PERGAMON

Information Processing and Management 36 (2000) 415–444

INFORMATION PROCESSING & MANAGEMENT

www.elsevier.com/locate/infoproman

Automatic classification using supervised learning in a medical document filtering application

J. Mostafa [a,*], W. Lam [b]

[a] *School of Library and Information Science, Indiana University, Bloomington, IN 47405-1801, USA*
[b] *Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong, Shatin, New Territories, Hong Kong, People's Republic of China*

Abstract

Document classifiers can play an intermediate role in multilevel filtering systems. The effectiveness of a classifier that uses supervised learning was analyzed in terms of its accuracy and ultimately its influence on filtering. The analysis was conducted in two phases. In the first phase, a multilayer feed-forward neural network was trained to classify medical documents in the area of cell biology. The accuracy of the supervised classifier was established by comparing its performance with a baseline system that uses human classification information. A relatively high degree of accuracy was achieved by the supervised method, however, classification accuracy varied across classes. In the second phase, to clarify the impact of this performance on filtering, different types of user profiles were created by grouping subsets of classes based on their individual classification accuracy rates. Then, a filtering system with the neural network integrated into it was used to filter the medical documents and this performance was compared with the filtering results achieved using the baseline system. The performance of the system using the neural network classifier was generally satisfactory and, as expected, the filtering performance varied with regard to the accuracy rates of classes.

Keywords: Supervised learning; Neural networks; Document classification; Information filtering

* Corresponding author. Tel.: +1-812-856-4182; fax: +1-812-855-6166.
E-mail address: jm@indiana.edu (J. Mostafa).

0306-4573/00/$ - see front matter
PII: S0306-4573(99)00033-3

1. Introduction

Improving the dissemination of biomedical information has been a prominent driver behind the two recent National Science Foundation initiatives (NSF) — Knowledge and Distributed Intelligence (KDI) and Digital Libraries (DL) — indicating that this particular area is of critical scientific interest and requires closer attention from researchers (NSF, 1998a, 1998b). The World Wide Web offers an extremely efficient means for achieving wide-scale information dissemination. The rapid growth rate of the web, however, has made it difficult for web search-engines to cover comprehensively any particular domain, let alone the universe of web documents. For example, Lawrence and Giles (1998) found that the coverage of the most comprehensive search-engine (HotBot) was about 34%. An important associated problem, which unfortunately receives less emphasis, is the effort required on the part of users to keep up with information updates. The web makes it possible to easily add and remove documents, modify published documents, or change the location of documents. These conveniences and the resulting frequency of updates make it difficult to keep up with changes. The expectation that users on a regular basis would take the time to select reliable search-engines, formulate the correct search, execute and refine the search and then download the appropriate documents is somewhat unrealistic.

As a way to deal with some of the problems mentioned above, information filtering (IF) systems have recently attracted the attention of researchers and developers. IF systems rely on many established techniques applied in information retrieval (IR), for example, indexing, matching, feedback, etc. However, IF systems generally are constructed to deliver 'personalized' information from sources that are dynamic or change frequently. Once created, an IF system can be used in various ways. For example, an IF system can complement a search-engine supporting customized web-pages (e.g. Yahoo's www.my.yahoo.com site), it can be part of a more conventional selective dissemination of information service (e.g. Uncover Reveal's uncweb.carl.org site), or it can support personalized access to a single site containing many documents (e.g. Publications agent site at www.ics.uci.edu/~pazzani/Agents.html). The popularity and proliferation of IF services demonstrate that they can aid in reducing the manual effort necessary to keep up with information updates (Maes, 1995; Robertson, 1997).

The objective of this paper is to present and analyze a new model of the filtering process called the multilevel model. The model permits incorporation of various procedures and schemes to conduct document classification. A particular classification approach, based on a supervised learning algorithm, is evaluated in this paper. The accuracy of the algorithm is measured by comparing it with a baseline system that directly uses human classification information. Then a system using the algorithm is compared with a baseline system to determine the overall impact of the supervised classification approach on filtering effectiveness.

We begin the paper by describing the operational definitions of key concepts and presenting the multilevel filtering model in Section 2. In Section 3, we review related literature and elaborate on the problem addressed in this research. In Section 4, the supervised learning process is explained along with a description of the filtering system used in this research. The research methodology, dependent and independent variables and the experimental process are

presented in Section 5. This is followed by the data analysis in Section 6. The paper concludes in Section 7 with a review of the major findings and recommendations for future research.

2. Operational definitions of key concepts

The overarching objective of this research is to analyze information filtering performance. The goal is to evaluate a multilevel model of filtering that subdivides its principal operations into distinct modules. A parallel focus is to clarify the role of a particular module in the multilevel model that is responsible for document classification. Hence, filtering and classification are both core concepts in this research. A literature search of cognate areas revealed numerous, often conflicting, definitions of these two concepts, making the need for tightly-bounded and concrete definitions especially desirable. Below we present definitions for the core concepts, as operationalized in this research.

Filtering has the exclusive objective of identifying the relevance of documents according to interest profiles. An interest profile is a data structure, usually covering a set of topics, representing the long-term interest of the user (Belkin & Croft, 1992). We assume the input to the filtering system is a batch of newly arrived documents and the output is a ranked list of the documents created according to distribution of interest values in the profile. We envision a scenario where a networked user regularly receives new documents in a pre-designated document 'bin' (via e-mail or a web-robot). The function of the filter is to establish the relevance value of each document according to interest values in the profile and then present the documents to the user. Presentation can involve