPNN) was used as a retrieval system. The retrieved documents for a query are compared against the corresponding relevant document set and if any nonrelevant document is also retrieved the network relearns to remove it from the list. The documents are simply listed as relevant or nonrelevant and are not ordered in accordance with their relevancy to the query. The results in [13], however, indicated poor performance of the network when the training was done only with a limited set of relevant documents. Boughanem *et al.* [15] used an unsupervised network with two fully interconnected layers where the neurons in the first layer represent terms and those in the second layer represent documents. Hebb's learning rule [16] was used to modify the connection weights. Recently, Bouchachia [17] proposed a hierarchical fuzzy NN architecture for document retrieval. The documents and queries are represented as fuzzy sets and a two-layer NN is used to learn the implicit relationship among the documents.

Tong and Koller [6] developed an active learning scheme using SVM, called $\mathrm{SVM}_{\mathrm{Active}}$, to quickly and effectively learn the boundary that separates samples satisfying the user's query concept from the rest of the text database. To apply relevance feedback, the user is asked to label a small set of documents as relevant or nonrelevant classes. Using these initially labelled

Manuscript received December 28, 2005; revised September 15, 2006; accepted February 7, 2007. This work was supported by the Hewlett Packard, Boise, ID management and business teams under Contract 50B000553.

M. R. Azimi-Sadjadi is with the Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523 USA (e-mail: azimi@engr.colostate.edu).

J. Salazar was with the Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523 USA. He is now with the Instituto Teconológico y de Estudios Superiores de Monterrey (ITESM), Toluca, Mexico.

S. Srinivasan was with the Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523 USA. He is now with the Information System Technologies Inc., Fort Collins, CO 80521 USA.

S. Sheedvash is with the Hewlett Packard, San Diego, CA 92127 USA (e-mail: sassan_sheedvash@hp.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2007.895912

samples, the system finds the separating hyperplane and performs a series of querying rounds. In each round, the hyperplane parameters are adjusted based upon user votes on the unlabeled samples closest to the hyperplane. Upon the completion, the $\mathrm{SVM}_{\mathrm{Active}}$ returns the top k-most relevant samples that are farthest from the hyperplane on the query concept side (i.e., relevant samples). Comparison of the $\mathrm{SVM}_{\mathrm{Active}}$ results with those obtained using the query-by-committee (QC) algorithm [18] indicated the superiority of $\mathrm{SVM}_{\mathrm{Active}}$ regardless of the initial number of labeled samples.

In [19], using the risk minimization framework of SVM and the description-oriented class of ranking functions [20], a learning method for linear retrieval function using click-through data is presented. Discordance pairs between the ranked documents and the click-through data are used to create a set of training samples. The goal is to learn a ranking function with the minimum number of discordance pairs. This is equivalent to maximizing the Kendall's τ [21] factor, which measures the degree of correspondence between two ranking schemes. A suboptimal solution is suggested by formulating the SVM problem with a penalizing factor that accounts for the errors of the discordance pairs. More recently, Chang and Chen [22] developed a new query reweighting mechanism based upon the relevance feedback from users. Genetic algorithm is employed to assign optimal weights to the terms in the user query in order to improve the overall document retrieval accuracy. Experimental data on a small database showed improved precision and recall rates. In [23], a new query expansion scheme was proposed that uses the recorded user logs to extract implicit relevance information, and hence, improve the retrieval accuracy. The associations between the terms in the user queries and documents are then established based upon the user logs. The results indicate that exploiting user logs is indeed effective for improving the overall retrieval accuracy.

Current TRS tools are typically designed for general purpose text search and retrieval applications. In domain-specific environments with expert end users, such as customer support of various corporations (application considered in this paper), hospital databases such as MEDLINE [24], homeland security [25], and other similar applications, adaptable TRS is needed to continuously learn from the users and enhance the relevancy of the suggested solutions without the slow process of authoring or modifying the information content within the query directly. In these environments, it is important to accurately meet the expert user requirements when queries normally do not receive numerous relevance feedback. Additionally, it is crucial to capture and retain, within the adaptable TRS, the expertise of different tier expert users for more refined future searches. This adaptability must be achieved without jeopardizing the stability of the previously learned information.

In [25], a domain-specific adaptable text search engine, referred to as *vista* system was developed that supports context-sensitive information access and monitoring for effective and timely information exchange and coordination among various homeland security and emergency management agencies. The goal of [25] was to develop new technologies that can dynamically exploit the output of new information providers to offer both vastly improved information/situation awareness and the ability to coordinate crisis response. The *vista* system consists of several adaptable engines that are designed for specific databases at different agencies, e.g., Federal Bureau of Investigation (FBI), Central Intelligence Agency (CIA), and Federal Emergency Management Agency (FEMA). These local search profiles manage their own set of documents and receive limited training from within expert users via relevance feedback learning. Queries are keyword-based, e.g., "anthrax letters," leading to a list of relevant/nonrelevant documents. Leveraging on the expert user's operational context, the system is able to produce refinements to user queries and identify the most relevant "query anchors" to the user tasks via the relevance feedback. The *vista* system in [25] also utilizes "concept switching" that allows for expanding the domain of search across different communities (or agencies) for broader concept matching and search.

In this paper, an adaptable and robust TRS for special-purpose application is developed which incorporates the users' information and expertise to improve the relevancy of solutions. The proposed approach uses a new framework referred to as model-reference text retrieval system (MRTRS), which is inspired from the well-known model-reference adaptive control theory. The proposed learning involves the following three phases: 1) initial model-reference learning, 2) model-reference following, and 3) relevance feedback learning from expert users. These learning phases that can be implemented effectively using a three-layer connectionist network are driven based upon an ensemble of input–output relations from a reference TRS model (phases 1 and 2) or from the user feedback (phase 3) and their specific characteristics such as relevance feedback frequencies and expertise level of the users. The purpose of initial model-reference learning is to set up (or initialize) the weights of the first layer to capture the behavior of a reference model or the results of an indexing system. The latter is motivated by the fact that even though an "ideal" TRS may not be available, one can always have access to a set of input–output relations that can be used to initially train the MRTRS. The model-reference following is needed when documents are added, deleted, or updated, using both structural and weight adaptation mechanisms. Structural adaptation corresponds to adding or deleting nodes to the hidden and output layers of the network. New relevance feedback learning methods are also developed for single- and multiterm queries using either score-based or click-through feedback from multiple users of different expertise levels. Relation of the proposed learning to SVM is also established. The effectiveness of the developed algorithms is demonstrated on a domain-specific text database for customer support on various ranges of HP products consisting of over 32 000 documents. This database involves over 108 000 terms and 5900 commonly used keyword-based single- and multiterm queries. A benchmarking with the BM25 method [26] is also presented.

The organization of this paper is as follows. Section II presents the proposed MRTRS and its different operational phases. Section III presents the initial model-reference learning and the query mapping mechanism. The implementation using a three-layer connectionist network is also discussed. In Section IV, model-reference following learning is introduced

to capture the changes in the model due to document addition, deletion, or updating. Structural and weight adaptation schemes are also proposed to implement these operations using the network. Section V develops new relevance feedback learning algorithms from multiple users using both score-based and click-through selection processes. Relationship to SVM-based learning is also demonstrated. The test results on a domain-specific database of various HP products are presented in Section VI. Finally, conclusions and observations are given in Section VII

II. Model Reference TRS

A typical TRS consists of several subsystems namely storage, document indexing system, user interface, and search/retrieval system. The indexing system processes each document in the database and generates an indexed file based upon certain attributes in the documents. These attributes represent the importance of different terms contained in the document. These attributes form a vector $\underline{\mathrm{d}}_j = [d_{1j}, d_{2j}, \ldots, d_{Mj}]^T$ that represents jth document, $j \in [1, N]$, where N is the total number of documents in the database and T represents the transposition operation. The component $d_{ij}, i \in [1, M]$ where $M \gg N$ is the total number of terms in the entire corpus, gives the weight or importance of the term t_i in document vector $\underline{\mathrm{d}}_j$. When a specific term is not present in the document, the corresponding entry in the vector is 0.

The retrieval and search system performs, upon the user request or query, a similarity measure $s(\underline{\mathrm{q}}, \underline{\mathrm{d}}_j)$ between the submitted query $\underline{\mathrm{q}}$ and each document vector $\underline{\mathrm{d}}_j$ in the entire database and delivers $n \leq N$ closest matches. In the simplest case, this similarity measure could be $s(\underline{\mathrm{q}}, \underline{\mathrm{d}}_j) = \underline{\mathrm{d}}_j^T \underline{\mathrm{q}}$. The search and matching processes result in a list of the relevant and nonrelevant documents arranged in order of their relevancy (or match) to the submitted query. The "retrieved status value" or relative score of each listed document is clearly a function of the adopted similarity measure and the model used by the particular TRS [27]. If the search results and the retrieved status values are not arranged in their relevancy to the items of interest, the user may need to interactively modify the query until the refinements lead to results that most closely carry the required query concept. Since the identification of an optimum query is difficult and some times an impossible task, the user can browse through the list of documents and identify the most relevant document(s). Such a user relevance feedback is typically applied in a binary (positive or negative cases) fashion to obtain the desired results.

In special-purpose TRS, it is crucial that the system exactly meets the specific requirements of the expert users by proper adaptation of the system parameters while maintaining the previous learning. The main benefit of the proposed approach is that the expert users can contribute to the decision-making capability of the system and enhance the relevancy of the suggested solutions. This can be accomplished without the slow and expensive process of authoring or modifying the information content within the query directly. The efficiency of the system improves over time as expert users actively provide relevance information in the context of their needs. The learning eventually culminates at the optimal association for mapping queries to documents, which will be captured in the system for future use. The proposed system offers high retrieval accuracy needed in special-purpose applications and more importantly preserves stability of the stored information, while offering plasticity needed in these situations.

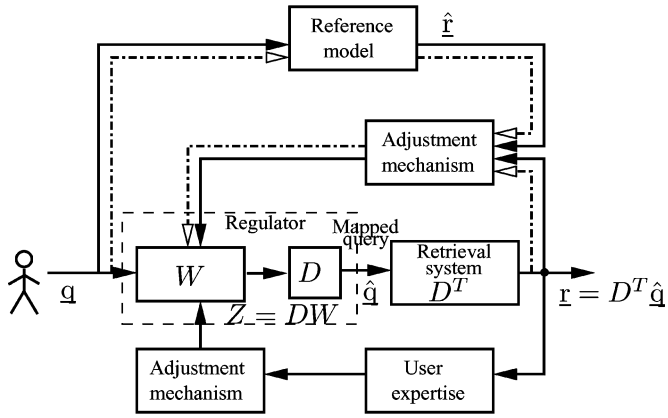


Fig. 1. MRTRS.

Fig. 1 shows the block diagram of the proposed MRTRS. As discussed previously, there are the following three operational phases for this system: 1) initial model-reference learning under invariant model or static environment, 2) model-reference following in dynamic environments where documents are to be added, deleted, or updated, and 3) learning from users through score-based relevance feedback or click-through selection. These different modes of operation are described in the next sections.

III. Initial Model-Reference Learning

The goal of initial model-reference learning phase is to set up or initialize the weights of the adaptable TRS in Fig. 1 based upon an ensemble set of training samples. If a reference TRS model exists, then this ensemble set corresponds to a set of queries and the corresponding listed documents and their retrieval scores. The function of the initial model reference learning, in this case, is to capture the behavior of the unknown and possibly inflexible *black-box* TRS. This is due to the fact that, in practice, access to the internal parameters of the reference model is not typically available or feasible to influence its behavior via relevance feedback. Thus, our MRTRS becomes an independent adaptable system on top of any available reference model to dynamically incorporate user feedback and/or other input–output characteristics. In absence of a reference model, the indexed documents generated by a simple document indexing system (see Remark II.1) can be used. In this case, only the document vectors $\underline{\mathrm{d}}_j$s are needed to initialize the system, without the need to have queries and their associated listed documents. It will be shown later that even with this crude initialization the system quickly learns to produce the desired solutions using relevance feedback learning.

In the initial learning phase, the query mapping subsystem in Fig. 1 plays a similar role as an adaptable regulator in a control system. It learns to map the original submitted query $\underline{\mathrm{q}}$ to the modified query (control signal) $\hat{\underline{\mathrm{q}}}$ that yields the desired response or the document list $\hat{\underline{\mathrm{r}}}$. Note that the dimension of the original query space $\underline{\mathrm{q}}$ is the same as that of the mapped query

$\underline{\hat{q}}$. The process is shown using the dotted–dashed lines in the upper loop of the block diagram in Fig. 1. The desired response $\underline{\hat{r}}$ for a submitted query is generated by the reference model, if available. The retrieval system, which plays a similar role as a plant in an adaptive control system, is a linear mapping system described by matrix D^T where $D = [\underline{d}_1 \underline{d}_2 \dots \underline{d}_j \dots \underline{d}_N]$ is the document matrix for the entire collection and $\underline{d}_j$ is the jth document vector of size $M \times 1$, as defined previously. The document matrix is generated either by the indexing system within the TRS or any other indexing system. Once the initial model reference learning is completed, the reference model and the components shown in dotted lines are removed for subsequent relevance feedback learning phase.

In the proposed system, initial model reference learning could be accomplished either in a regression mode using a score-based matching or in a classification mode using an SVM-type framework as will be discussed in the next section.

A. Regression Mode

In the regression model, the goal of this initial model-reference learning is to find the optimal mapped query $\underline{\hat{q}}_i$ that yields the desired response $\underline{\hat{r}}_i$ for the submitted original ith query $\underline{q}_i$. Since typically $M \gg N$, this parameter estimation problem is underdetermined. Thus, the problem can be cast as a minimum-norm least square (LS) [28] where it is desirable to find a mapped query $\underline{\hat{q}}_i$ with minimum distance from the origin (i.e., small number of terms) subject to constraint $D^T \underline{\hat{q}}_i = \underline{\hat{r}}_i$, where $\underline{\hat{r}}_i = [\hat{r}_{i1}, \hat{r}_{i2}, \dots, \hat{r}_{iN}]^T$ is the desired score vector for the ith submitted query. Accordingly, we can construct the Lagrangian function

$$J(\underline{\hat{q}}_i, \underline{w}_i) = \frac{1}{2} \underline{\hat{q}}_i^T \underline{\hat{q}}_i + \sum_{j=1}^{N} w_{ij} \left(\hat{r}_{ij} - \underline{d}_j^T \underline{\hat{q}}_i \right) \tag{1}$$

where $\underline{w}_i = [w_{i1} \dots w_{iN}]^T$ and w_{ij}s are Lagrangian multipliers. Differentiating $J(\underline{\hat{q}}_i, \underline{w}_i)$ with respect to (w.r.t.) $\underline{\hat{q}}_i$ and setting the result to zero yields

$$\underline{\hat{q}}_i = \sum_{j=1}^{N} w_{ij} \underline{d}_j = D \underline{w}_i. \tag{2}$$

Now, taking the derivative of $J(\underline{\hat{q}}_i, \underline{w}_i)$ w.r.t. $\underline{w}_i$ and setting the result to zero yields $D^T \underline{\hat{q}}_i = \underline{\hat{r}}_i$. Combining with (2) gives the solution for the optimal $\underline{w}_i$

$$\underline{w}_i = (D^T D)^{-1} \underline{\hat{r}}_i \tag{3}$$

which generates the desired result $\underline{\hat{r}}_i$ at the output of the retrieval system. Thus, the LS solution for $\underline{\hat{q}}_i$ lies in the space spanned by the documents. This can be viewed as a generalization of the Rocchio's formula [4] where all the documents are included and their associated weights are obtained using the learning mechanism in this section.

The objective function $J(\underline{\hat{q}}_i, \underline{w}_i)$ can equivalently be represented in terms of documents $\underline{d}_i$s and the Lagrangian multipliers leading to the following "dual problem":

$$\xi(\underline{w}_i) = \sum_{j=1}^{N} w_{ij} r_{ij} - \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{N} w_{ij} w_{ik} \underline{d}_j^T \underline{d}_k \tag{4}$$

$$= \underline{w}_i^T \underline{r}_i - \frac{1}{2} \underline{w}_i^T D^T D \underline{w}_i \tag{5}$$

which should be maximized w.r.t. $\underline{w}_i$. This cost function is represented in terms of weights and dot product of documents $\underline{d}_j^T \underline{d}_k$. This implies that there is a close similarity between the proposed query mapping approach and SVM in the original linear space. This is described in the next section.

B. Classification Mode

The initial model reference learning and relevance feedback (Section V) in our framework can also be implemented in classification mode, if desired. To see this, let us compare the dual cost function [8] of the SVM to that in (4) or (5). If we change $\underline{w}_i \Rightarrow \Lambda_i \underline{w}_i$ where Λ_i is a diagonal matrix with elements 1 (class 1) or -1 (class 2), and further $\Lambda_i \underline{r}_i = \underline{1}$ with $\underline{1}$ being the one vector, then the cost function in (5) becomes exactly the same as that of SVM. Note that since $\Lambda_i^2 = I$, then $\underline{r}_i = \Lambda_i \underline{1}$ which implies that the score vector consists of elements 1 (relevant documents) or -1 (nonrelevant documents) as in a two-class problem. Now, taking the partial derivative of the resultant $\xi(\underline{w}_i)$ w.r.t. $\underline{w}_i$ and setting the result to zero yields

$$\underline{w}_i = \Lambda_i (D^T D)^{-1} \underline{\hat{r}}_i. \tag{6}$$

Moreover, from the modified primal problem in (1)

$$J(\underline{\hat{q}}_i, \underline{w}_i) = \frac{1}{2} \underline{\hat{q}}_i^T \underline{\hat{q}}_i + \underline{w}_i^T \Lambda_i (\underline{r}_i - D^T \underline{\hat{q}}_i) \tag{7}$$

$$= \frac{1}{2} \underline{\hat{q}}_i^T \underline{\hat{q}}_i + \underline{w}_i^T (\underline{1} - \Lambda_i D^T \underline{\hat{q}}_i) \tag{8}$$

the solution for the optimal query for this two-class classification problem becomes

$$\underline{\hat{q}}_i = D \Lambda_i \underline{w}_i. \tag{9}$$

Clearly, this optimal query yields the desired output of $D^T \underline{\hat{q}}_i = \underline{r}_i$. These results show that the proposed learning can be implemented in either regression mode or classification mode, which is closely related to the SVM framework. This offers the potential for development of kernel-based text search and retrieval machines using the proposed framework.

C. Connectionist Network Implementation

The query mapping and retrieval processes in the feedforward path of Fig. 1 can be implemented using a simple three-layer network, as illustrated in Fig. 2. This network facilitates the implementation and understanding of all the three learning phases as will be shown later. Moreover, it provides a unified system that captures all the components in Fig. 1 excluding the reference model, if that exists. Each node j in the first and third layers represents a document $\underline{d}_j$, $j \in [1, N]$, whereas each node

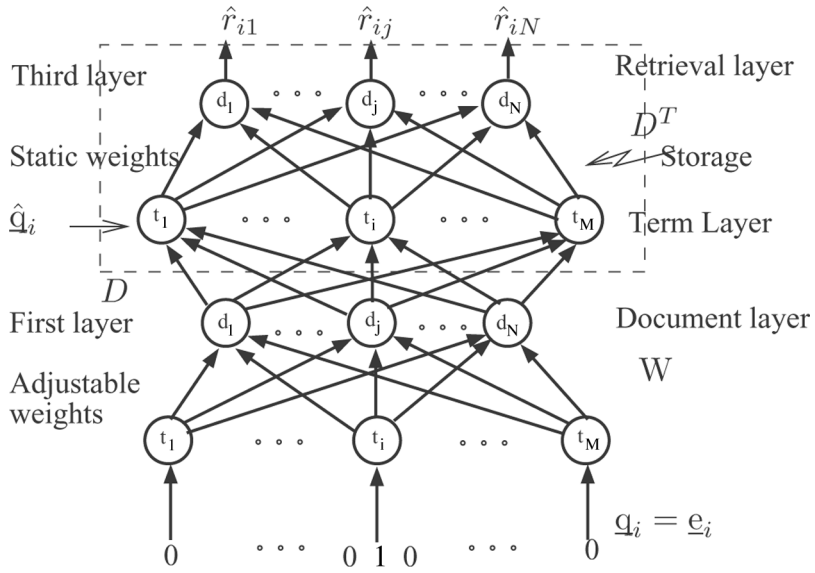


Fig. 2. Proposed flexible network structure.

$i \in [1, M]$, in the second layer represents a term t_i. The connection weight from node j in the first layer to node i in the second layer is d_{ij}. Similarly, the connection weight between node i in the second layer and node j in the third layer is the same weight d_{ij}. Thus, the weight matrices for the second and third layers are D and D^T, respectively. These weights remain unchanged unless the documents are reindexed or updated. The weight matrix (adjustable) of the first layer is $W = [\underline{w}_1 \underline{w}_2 \dots \underline{w}_M]$. As shown before, in the initial model-reference learning, the objective is to find W to capture input–output behavior of the reference system for every query in the ensemble set. The first and second layers combined form the mapped query at the term layer, i.e., $\hat{\underline{q}}_i = DW\underline{q}_i$, which in turn yields the retrieved documents and their desired score vector $\hat{\underline{r}}_i = D^T DW\underline{q}_i$ at the output of the retrieval layer for the optimal W. Consequently, the first and second layers combined function like the regulator in Fig. 1. Note that the space that spans the original input query is the same as the mapped term space (second layer) that forms the optimal query.

In this network, the inputs are indexed representing different possible terms in the submitted query. Each input can take either 0 or 1 values depending on the absence or presence of the corresponding term in the query. A single-term query consisting of term t_i can be represented by input vector $\underline{q}_i = \underline{e}_i$ where $\underline{e}_i = [0 \dots 1 \dots 0]$ is a unit norm vector with the ith component being 1. If this single-term query is applied to the network, the output of the first layer extracts the ith column of weight matrix W, i.e., $W\underline{e}_i = \underline{w}_i$, which in turn generates the mapped query $\hat{\underline{q}}_i = D\underline{w}_i$ at the output of the second layer. Thus, $\underline{w}_i$ is the weight vector that connects the term t_i to all the document (first) layer nodes. This implies that learning for each single-term query can be performed independently by only updating $\underline{w}_i$ to meet the desired scores at the output layer. This interesting feature of this network guarantees the stability of the weights for other queries while offering flexibility that is needed to accommodate new model- or user-based information.

For the initial training phase, (3) can be rewritten as

$$\underline{w}_i^{(0)} = A_0^{-1} \hat{\underline{r}}_i^{(0)} \tag{10}$$

where $\underline{w}_i^{(0)}$ is the initial weight vector, $\hat{\underline{r}}_i^{(0)}$ is the initial document rank vector provided either by the TRS or the indexing system (see Remark II.1), and $A_0 = D^T D$ is a symmetric positive–definite (PD) Gram matrix with element $[A_0]_{i,j} = \underline{d}_i^T \underline{d}_j$. The superscript "0" is used to represent the initial training phase. Equation (10) is solved once for all the queries in the ensemble set. The Gram matrix $A_0 \in \mathbb{R}^{N \times N}$ can be expressed as $A_0 = G_0 G_0^T$ where $G_0 \in \mathbb{R}^{N \times N}$ is a lower triangular matrix with positive diagonal entries. Fast algorithms using Cholesky decomposition and triangular matrix inversion [29] can be applied to solve for the weight vector $\underline{w}_i^{(0)}$ for each query, i.e., $\underline{w}_i^{(0)} = G_0^{-T} G_0^{-1} \hat{\underline{r}}_i^{(0)}$.

Once the initial model reference learning is completed, the weights of the first layer can be updated in response to the relevance feedback from expert users. Relevance feedback learning will only impact those weight vectors corresponding to the terms in the submitted query. This process will be discussed in Section V

Remark III.1: From the definition of W weight matrix and the result in (3), it can easily be shown that $W = (D^T D)^{-1} R$, where $R = [\underline{r}_1, \dots \underline{r}_M]$ is the score matrix for all the queries. It is interesting to note that when a reference TRS is not present, the results of an indexing system can be directly used to initially train the network. In this case, we have $W = (D^T D)^{-1} D^T$ which yields the regulator mapping matrix of $P_D = D(D^T D)^{-1} D^T$, i.e., the projection matrix associated with document space D. This implies that the regulator projects the original query onto a space spanned by the documents to generate the mapped query. Clearly, for the ith query $\underline{q}_i = \underline{e}_i$, this gives the retrieved score vector $\underline{r}_i = [d_{i1}, \dots d_{iN}]^T$ that contains the attributes or the weights for the ith term in all documents. Thus, the document that has the highest weight

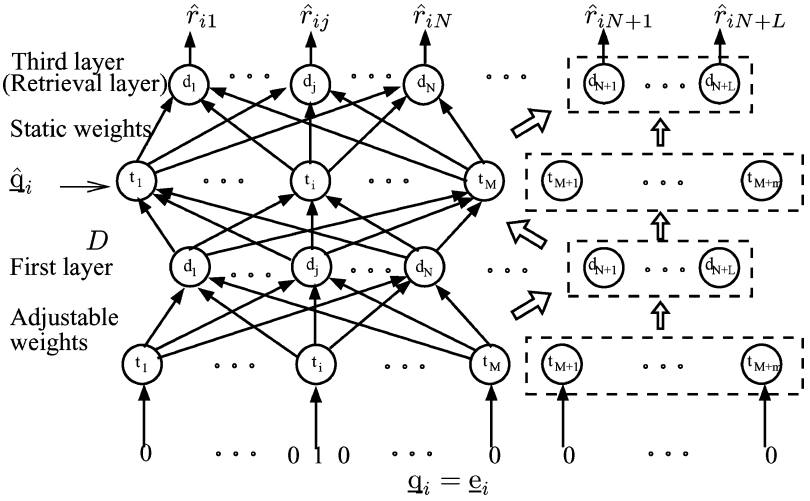


Fig. 3. Network after the insertion of L new documents.

attribute for term t_i will have the highest retrieved score. Although these scores are not directly representative of relevancy, subsequent learning based upon users feedback for every query will gradually improve the relevancy scores of the documents. This will be shown in Section VI-C.

Remark III.2: If two document vectors are identical, the Gram matrix A_0 becomes singular. This situation can be avoided, if for each document its ID, which is unique to each document, is also added as a representative term. If α is the weight assigned to the term that represents the document name, and $\underline{\mathrm{d}}'_i$ is the document vector after augmenting the original document vector $\underline{\mathrm{d}}_i$ with the document name, then we have

$$\underline{\mathrm{d}}'^T_i \underline{\mathrm{d}}'_j = \underline{\mathrm{d}}^t_i \underline{\mathrm{d}}_j, \qquad i \neq j \text{ and } \forall i, j \in [1, N] \tag{11}$$

$$\underline{\mathrm{d}}'^T_i \underline{\mathrm{d}}'_i = \underline{\mathrm{d}}^t_i \underline{\mathrm{d}}_i + \alpha^2, \qquad i \in [1, N]. \tag{12}$$

Therefore, the new Gram matrix is $A'_0 = A_0 + \alpha^2 I$, which is a regularized version of A_0. In the sequel, it is assumed that the Gram matrix is regularized when needed. Thus, for simplicity in notation, the superscript "$'$" is dropped.

IV. Model-Reference Following

Once the initial model-reference learning is completed and the system in the feedforward path of Fig. 1 captures the underlying input–output relationship of a reference model (or an indexing system), it is crucial that the regulator adapts to the changes in the model or the environment (database). These changes can be brought about as a result of document reindexing, adding new documents that may contain new terms, and/or deleting the obsolete ones. The key requirement is that these changes must be incorporated into the system without impacting the performance or sacrificing the stability of the previously established learning.

This model-reference following can be accomplished efficiently in the regulator of the proposed system using either an online recursive or a batch learning scheme. As in the initial learning phase, the upper loop in Fig. 1 together with an appropriate adjustment mechanism is used in this phase. In the proposed three-layer network, these changes can easily be implemented via structural and weight adaptation mechanisms. Structural adaptation involves node addition and deletion in the network layers. Sections IV-A–IV-C describe the adaptation rules for these scenarios for the case where the learning is carried out in regression mode. Note that the results in Section III can be used to derive similar adaptation rules for the classification mode of model-reference following.

A. Document Addition

To incorporate L new documents into the three-layer network, additional nodes with new connection weights must be added while the old weights of the network are updated. Let $\underline{\mathrm{d}}_{(N+1)}$, $\underline{\mathrm{d}}_{(N+2)}$, and $\underline{\mathrm{d}}_{(N+L)}$ be the new document vectors to be added into the system, and m be the total number of new terms introduced as a result of adding these documents. To accommodate these documents, L new nodes must be inserted into the first and third (output) layers. The weight vectors corresponding to connections emanating from the added nodes in the first hidden layer and the weight vectors for the incoming connections to the added nodes in the third layer are $\underline{\mathrm{d}}_{N+l}$, $\forall l \in [1, L]$. Additionally, to account for the addition of m newly introduced terms, m new nodes must be added in the second hidden layer as well as in the input. The network has to be updated in such a way that even after the insertion of the new documents with the corresponding new terms, the system retains the previous training. Fig. 3 shows the network after the insertion of the new documents and terms. The newly added nodes are shown inside the dashed boxes. For all of the already existing terms, i.e., t_i, $i \in [1, M]$, the old connections weights w_{ij}, $j \in [1, N]$ have to be updated; while the new connection weights $w_{i(N+l)}$, $\forall l \in [1, L]$ have to be computed. For those newly introduced terms t_i, $i \in [M+1, M+m]$, all the connection weights must be computed as well.

Assume that the system is initially trained using N documents and that data matrix D contains these documents. Let D_L be the data matrix of the L newly introduced document vectors. These new documents are added to the column space of matrix

D. Now, using (2), the optimal query for the single-term query t_i in this augmented space is given by

$$\underline{\hat{q}}_i^{(1)} = \sum_{j=1}^{N+L} w_{ij}^{(1)} \underline{d}_j. \tag{13}$$

The desired score vector $\underline{\hat{r}}_{iL} = [\hat{r}_{iN+1}^{(1)} \dots \hat{r}_{iN+L}^{(1)}]^T$ of the L newly added documents is assumed to be available from the reference model. Since the score of the already existing documents should not change, the score vector of all the documents for the mapped query in (13) is $\underline{\hat{r}}_i^{(1)} = [\underline{\hat{r}}_i^{(0)^T} \underline{\hat{r}}_{iL}^T]^T$. We redefine $\underline{w}_i^{(1)} = [w_{i1}^{(1)}, w_{i2}^{(1)}, \dots w_{iN+L}^{(1)}]^T$ as the new weight vector after inserting these L new documents. Clearly, we still have $\underline{w}_i^{(1)} = A_1^{-1} \underline{\hat{r}}_i^{(1)}$ where matrix A_1 is given as

$$A_1 = \begin{pmatrix} A_0 & B \\ B^T & C \end{pmatrix}. \tag{14}$$

Here, A_0 is the same as before, $B = D^T D_L$ is a matrix of dot products between old and new documents, and $C = D_L^T D_L$ is a matrix formed with the dot products of the new documents only. If the inverse of matrix A_1 is expressed as

$$A_1^{-1} = \begin{pmatrix} F & V \\ V^T & U \end{pmatrix} \tag{15}$$

where

$$\begin{aligned} U &= (C - B^T A_0^{-1} B)^{-1} \\ V &= -A_0^{-1} B U \\ F &= A_0^{-1}(I - B V^T) \end{aligned} \tag{16}$$

then, using the expansion for matrix A_1^{-1}, the updating equation for the weight vector $\underline{w}_i^{(1)}$ can be given as

$$\underline{w}_i^{(1)} = \begin{pmatrix} \underline{w}_i^{(0)} + V\left(\underline{\hat{r}}_{iL} - B^T \underline{w}_i^{(0)}\right) \\ U\left(\underline{\hat{r}}_{iL} - B^T \underline{w}_i^{(0)}\right) \end{pmatrix}. \tag{17}$$

This weight update equation is performed for all nodes $i \in [1, M]$ or single-term queries in the input layer. The first and second parts of the vector in the right-hand side of (17) correspond to updating the old weights and computing the new added weights, respectively. For those newly introduced terms, since $\underline{w}_i^{(0)} = \underline{0}, \forall i \in [M+1, M+m]$, we have

$$\underline{w}_i^{(1)} = \begin{pmatrix} V \underline{\hat{r}}_{iL} \\ U \underline{\hat{r}}_{iL} \end{pmatrix}. \tag{18}$$

From (17) and (18), it can be seen that only U and V are needed to update the weights. Computing U involves inverting a matrix of dimension L where $L \ll N$.

This batch learning reduces to an iterative online learning if for every time that a new document is added the weight updating is performed. This leads to the following recursive equations:

$$\underline{w}_i^{(1)} = \begin{pmatrix} \underline{w}_i^{(0)} - \beta^{-2} A_0^{-1} \underline{b}\left(\hat{r}_{i(N+1)} - \underline{b}^T \underline{w}_i^{(0)}\right) \\ \beta^{-2}\left(\hat{r}_{i(N+1)} - \underline{b}^T \underline{w}_i^{(0)}\right) \end{pmatrix} \quad \forall i \in [1, M] \tag{19}$$

and

$$\underline{w}_i^{(1)} = \begin{pmatrix} -\beta^{-2} A_0^{-1} \underline{b} \hat{r}_{i(N+1)} \\ \beta^{-2} \hat{r}_{i(N+1)} \end{pmatrix} \qquad \forall i \in [M+1, M+m] \tag{20}$$

where A_1 matrix, in this case, is given by

$$A_1 = \begin{pmatrix} A_0 & \underline{b} \\ \underline{b}^T & c \end{pmatrix} \tag{21}$$

matrix A_0 was defined before with $\underline{b} = D^T \underline{d}_{(N+1)}$, and $c = \underline{d}_{(N+1)}^T \underline{d}_{(N+1)}$. Also, it can easily be shown that

$$\beta^2 = \underline{d}_{(N+1)}^T P_D^{\perp} \underline{d}_{(N+1)} \tag{22}$$

where $P_D^{\perp} = I - P_D$ and the projection matrix P_D was defined before.

B. Document Deletion

When multiple (L) documents are to be simultaneously removed, they could be at any position within the network structure. To facilitate the deletion of these documents (or nodes), we shift them to the last L columns of matrix D. To accomplish this, let A_0' be the Gram matrix after shifting the document vectors to be deleted to the far right side of the data matrix D. Then, A_0' can be defined in terms of the original Gram matrix A_0 by permutations where we have $A_0' = (DP_1P_2 \dots P_L)^T (DP_1P_2 \dots P_L) = (P_1P_2 \dots P_L)^T A_0 (P_1P_2 \dots P_L)$, where P_i's, $i \in [1, L]$ are the appropriate permutation matrices. Each permutation matrix P_i is formed such that it shifts a particular document vector to be deleted to the far right side of the data matrix so that it can easily be removed. Additionally, we partition the weight and score vectors such that the connection weights corresponding to the documents to be deleted are at their lower end, hence we have $\underline{w}_i'^{(0)} = [\underline{w}_{i(N-L)}'^{(0)^T} \underline{w}_{iL}'^{(0)^T}]^T$, and similarly for the score vector, $\underline{\hat{r}}_i' = [\underline{\hat{r}}_{i(N-L)}^T \underline{\hat{r}}_{iL}^T]^T$. Thus, the weight vector solution can be written as

$$\underline{w}_i'^{(0)} = A_0'^{-1} \underline{\hat{r}}_i'. \tag{23}$$

Rewriting A_0' in the block matrix form, we have

$$A_0' = \begin{pmatrix} A_1 & B' \\ B'^T & C' \end{pmatrix}. \tag{24}$$

The inverse of matrix A_0' can be computed using $A_0'^{-1} = P^T A_0^{-1} P$. Now, rewriting $A_0'^{-1}$ in the block matrix form

$$A_0'^{-1} = \begin{pmatrix} F_0 & V_0 \\ V_0^T & U_0 \end{pmatrix} \tag{25}$$

and using (25) gives

$$\begin{pmatrix} \underline{w}_{i(N-L)}'^{(0)} \\ \underline{w}_{iL}'^{(0)} \end{pmatrix} = \begin{pmatrix} F_0 \underline{\hat{r}}_{i(N-L)} + V_0 \underline{\hat{r}}_{iL} \\ V_0^T \underline{\hat{r}}_{i(N-L)} + U_0 \underline{\hat{r}}_{iL} \end{pmatrix} \tag{26}$$

where F_0, U_0, and V_0 are defined as before. Now, the solution for the connection weights of the network after deleting L documents can be written as

$$\underline{\mathrm{w}}_{i(N-L)}^{(1)} = A_1^{-1}\hat{\underline{\mathrm{r}}}_{i(N-L)}. \tag{27}$$

Note that (27) is obtained by deleting the last L columns and L rows of matrix A_0' in (24) and then solving for $\underline{\mathrm{w}}_{i(N-L)}^{(1)}$. From (24) and (25), we can express $A_1^{-1} = F_0(I - B'V_0^T)^{-1}$. Using the matrix inversion lemma [29], we have

$$A_1^{-1} = F_0\left(I + B'\left(I - V_0^T B'\right)^{-1} V_0^T\right) = F_0 - V_0 U_0^{-1} V_0^T. \tag{28}$$

Substituting for A_1^{-1} in (27) yields

$$\underline{\mathrm{w}}_{i(N-L)}^{(1)} = F_0\hat{\underline{\mathrm{r}}}_{i(N-L)} - V_0 U_0^{-1} V_0^T \hat{\underline{\mathrm{r}}}_{i(N-L)}. \tag{29}$$

Using (26), the updating equation for document deletion becomes

$$\underline{\mathrm{w}}_{i(N-L)}^{(1)} = \underline{\mathrm{w}}_{i(N-L)}^{\prime(0)} - V_0 U_0^{-1} \underline{\mathrm{w}}_{iL}^{\prime(0)} \tag{30}$$

where weight vector $\underline{\mathrm{w}}_{i(N-L)}^{(1)}$ after the deletion is expressed in terms of the weight vectors $\underline{\mathrm{w}}_{i(N-L)}^{\prime(0)}$ and $\underline{\mathrm{w}}_{iL}^{\prime(0)}$ before the deletion. In (30), the inverse of matrix U_0 of dimension $L \ll N$ needs to be computed.

C. Document Updating

When a document is reindexed (or updated) care must be taken to include it into the system because of possibly new document terms that are introduced as a result of the reindexing. Although the weights in the second and third layers can easily be updated, additional steps should be carried out to modify the first layer connections in order to retain the previously stored information. Suppose that document $\underline{\mathrm{d}}_j$ is updated and also new terms are introduced. Let T^0 be the set of remaining terms in the system after the document to be updated is removed from the system. The process of finding the weights can be accomplished in two steps, where the document to be updated is first removed and then added into the system with new attributes and terms. However, the equations for removal and addition of a document can be simplified using a sequential updating. If document j is to be updated and $\underline{\mathrm{w}}_{i(N-1)} = [w_{i1} \ldots w_{ij-1} w_{ij+1} \ldots w_{iN}]^T$ represents the weight vector emanating from term $t_i \in T^0$ in the input to the remaining documents, i.e., excluding the jth one, then using (30) with $L = 1$ the weight vector $w_{i(N-1)}^{(1)}$ after deletion will be

$$\underline{\mathrm{w}}_{i(N-1)}^{(1)} = \underline{\mathrm{w}}_{i(N-1)}^{(0)} - u_0^{-1}\underline{\mathrm{v}}_0 w_{ij}^{(0)} \qquad \forall t_i \in T^0 \tag{31}$$

where $w_{i1}^{(0)} \rightarrow w_{ij}^{(0)}$ represents the weight connection of the document to be removed (i.e., document j), $U_0 \rightarrow u_0$ is a scalar and $V_0 \rightarrow \underline{\mathrm{v}}_0$ is a column vector. Now, we use (17) to add the updated document along with its new terms and its new score $\hat{r}_{ij}$. In this case, $V \rightarrow \underline{\mathrm{v}}_1$ and $B \rightarrow \underline{\mathrm{b}}_1$ will be column vectors and $U \rightarrow u_1$ is a scalar. Therefore, we have

$$\underline{\mathrm{w}}_i^{(2)} = \begin{pmatrix} \delta(i)\underline{\mathrm{w}}_{i(N-1)}^{(1)} + \underline{\mathrm{v}}_1\left(\hat{r}_{ij} - \delta(i)\underline{\mathrm{b}}_1^T \underline{\mathrm{w}}_{i(N-1)}^{(1)}\right) \\ u_1\left(\hat{r}_{ij} - \delta(i)\underline{\mathrm{b}}_1{}^T \underline{\mathrm{w}}_{i(N-1)}^{(1)}\right) \end{pmatrix} \tag{32}$$

where

$$\delta(i) = \begin{cases} 1, & \text{if } t_i \in T^0 \\ 0, & \text{otherwise} \end{cases}. \tag{33}$$

Note that $\delta(i)$ is needed to account for the remaining terms after document j is deleted. Combining (31) and (32), the final equations to modify the weights of the system, when document j is updated, are shown in (34), at the bottom of the page, where the last element of $\underline{\mathrm{w}}_i^{(2)}$ is the weight between term t_i and document j.

V. Relevance Feedback Learning

Often the original submitted query does not meet the specific user requirements in terms of the listed documents, their relevancy, or relative scores. To meet the expert users' requirements and at the same time preserve the previous learning, in our proposed MRTRS, the relevance feedback information can be incorporated using two possible mechanisms, depending on the nature of the user feedback. This can be accomplished by updating the parameters of the regulator or the weights of the first layer. In the MRTRS framework, the lower feedback loop (relevance feedback loop) of the system in Fig. 1 provides expert users' votes on relevant and nonrelevant documents to the adjustment mechanism, which in turn updates the parameters of the regulator to meet the users' requirements by imposing relevance feedback. The user may provide relevance feedback to the adjustment mechanism either by assigning desired scores to the most relevant document(s) that he/she selects or simply by click-through selection. These relevance feedback types can be implemented using either regression or classification-based learning. Clearly, this phase of learning captures certain user-based information and expertise that cannot be learned from the reference model alone.

A. Regression Mode

As in Section III, the problem of relevance feedback learning can be cast in a constrained optimization framework. In this case, the main objective is to transform the previously mapped query (obtained in phase 1), to a new optimal query $\hat{\underline{\mathrm{q}}}_i$ that is

$$\underline{\mathrm{w}}_i^{(2)} = \begin{pmatrix} \delta(i)\underline{\mathrm{w}}_{i(N-1)}^{(0)} - \delta(i)\underline{\mathrm{v}}_1\underline{\mathrm{b}}_1^T\underline{\mathrm{w}}_{i(N-1)}^{(0)} - \delta(i)u_0^{-1}\underline{\mathrm{v}}_0 w_{ij}^{(0)} + \delta(i)u_0^{-1}\underline{\mathrm{v}}_1\underline{\mathrm{b}}_1^T\underline{\mathrm{v}}_0 w_{ij}^{(0)} + \underline{\mathrm{v}}_1\hat{r}_{ij} \\ u_1\hat{r}_{ij} - \delta(i)u_1\underline{\mathrm{b}}_1^T w_{i(N-1)}^{(0)} + \delta(i)u_0^{-1}u_1\underline{\mathrm{b}}_1^T\underline{\mathrm{v}}_0 w_{ij}^{(0)} \end{pmatrix} \tag{34}$$

the closest to the old one (in the Euclidian norm) and further satisfies the new constraint $D^T\underline{\hat{q}}_i = \underline{\hat{r}}_i$. Thus, the Lagrangian function in this regression-based learning should be modified to

$$J(\underline{\hat{q}}_i, \Delta\underline{w}_i) = \frac{1}{2}(\underline{\hat{q}}_i - \underline{q}_i)^T(\underline{\hat{q}}_i - \underline{q}_i) + \sum_{j=1}^{N} \Delta w_{ij}\left(\hat{r}_{ij} - \underline{d}_j^T\underline{\hat{q}}_i\right) \tag{35}$$

where Δw_{ij} represents the incremental change in the Lagrangian multiplier. Then, the LS solution for $\underline{\hat{q}}_i$ becomes

$$\underline{\hat{q}}_i = \underline{q}_i + \sum_{j=1}^{N} \Delta w_{ij}\underline{d}_j = \underline{q}_i + D\Delta\underline{w}_i \tag{36}$$

and the optimal solution for $\Delta\underline{w}_i$ is

$$\Delta\underline{w}_i = (D^TD)^{-1}\underline{\delta}_i \tag{37}$$

where $\underline{\delta}_i = \underline{\hat{r}}_i - \underline{r}_i$ with $D^T\underline{q}_i = \underline{r}_i$, i.e., score vector for the old query. Thus, the weight vector $\underline{w}_i$ in the first layer can be updated to $\underline{\hat{w}}_i$ using $\underline{\hat{w}}_i = \underline{w}_i + \Delta\underline{w}_i$, where $\underline{\hat{w}}_i = (D^TD)^{-1}\underline{\hat{r}}_i$ meets the new score requirements $\underline{\hat{r}}_i$.

If there are K voted documents whose scores need to be modified in the regression mode, the new score vector becomes

$$\underline{\hat{r}}_i = \underline{r}_i + \sum_{k\in S} \underline{e}_k \Delta r_k \tag{38}$$

where $\underline{e}_k$ is as defined before, Δr_k is the incremental change corresponding to user-specified score of the voted documents $\underline{d}_k$, and S represents the set of voted documents with cardinality $|S| = K$. In this case, the parameters of the regulator or the weights of the first layer must be updated using the weight increment vector

$$\Delta\underline{w}_i = \sum_{k\in S} \underline{b}_k \Delta r_k \tag{39}$$

where $\underline{b}_k = A_0^{-1}\underline{e}_k$ is the kth column of the matrix $A_0^{-1} = (D^TD)^{-1}$.

From (39), it can be observed that document voting corresponds to linearly adjusting the weights to incorporate user feedback. Since this weight updating is only implemented for the weights associated with the query term t_i, it is ensured that information learned in the previous learning is not lost while at the same time allowing for adaptation of new associations.

Remark V.1: The relevance feedback learning in this section can be implemented either online for every user (not typical) or in batch mode after collecting all the users' votes on various queries and forming a log file. In the former case, the score for a click-through selection can be specified internally using a particular scoring scheme [19] and without any user involvement. In the latter case, frequency of votes, expertise level of the user, date in which voting takes place, date in which the document is last modified, or any other meaningful criterion can be used in conjunction with some specific heuristic rules to arrive at the desired score vector $\underline{\hat{r}}_i$ for every query in the log file. This is used in Section VI, though the same updating rules can also be applied for online iterative-based learning.

Remark V.2: Although during the initial model-reference learning (phase 1) the weights of the first layer of Fig. 2 are computed for all the single-term queries, due to the linearity of the network, one can perform weight adaptation during relevance feedback for multiterm queries as well. For instance, if a 2-term query containing terms t_i and t_j is applied, the weights associated with these terms, i.e., $\underline{w}_i$ and $\underline{w}_j$ will undergo adaptation. Since the global corpus weights of these terms are known, the contribution of each term toward the desired response will be determined based upon these weights. However, this fine-tuning may slightly change the response for other queries that contain the same terms. To remedy this problem, a new learning algorithm is developed in Appendix A, which starts from the indexing results, i.e., projection matrix P_D for the initial training of the regulator, and then applies relevance feedback learning based upon a set of single- and multiterm queries and their associated votes in the log file. Section VI-C gives the results of this method and their benchmarking with the original single-term relevance feedback learning method.

Remark V.3: An alternative relevance feedback mechanism [30] that provides flexibility in the position of the documents (typical in general-purpose TRS) can be devised using our network structure. In this scheme, the terms in the voted documents are modified via updating the corresponding document terms in the retrieval layer in order to elevate the relevant documents while demoting the nonrelevant ones. The learning is based upon Gram–Schmidt orthogonalization in conjunction with a node creation strategy to incorporate the user feedback.

B. Classification Mode

If the user identifies the most relevant document(s) via click-through selection, he/she may not be interested in assigning scores, rather is content with specifying whether a document is relevant to a particular query or not. Then, the new score vector can still be found using (38) for this classification-based learning (two class) with the minor difference that $\Delta r_k = \pm 2$, where $+$ is used when the kth document status should be changed from nonrelevant to relevant while $-$ is used when the status should be changed from relevant to nonrelevant. Additionally, the new diagonal matrix $\widehat{\Lambda}_i$ should be changed to

$$\widehat{\Lambda}_i = \Lambda_i + \sum_{k\in S} \underline{e}_k\underline{e}_k^T \Delta r_k \tag{40}$$

in order to satisfy the requirement $\widehat{\Lambda}_i\underline{\hat{r}}_i = \underline{1}$. With these minor modifications, the weight increment $\Delta\underline{w}_i$ becomes

$$\Delta\underline{w}_i = \widehat{\Lambda}_i \sum_{k\in S} \underline{b}_k \Delta r_k + \sum_{k\in S} \underline{e}_k\underline{b}_k^T \Delta r_k \underline{r}_i \tag{41}$$

where $\underline{b}_k$ is defined as before.

VI. Test Results

The main problem considered here is to drastically reduce service call duration received by call center agents responding to customers' issues in a knowledge management system. The proposed system is used to capture and incorporate the expertise

of certain subject matter experts to help and expedite the resolution rates for subsequent searches automatically and without any manual or laborious operations.

The entire knowledge base for the TRS consists of several major collections of about 75 000 documents and about 130 000 distinct terms. These collections provide a wide range of information for support, diagnostics, and specifications for consumers and commercial suites of HP products. The documents contain both unstructured and structured information on various product types. The majority of content is represented in text format, while the collections also contain a mixed graphical and multimedia formats. The number of searches or query sessions received from users within the United States and abroad can vary in the range of 650 000–720 000 searches per month, while the number of relevance feedback is less than 1000 per month. The search strings are typical single- or multiterm queries (average query length of 2.5 terms) that are used in daily conversations to respond to various support calls to diagnose and resolve customer's questions or issues in real-life production environment. Some limited examples of such queries can be found in Tables II and III. The major collections are based on product types. The results presented in this paper are obtained based upon 17 different product collections containing over 32 000 documents and about 108 000 distinct terms.

The document survey feature available within the TRS helps users to log their feedback on one or more documents for the query submitted. The voting process is influenced by the users' perception on the desired solution documents. The influence comes in the form of a vote, either a positive or a negative one, that the user assigns to a particular document. To incorporate user expertise level and additional dynamics into the retrieval system, the vote is weighted by a factor that depends on the users' expertise level and the elapsed time since the document was last created or modified. Using the log file of queries along with their respective feedback information, we recreate the voting process used by the users to generate a set of prototype pairs $(\underline{\mathrm{q}}, \underline{\hat{\mathrm{r}}})$s with the goal of training our MRTRS. The relevance feedback learning can be applied either iteratively or in the batch mode. It is important to mention that although a large number of queries were available, only a subset of approximately 5900 most commonly used queries were used to form the prototypes. The prototype query set consists of 2386 1-term, 1664 2-term, and 1846 3^{+}-term queries.

To assess the performance of the learning algorithms, the rank order correlation measure based on Kendall's τ [21] is used. This nonparametric measure is useful when the underlying distribution that generates the scores cannot be easily estimated. The performance measure based on Kendall's τ compares two ranked or unranked lists and generates a coefficient $(\tau \in [-1, 1])$ that represents the closeness of the two lists. The coefficient τ [21] is given by

$$\tau = \frac{P - Q}{\frac{1}{2}N(N-1)} \tag{42}$$

where P and Q are the number of concordant and discordant pairs, respectively. The denominator in (42) is the number of combinations of taking two elements out of N and is equal to the sum of the concordant pairs P and discordant pairs Q found in the two lists when they do not contain ties. However, when there are ties, the measure in (42) is not appropriate. In this case, a modification can be made to yield Kendall's τ_b as defined by

$$\tau_b = \frac{P - Q}{\sqrt{\left(\frac{1}{2}N(N-1) - T_1\right)\left(\frac{1}{2}N(N-1) - T_2\right)}} \tag{43}$$

where T_1 and T_2 are the number of tied pairs for lists 1 and 2, respectively. Clearly, (43) reduces to (42) when there are no ties.

In the sequel, the testing of the algorithms is performed in the three different phases described in Sections III–V. In all these phases, the results are evaluated by comparing them to the "benchmark," which consists of the reference model TRS results augmented or enhanced by several votes received from different tier expert users collected in the log file. The Kendall's τ measure is then generated in each case to determine how closely the learning in different phases can reach the desired benchmark. Additionally, the standard performance metric "recall," which presents the ability of the TRS to retrieve all the relevant documents for a given query is used. More specifically, recall is the ratio of the number of relevant documents retrieved to the total number of relevant documents. To compute the recall performance measure for the same "benchmark," the recall values for different queries in the log file are averaged to yield one point on the recall curve.

The initial model-reference learning is also compared against the advanced BM25 IR method. The BM25 document scoring function [26] is

$$\sum_{T \in Q} w^{(1)} \frac{(k_1+1)\mathrm{tf}}{(K+\mathrm{tf})} \frac{(k_3+1)\mathrm{qtf}}{(k_3+\mathrm{qtf})} + k_2|Q|\frac{\mathrm{avdl}-\mathrm{dl}}{\mathrm{avdl}+\mathrm{dl}} \tag{44}$$

where the Robertson-Sparck–Jones weighting function $w^{(1)}$ is given by

$$w^{(1)} = \log\left(\frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)}\right). \tag{45}$$

Here, Q is the query containing the term T, N is the total number of documents in the collection, n is the total number of documents containing the term T, R is the number of documents known to be relevant to a specific query, r is the number of relevant documents containing the term T, dl is the document length, avdl is the average document length, $K = k_1((1-b) + b\ \mathrm{dl}/\mathrm{avdl})$, tf is the term frequency within a specific document, qtf is the term frequency within the query Q, and k_1, k_2, k_3, and b are some prespecified constants. The values of these constants in our experiments are chosen to be $k_1 = 2.0$, $k_2 = 0$, $k_3 = 8.0$, and $b = 0.75$.

Sections VI-A–VI-D describe the details of the experiments, the results, and observations.

A. Initial Model-Reference Learning—Phase I

As pointed out before, the goal of the initial model-reference learning is to capture the input–output behavior of a reference TRS or to use the results of the indexing system in the absence of a model TRS. Thus, four scenarios are considered here: the first and second use the response of the model TRS to all the single-term queries in the log file to set up the weights of the network (initial training) through the regression or classification

TABLE I
PERFORMANCE MEASURE, τ_b, FOR DIFFERENT LEARNING MODES

Learning Modes	1-Term	2-Terms	3+ Terms
Regression mode	0.935	0.892	0.935
Classification mode	0.927	0.880	0.918
Indexer mode	0.613	0.033	-0.123
BM25	0.634	0.219	0.063

learning modes, while the third one uses the projection matrix P_D, i.e., the indexing results, and the fourth one uses document scoring generated using (44) and (45). The regression mode is useful when the scores of the documents are available from the TRS, while the classification mode is used when a set of relevant document(s) for each submitted query in the training set is known. However, the initial training using the indexer solely relies on documents' terms and their term weights and not on any querying results.

The networks are trained in the batch mode and the weights of the networks are obtained using (3), (6), and the projection matrix P_D for the regression, classification, and indexer-based learning modes, respectively. The Kendall's τ_b measure is then generated based on the top 20 documents listed by the initially trained networks evaluated against the "benchmark," i.e., the results of the TRS augmented by the users' votes. The purpose is to determine how close each initially trained system can approach the ultimate benchmark. The results are obtained for the most commonly used 1-, 2-, and 3^+-term queries in the log file. Table I gives the values of τ_b for these most commonly used prototype queries. In the regression and classification learning modes, the value of τ_b gets close to 1 (perfect match) for all the single- and multiterm queries. The results obtained based upon regression mode learning, however, are slightly better than those of the classification mode learning. This is due to the fact that unlike the regression mode, in the classification mode, document scores are ignored and only their binary relevance information is considered. However, the high value of τ_b indicates that the learning through both the regression and classification modes is able to capture the reference model behavior very closely. The initial learning results obtained based upon the document indexer indicate that even though for single-term queries the value of τ_b is reasonable, the corresponding values for the multiterm queries become unacceptable. This is due to the fact that in this case the querying information and the response of the reference model are not used in the initial training. Nonetheless, we will show that even with this crude initial training, the proposed relevance feedback learning leads to excellent final results. The results of the BM25 scoring function in (44) and (45) are given in the last row of Table I. Although, these results are slightly better than those generated based upon the document indexer without any querying and scoring process, they are certainly worse than those of the regression and classification learning modes proposed in this paper. This is mainly due to fact that the regression and classification learning modes are specifically designed to capture the behavior of the model-reference TRS.

Next, the recall capability of the system initially trained using three different initial learning modes is evaluated. Fig. 4(a)–(c) shows the recall plots for various thresholds of the retrieved list for the regression, classification, and indexer-based learning modes, respectively, for the most commonly used 1-, 2-, and 3^+-term queries in the log file. Fig. 4(d) shows the same plots for the BM25 algorithm. As can be observed from the plots in Fig. 4, the results obtained based on the regression mode are significantly better than those of the other three modes. The results of the classification mode learning are inferior as this method only uses the binary relevance information of the top 20 documents without considering the scores of the relevant documents. The results of the indexer-based learning, which requires the least amount of prior information (no reference TRS model and document scores), are better than those of the classification mode and very comparable to those of the BM25 method. Thus, in Section VI-C, this system is used as the initially trained system for the subsequent learning via relevance feedback.

B. Model-Reference Following

The model-reference following developed in Section IV for learning in dynamic environments is tested in this section when new documents with new terms are introduced or the obsolete ones are removed from the initially trained system. The experiments in this section are performed on only one collection consisting of about 3700 documents and the results are obtained for 800, 724, and 1037 1-, 2-, and 3^+-term queries, respectively. Two experiments are conducted here. In the first experiment, 500 new documents with new terms are introduced into the initially trained system in steps of 20. The algorithm in Section IV was then applied to update the weights of the initially trained network using the regression-based learning. Every time a group of 20 new documents are added, the value of τ_b is computed for the most commonly used 1-, 2-, and 3^+-term queries. Fig. 5(a) shows the plots of the averaged τ_b for these three cases during the course of the model-reference following. Again, the Kendall's τ_b measure is generated based on the top 20 documents listed by the updated system against the benchmark. Note that, in this phase, the benchmark corresponds to the log file of users' votes for the enlarged document set that included the additional 500 documents. As can be seen, for the 1-, 2-, and 3^+-term queries the values of τ_b increased from 0.714, 0.659, and 0.792 to 0.93, 0.89, and 0.93, respectively. This increasing trend of τ_b in Fig. 5(a) implies that the retrieved results of the system after adding the new documents approach those of the initially trained system based upon the enlarged document set. It is interesting to note that the retrieved results for the 3^+-term queries are much closer to the benchmark. This is also consistent with the results of phase 1 study in Table I. This may be attributed to the fact that incorporating more than three relevant terms in queries closely captures the user concepts and requirements.

Another experiment was conducted when 500 documents were deleted from the initially trained system. The benchmark for this paper was the log file of the users' votes and TRS results for the reduced set of documents that excluded the 500 documents. Documents were deleted in steps of 20 and the value of τ_b was generated at every step for the most commonly used 1-,

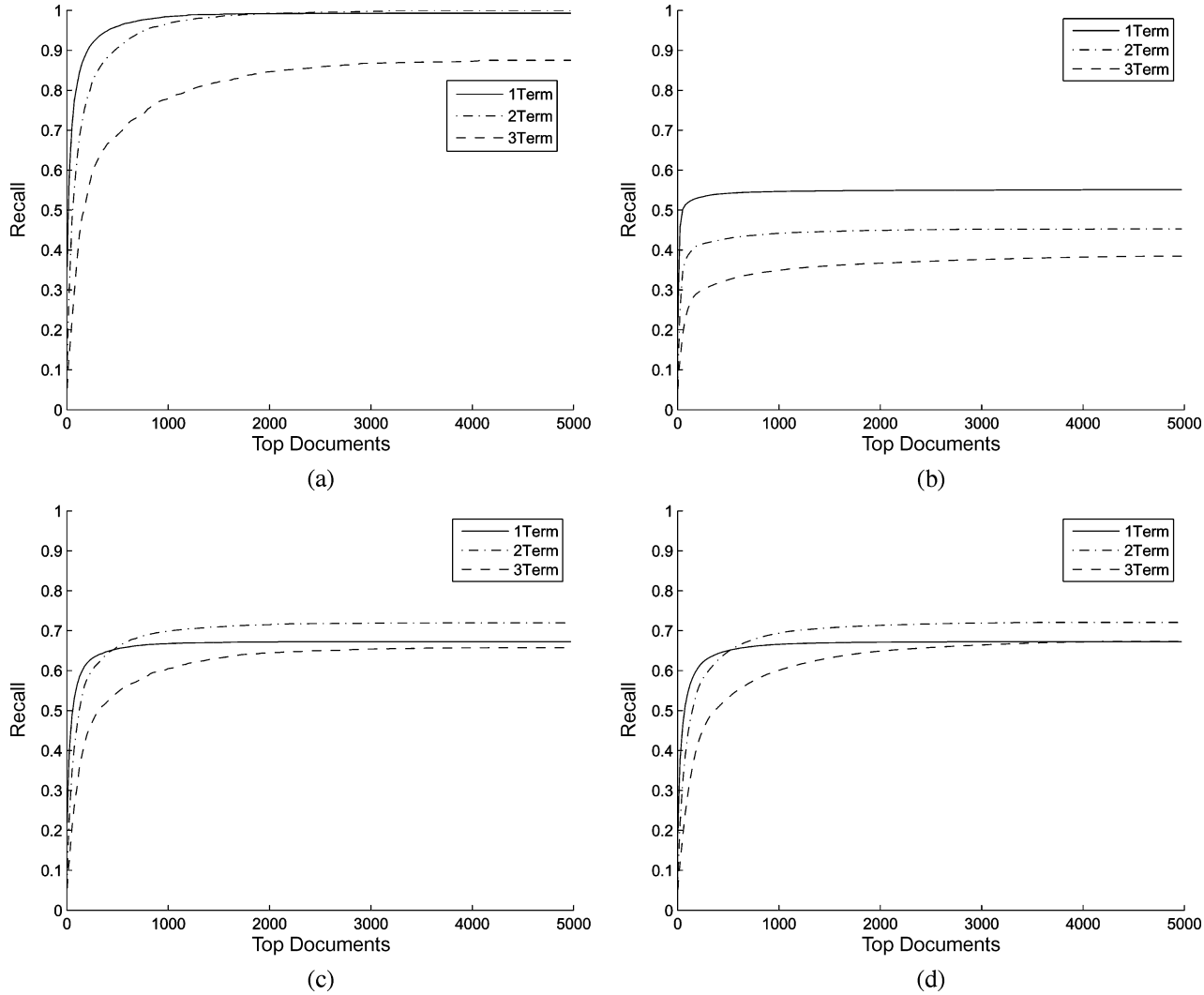


Fig. 4. Recall plots for learning in (a) regression mode, (b) classification mode, (c) indexer mode, and (d) BM25 method.

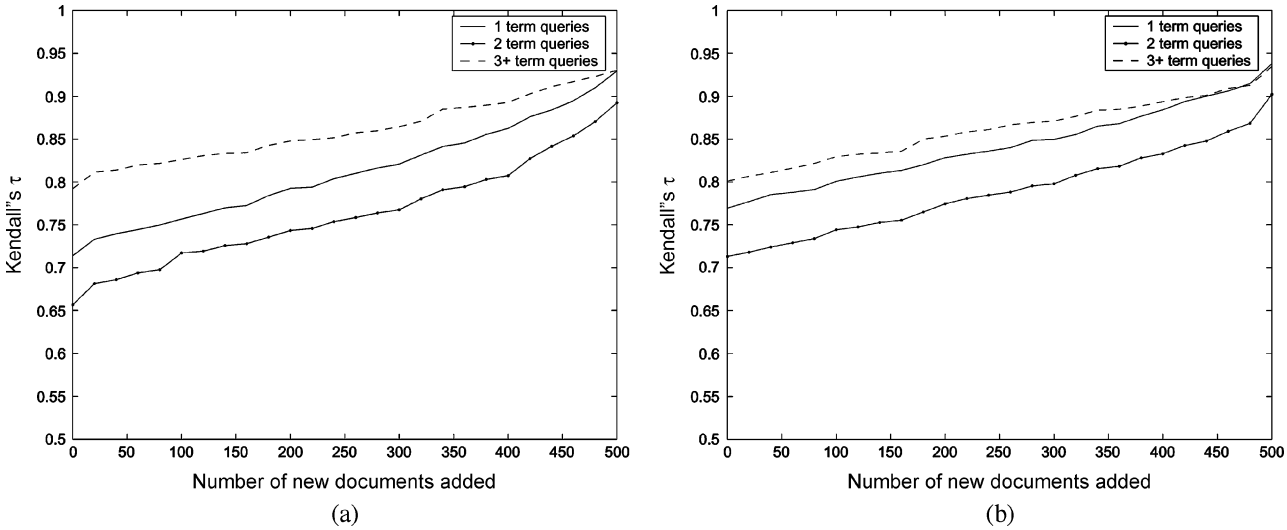


Fig. 5. Model-reference following: (a) τ_b when 500 new documents are added and (b) τ_b 500 when documents are deleted.

2-, and 3^+-term queries in the log file. Fig. 5(b) shows the plots of the averaged τ_b for these queries. As can be observed from these plots, the value of τ_b for these queries increased from 0.77, 0.71, and 0.80 to 0.938, 0.902, and 0.935, respectively. Again, these results attest to the fact that the system after the model-reference following closely captures the underlying dynamic behavior of the model when documents are added or deleted.

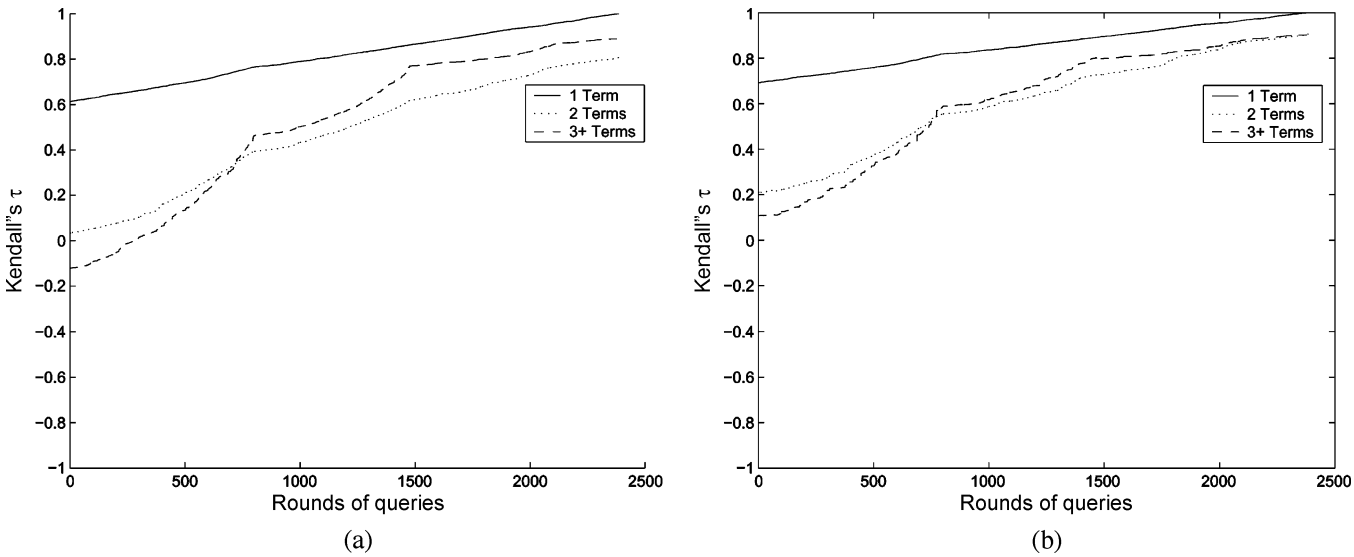


Fig. 6. Relevance feedback learning: (a) weight update in regression mode and (b) weight update in classification mode.

C. Relevance Feedback Learning

The goal of the relevance feedback is to promote/demote one or more documents to the required positions based upon the users' votes and their characteristics such as expertise level of the users, frequency of votes, and document publish dates. Although the relevance feedback can be implemented either in an iterative online mode for every expert user, or in a batch mode based upon votes collected over certain period of time, in this section, the effectiveness of the proposed algorithms is demonstrated for the batch mode relevance feedback based upon the votes collected in the log file for single- and multiterm queries. It will also be shown that voting for 1-term queries improves the accuracy of the retrieval system for multiterm queries.

1) Training Based Upon Single-Term Queries: In this paper, the votes collected in the log file for 1-term queries are used for relevance feedback training and the updated system is then evaluated on the multiterm (testing) queries. The system was initially trained based on the document indexer results. Relevance feedback learning is then performed in both the regression and classification modes by incrementally adjusting the weights of the first layer based on (39) and (41). After the relevant feedback learning for every 1-term query is performed, the Kendall's τ_b is computed for the training and testing queries. Fig. 6(a) and (b) shows the plots of τ_b for the entire 2386 iterations of relevance feedback for these two learning modes. In the regression mode, the initial values of Kendall's τ_b were 0.611, 0.026, and -0.124, while the final values after relevance feedback learning became 1, 0.805, and 0.89 for 1-, 2-, and 3^+-term queries, respectively. Similarly, in the classification mode, initial values of τ_b were 0.692, 0.210, and 0.109, whereas the final values became 1, 0.902, and 0.906 for 1-, 2-, and 3^+-term queries, respectively. This shows that although relevance feedback learning is exclusively applied for 1-term queries with $\tau_b = 1$ indicating a perfect match, the substantial increase in the final values of τ_b for the multiterm queries is indicative of the generalization capability of the system. This is significant especially for cases when some terms in the multiterm queries did not receive any relevance feedback during training. Also, note that the final τ_b values for the classification mode are higher as the documents are not ordered according to their scores.

Fig. 7(a) and (b) shows the plots of recall measure for various choices of top documents after the relevance feedback, using the regression and classification learning modes, respectively. These plots are generated for the most commonly used 1-, 2-, and 3^+-term queries in the log file by taking the average of the recall values. As mentioned before, the system was initially trained using the indexer results. As can be seen from Figs. 4 and 7, the maximum achieved recall increased considerably after the relevance feedback learning process was applied. In the case of regression-based relevance feedback learning, the maximum achieved recall increased from 0.673, 0.720, and 0.657 [see Fig. 4(a)] to 0.961, 0.980, and 0.869 [see Fig. 7(a)] for the 1-, 2-, and 3^+-term queries, respectively. Similarly, for the classification mode, the maximum recall achieved increased from 0.551, 0.453, and 0.385 [see Fig. 4(b)] to 0.808, 0.797, and 0.698 [see Fig. 7(b)] for the 1-, 2-, and 3^+-term queries, respectively. These results indicate the effectiveness of the proposed relevance feedback learning in both regression and classification modes.

2) Training Based Upon Multiterm Queries: In this paper, the multiterm query learning method developed in Appendix A is tested and analyzed. Two different experiments were conducted. In the first experiment, we split the queries in the log file into a training set consisting of 2386 1-term queries, 1664 2-term queries, and 1661 3^+-term queries randomly drawn from the set of 1846 3^+-term queries, and a testing set consisting of the remaining 185 3^+-term queries. Note that the purpose of this experiment is to evaluate the system performance when new 3^+-term queries are encountered. In the second experiment, we split the queries in the log file into a training set consisting of 2386 1-term queries and 1664 2-term queries, and a testing set consisting of 3^+-term queries that have common terms with the 1- or 2-term queries or both. That is, only those 3^+-term queries that contained terms that participated in the training are used for testing and performance evaluation.

Fig. 8(a) and (b) shows the plots of Kendall's τ_b versus the number of epochs for the training (solid line) and testing sets

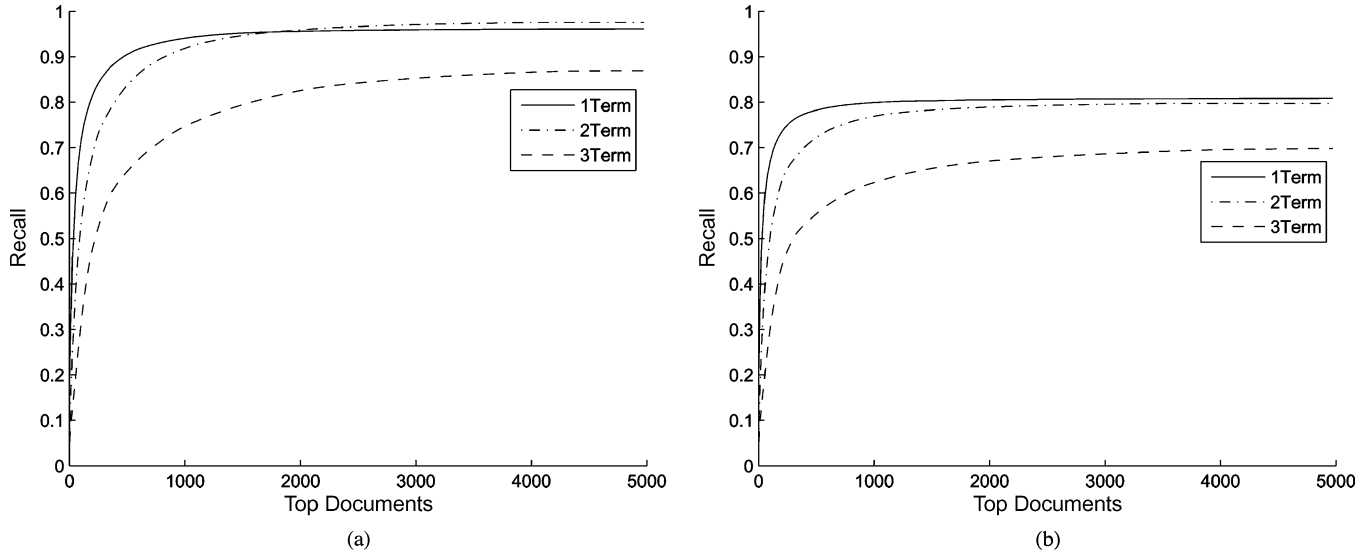


Fig. 7. Recall plot after relevance feedback learning in (a) regression mode and (b) classification mode.

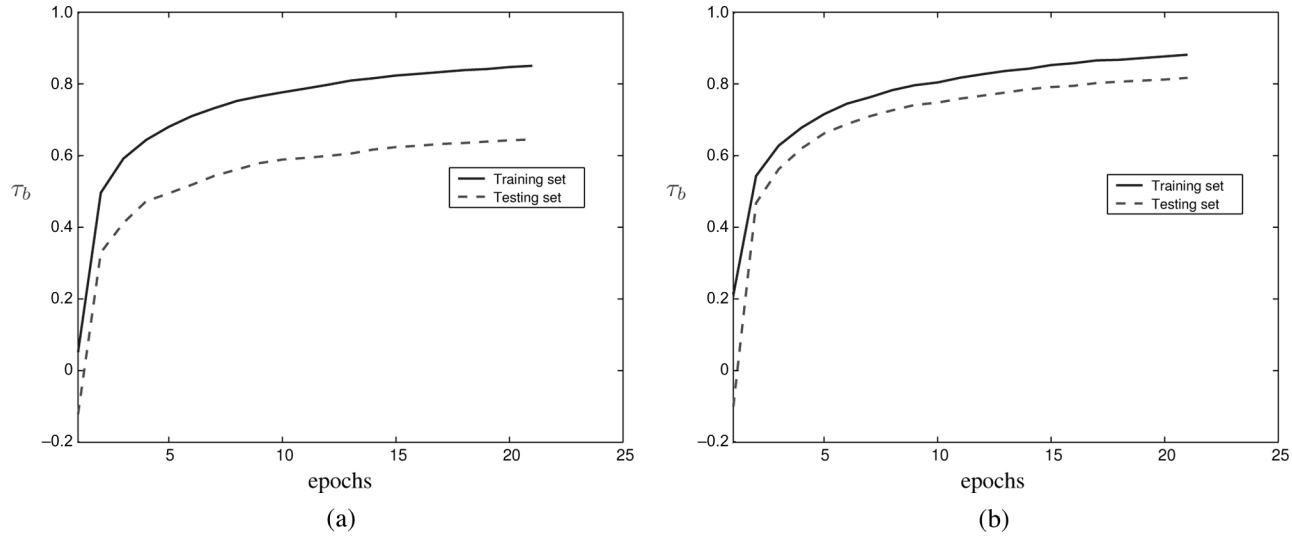


Fig. 8. Kendall's τ_b for two experiments. (a) Kendall's τ_b versus epochs (first experiment). (b) Kendall's τ_b versus epochs (second experiment).

(dashed line) for the first and second experiments, respectively. For any submitted query, only the top 20 documents where considered to generate Kendall's τ_b measure. The training process consisted of a series of epochs in which all the samples in the training set were submitted to the system and necessary adjustments to H^{-1} and $\widehat{W}$ using (A.5) and (A.6) were made. Furthermore, at each epoch, the Kendall's τ_b performance measures for all the samples in the training and testing sets were generated and plotted. As can be seen, in both figures, τ_b starts from very low values, indicating that the document lists from the initially trained system using the projection matrix P_D and those of the benchmark are not highly correlated. However, as relevance feedback training progresses, the τ_b values for the training set ends up at high values of approximately 0.85 and 0.90 for the first and second experiment, respectively. This indicates that the retrieval system closely captures the information content in the benchmark. Moreover, comparing to the results of the first experiment, in the second experiment τ_b values for the testing set follow more closely those of the training set. This behavior is expected since the testing queries in the second experiment have common terms with the training queries. The performance of the system on the testing set, as shown in Fig. 8(b), for the second experiment illustrates the generalization ability of the learning algorithm on multiterm queries. It is important to point out that Kendall's τ_b plots resemble most learning curves in the sense that the learning rate is high during the early stages of learning and decreases gradually as learning progresses.

D. Query Association and Clustering

The great difficulty in querying is that the user has to specify the right query in order to retrieve the desired results. Hence, it is prudent for the system to suggest relevant terms from which the user can pick one or more terms to augment and fine-tune the original query. In this section, we show how the weights learned by the query mapping mechanism may be used for document term associations to provide such query refinement suggestions.

TABLE II
TERM ASSOCIATION FOR THE QUERY "PCLXL ERROR"

Associated relevant queries	Matching value
PCLXL ERROR	1
ERROR	0.8325
79 ERROR	0.6520
ERROR CODE	0.6288
NUMER ERROR	0.5949
SEVER ERROR	0.4833
PCL XL ERROR	0.4444
SPOOL 32 ERROR	0.4331

TABLE III
TERM ASSOCIATION FOR THE QUERY "USB CHIPSET"

Associated relevant queries	Matching value
USB CHIPSET	1
USB CHIPSET ISSU	0.9165
LJ1000 USB CHIPSET ISSU	0.9043
LJ1200 USB CHIPSET ISSU	0.9041
USB	0.7965
CHIPSET	0.6242
USB TROUBLESHOOT	0.4331
IPD	0.4124

To test the document term association, we have used a subset of the most frequently submitted queries taken from the log file. When a new query is submitted, the query terms are evaluated against these most frequently submitted queries and those queries that are more related (in concept) to the submitted query are retrieved and suggested to the user for query refinement. The selection of the related terms is based upon determining the amount of match between the weight vectors of the query terms, i.e., $\underline{w}_i$s, of the first layer of the network, and those of the queries in the log file. If the match is above a prespecified threshold (0.4), then they are selected and suggested to the user for query refinement. The weight vectors captured by the network for single-term queries have to be mean corrected and normalized prior to the matching process. Consequently, term-matching via dot product operation corresponds to finding the cosine of the angle between the two weight vectors.

The suggested query terms are then evaluated by the expert users for their relevance to the submitted query terms. The results reveal that on average 84.5% of the suggested terms are indeed relevant to the query term. This shows the usefulness of first layer mapping weights for term association based upon their captured concept. It should be noted that this approach of query refinement using term association, which is one of by-products of our system, is very fast and amenable for real-time operation. The results of query refinement are shown for two user submitted queries in Tables II and III. Columns 1 and 2 in these tables show the suggested terms and their corresponding match index, respectively. The terms on the first row of these tables, which have a match value of 1, are the actual user-submitted query terms while the rest are the suggested ones. Clearly, one can see the similarity in their concept and relevance to the original submitted queries. These results point to this interesting observation that the weight vectors for single-term queries that are captured in the first layer of our network indeed contain useful information for query association and clustering applications.

VII. CONCLUSION AND DISCUSSION

A new adaptive MRTRS is proposed in this paper. The learning can be implemented in three phases using a three-layer connectionist network structure. Initial model-reference learning captures the behavior of a reference model or the documents' content information. Model-reference following is needed in dynamic environments where documents are to be added, deleted, or updated. This feature makes the network suitable for adaptive and dynamic document retrieval applications. To capture users' expertise and knowledge, a relevance feedback learning process using either score-based or click-through selection is proposed. The learning can be implemented in regression or classification modes for single- and multiterm queries. The user feedback is employed to administer the expert user voting based upon frequency of votes, users' expertise, or any other externally imposed business rules and heuristic criteria. The second and third layers perform document-to-term mapping and search/retrieval tasks, respectively. The effectiveness of the proposed algorithms is demonstrated on a large domain-specific text database containing various HP products. A benchmarking with BM25 scoring algorithm is also provided indicating much better performance than BM25.

The proposed MRTRS provides a flexible text search and retrieval engine that possesses the following several desirable key benefits: 1) ability to continuously learn by modifying the internal parameters during the interaction with multiple expert users, 2) preserving stability of the stored information while offering flexibility to incorporate new information from the users via relevance feedback, 3) ability to update the knowledge-base in dynamic environments, and 4) simplicity needed for real-life implementation. Clearly, the main feature of our approach is the ability of the users to contribute to the decision-making capability of the system and to enhance the performance of the system by modifying the knowledge content retrieval directly in the context of their specific needs.

The proposed system is applicable to other similar domain-specific applications, e.g., homeland security, as in [25], where document collections are limited to their specific organizational content and needs. In this case, for example, an adaptable TRS can be designed for specific document collection, queries, and expertise of the users at different emergency management agencies. The user expertise and information is captured through an analysis of the context information via relevance feedback learning using either score-based (log file) or click-through selection. Similar to the *vista* model in [25], our system can refine the user

queries and autonomously identify the relevant key terminologies or the "query anchors" using the query clustering mechanism in Section VI-D. The most relevant terms are provided to the user as "suggestions" to modify or enhance the submitted query. This feature is very similar to the "traction" capability of *vista*. Finally, "concept switching" [25] in our system can be incorporated using a hierarchical search and retrieval structure employing various local adaptable TRSs by allowing the traction concepts to be shared among various organizations. This system has been implemented by Hewlett Packard Corporation for all their product collections where an adaptable TRS is used for every set of document collections depending on their categories.

APPENDIX A
MULTITERM QUERY LEARNING

The developed algorithms for phases 1 and 3 can be extended to account for multiterm as well as single-term query learning. To see this, let us divide the problem of finding mapping matrix $Z = DW$ (see Fig. 1) given an ensemble of training samples into the problem of finding certain columns of mapping matrix $Z = [\underline{z}_1\ \underline{z}_2 \dots \underline{z}_M]$ by using only specific samples that convey the necessary information. For instance, queries that contain one or more terms t_{j_1}, t_{j_2}, and t_{j_3}, $j_1, j_2, j_3 \in [1, M]$, can be used to find columns $\underline{z}_{j_1}$, $\underline{z}_{j_2}$, and $\underline{z}_{j_3}$, respectively.

Now, the problem of finding T columns $\underline{z}_{j_1}, \underline{z}_{j_2}, \dots \underline{z}_{j_T}$ given a set of T-term queries $\underline{q}_i = \sum_{l=1}^{T} q_{ij_l} \underline{e}_{j_l}$, $i \in [1, K]$ with $K \geq T$, with their respective outputs $\underline{r}_i$s, where $\underline{e}_{j_l}$ is the vector containing 1 at the j_l^{th} position and zero elsewhere, can be cast in an optimization framework. The goal here is to find $\underline{z}_{j_1}, \dots, \underline{z}_{j_T}$ and $\underline{w}_i$s that minimize the Lagrangian function

$$J(\underline{z}_{j_1}, \dots, \underline{z}_{j_T}, \underline{w}_1, \dots, \underline{w}_K) = \frac{1}{2} \sum_{i=1}^{K} \hat{\underline{q}}_i^T \hat{\underline{q}}_i + \sum_{i=1}^{K} \underline{w}_i^T (\hat{\underline{r}}_i - D^T \hat{\underline{q}}_i). \tag{A.1}$$

Since we require linear optimal mapping of the form $\hat{\underline{q}}_i = Z\underline{q}_i = \sum_{l=1}^{T} q_{ij_l} \underline{z}_{j_l}$, $i \in [1, K]$, we set the derivative of J w.r.t. $\underline{z}_{j_l}$ to $\underline{0}$. This yields

$$\sum_{i=1}^{K} \hat{\underline{q}}_i q_{ij_l} - \sum_{i=1}^{K} q_{ij_l} D \underline{w}_i = \underline{0}, \qquad l \in [1, T]. \tag{A.2}$$

To transfer the information of the ensemble of optimal queries $\hat{\underline{q}}_i$s into the mapping matrix, let us plug the ith optimal query $\hat{\underline{q}}_i = \sum_{l=1}^{T} q_{ij_l} \underline{z}_{j_l}$ into (A.2). This gives

$$\sum_{p=1}^{T} \left(\sum_{i=1}^{K} q_{ij_p} q_{ij_l} \right) \underline{z}_{j_p} = D \sum_{i=1}^{K} q_{ij_l} \underline{w}_i = D \hat{\underline{w}}(l), \qquad l \in [1, T] \tag{A.3}$$

which must hold for the query terms $t_{j_1}, \dots, t_{j_T}$. Here, $\hat{\underline{w}}(l) = \sum_{i=1}^{K} q_{ij_l} \underline{w}_i$, $l \in [1, T]$, and the term in the bracket in (A.3) represents a correlation measure between the terms in the set of queries. To find a solution for $\underline{z}_{j_l}$s, let us define the matrix $H = [h_{pl}]$, $p, l = 1, 2, \dots, T$ with elements $h_{pl} = \sum_{i=1}^{K} q_{ij_p} q_{ij_l}$ and matrix $\widehat{W} = [\hat{\underline{w}}(1)\, \hat{\underline{w}}(2) \dots \hat{\underline{w}}(T)]$. Then, (A.3) can be rewritten in matrix form $\widehat{Z}H = D\widehat{W}$, where matrix $\widehat{Z} = [\underline{z}_{j_1} \underline{z}_{j_2} \dots \underline{z}_{j_T}]$ contains columns $j_1, j_2, \dots, j_T$ of Z. If we solve for $\widehat{Z}$, we get

$$\widehat{Z} = D\widehat{W}H^{-1}. \tag{A.4}$$

As can be seen, to solve for $\widehat{Z}$, we need to compute $\widehat{W}$ and H^{-1}, where H^{-1} (inverse of a $T \times T$ correlation matrix) exists and is easy to compute as T is usually small ($T < K$) and queries with few terms are abundant.

A recursive equation for $\widehat{Z}$ can be found using the corresponding recursive equations for $\widehat{W}$ and H^{-1}. Since $H(k)$, the correlation matrix for query "k," can be written as a function of the new query sample $\underline{q}_k$ and $H(k-1)$ as $H(k) = H(k-1) + \underline{q}_k \underline{q}_k^T$, $k \in [1, K]$, its inverse can easily be computed using the matrix inversion lemma [31]

$$H^{-1}(k) = H^{-1}(k-1) - \frac{H^{-1}(k-1)\underline{q}_k \underline{q}_k^T H^{-1}(k-1)}{1 + \underline{q}_k^T H^{-1}(k-1) \underline{q}_k}. \tag{A.5}$$

Now, since the constrains in (A.1) are given by $\hat{\underline{r}}_i = D^T \hat{\underline{q}}_i$, $i = 1.2 \dots K$, this implies that (A.2) can be rewritten as $\sum_{i=1}^{K} (\hat{\underline{r}}_i - D^T D \underline{w}_i) q_{ij_l} = \underline{0}$, $j_l \in [1, T]$. From here, the Lagrange multipliers $\underline{w}_i = (D^T D)^{-1} \hat{\underline{r}}_i$, $i \in [1, K]$ are found and then used to form the columns of $\widehat{W}(k)$ to obtain $\widehat{W}(k) = (D^T D)^{-1} \sum_{i=1}^{k} \hat{\underline{r}}_i \underline{q}_i^T$. Consequently, the recursive equation for $\widehat{W}(k)$ is

$$\widehat{W}(k) = \widehat{W}(k-1) + (D^T D)^{-1} \hat{\underline{r}}_k \underline{q}_k^T. \tag{A.6}$$

Having found $\widehat{W}(k)$ and $H^{-1}(k)$, $\widehat{Z}(k)$ can be computed using $\widehat{Z}(k) = D\widehat{W}(k) H^{-1}(k)$.

Since the retrieval system initially starts from the projection matrix $P_D = D(D^T D)^{-1} D^T$, the corresponding initial values for $\widehat{W}$ and H^{-1} are

$$\widehat{W}(0) = (D^T D)^{-1} D^T \quad \text{and} \quad H^{-1}(0) = I \tag{A.7}$$

ACKNOWLEDGMENT

The authors would like to thank the TRS worldwide development teams at Hewlett Packard, Roseville, CA, for providing the data, review, extended verifications, and technical support.

REFERENCES

[1] D. Harman, *Relevance Feedback and Other Query Modification Techniques*. Englewood Cliffs, NJ: Prentice-Hall, 1992, pp. 241–263.
[2] J. J. Rocchio, "Relevance feedback in information retrieval," in *The Smart Retrieval System: Experiments in Automatic Document Processing*, G. Salton, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1971.
[3] E. Ide, "New experiments in relevance feedback," in *The Smart Retrieval System: Experiments in Automatic Document Processing*, G. Salton, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1971.
[4] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. New York: Addison-Wesley, 1999.
[5] C. Carpineto and G. Romano, "Order-theoretical ranking," *J. Amer. Soc. Inf. Sci.*, vol. 51, no. 7, pp. 587–601, 2000.
[6] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *J. Mach. Learn. Res.*, vol. 2, pp. 45–66, 2002.
[7] C. Cortes and V. Vapnik, "Support vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[8] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.
[9] G. D. Guo, A. K. Jain, W. Y. Ma, and H. J. Zhang, "Learning similarity measure for natural image retrieval with relevance feedback," *IEEE Trans. Neural Netw.*, vol. 13, no. 4, pp. 811–820, Jul. 2002.
[10] S. Wong and Y. Yao, "Query formulation in linear retrieval models," *J. Amer. Soc. Inf. Sci.*, vol. 41, pp. 334–341, July 1990.

[11] Y. Ouyang and W. Jermann, "Neural network based retrieval issue on prototype database systems," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 1991, vol. 3, pp. 1493–1497.
[12] K. L. Kwok, "A network approach to probabilistic information retrieval," *ACM Trans. Inf. Retrieval*, vol. 13, pp. 324–353, 1995.
[13] F. Crestani, "Learning strategies for an adaptive information retrieval system using neural networks," in *Proc. IEEE Int. Conf. Neural Netw.*, 1993, vol. 1, pp. 244–249.
[14] R. C. Muniyandi, "Neural network: An exploration in document retrieval system," in *Proc. TENCON*, Sep. 2000, vol. 1, pp. 156–160.
[15] M. Boughamen, A. Caron, and R. Layaida, "A neural network model for documentary self-organizing and querying," in *Proc. 5th Int. Conf. Comput. Inf. (ICCI)*, May 1993, pp. 512–518.
[16] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2 ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.
[17] A. Bouchachia and R. Mittermeir, "A neural cascade architecture for document retrieval," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2003, vol. 3, pp. 1915–1920.
[18] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," *Comput. Learn. Theory*, pp. 287–294, 1992.
[19] T. Joachims, "Optimizing search engines using clickthrough data," in *Proc. ACM Conf. Knowl. Discovery Data Mining (KDD)*, 2002.
[20] N. Fuhr, "Optimum polynomial retrieval functions based on the probability ranking principle," *ACM Trans. Inf. Syst.*, vol. 7, no. 3, pp. 183–204, 1989.
[21] M. G. Kendall, *Rank Correlation Methods*. New York: Hafner, 1962.
[22] Y.-C. Chang and S.-M. Chen, "A new query reweighting method for document retrieval based on genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 617–622, Oct. 2006.
[23] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma, "Query expansion by mining user logs," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 4, pp. 829–839, Jul./Aug. 2003.
[24] S. Feinglos, *MEDLINE: A Basic Guide to Searching*. Chicago, IL: Medical Library Assoc., 1985.
[25] T. Goan and I. Mayk, "Improving information exchange and coordination amongst homeland security organization," in *Proc. Int. Command Control Tech. Symp. (ICCRTS): Homeland Security*, Washington, DC, Jun. 13–16, 2005.
[26] S. E. Robertson, S. Walker, M. Hancock-Beaulieu, M. Gatford, and A. Payne, "Okapi at TREC-4," in *Proc. 4th Text Retrieval Conf. (TREC-4)*, 1995, pp. 73–86.
[27] R. Manmatha, T. Rath, and F. Feng, "Modeling score distributions for combining the outputs of search engines," in *Proc. 24th Ann. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2001, pp. 267–275.
[28] L. L. Scharf, *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*. New York: Addison-Wesley, 1991.
[29] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD: The John Hopkins Univ. Press.
[30] S. Srinivasan and M. R. Azimi-Sadjadi, "An iterative relevance feedback learning algorithm for image retrieval system," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Aug. 2005, vol. 1, pp. 604–609.
[31] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2 ed. New York: Wiley, 2001.

M. R. Azimi-Sadjadi (SM'89) received the M.S. and Ph.D. degrees from the Imperial College of Science and Technology, University of London, London, U.K., in 1978 and 1982, respectively, both in electrical engineering with specialization in digital signal/image processing.

Currently, he is a Full Professor at the Electrical and Computer Engineering Department, Colorado State University (CSU), Fort Collins. He is also the Director of the Digital Signal/Image Laboratory at CSU. His main areas of interest include digital signal and image processing, target detection, classification and tracking using broadband sonar, radar and IR systems, adaptive filtering and system identification, and NNs. His research efforts in these areas resulted in over 170 journal and refereed conference publications. He is the coauthor of the book *Digital Filtering in One and Two Dimensions* (New York: Plenum, 1989).

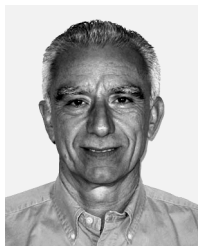
Dr. Azimi-Sadjadi is the recipient of the 1999 the ABELL Teaching Award, 1993 ASEE-Navy Senior Faculty Fellowship Award, 1991 CSU Dean's Council Award, and 1984 DOW chemical Outstanding Young Faculty Award. He served as an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING and the IEEE TRANSACTIONS ON NEURAL NETWORKS.

J. Salazar received the B.S. degree in physics and the M.S. degree in computer science from the Instituto Teconológico y de Estudios Superiores de Monterrey (ITESM), Mexico, in 1981 and 1991, respectively, and the Ph.D. degree in electrical and computer engineering from Colorado State University, Fort Collins, in 2006.

Currently, he is an Associate Professor at the Computer Science Department, ITESM, Toluca Campus, Mexico. His research areas of interest include: pattern recognition, neural networks, text and image retrieval systems, and image processing

S. Srinivasan received the B.E. degree in electronics and computer engineering from Coimbatore Institute of Technology, Coimbatore, India, in 2000 and the M.S. degree in electrical engineering from Colorado State University, Fort Collins, in 2004.

From 2000 to 2002, he was a Software Engineer at Larsen and Toubro Infotech. Since 2005, he has been with Information System Technologies Inc., Fort Collins, CO, where he is a member of the Technical Research Staff involved in several distributed acoustic sensing projects. His research interests include digital signal processing, target localization and classification, and NNs.



S. Sheedvash received the M.S. degree from University of Arkansas, Fayetteville, in 1975 and the Ph.D. degree from Colorado State University, Fort Collins, in 1990, both in electrical and computer engineering.

He joined Hewlett Packard, San Diego, CA, in 1999, where currently, he is conducting various R & D programs. His main research interests are in intelligent learning systems, NNs, pattern recognition/understanding, and data mining with applications to search, knowledge management, and multimedia retrieval systems.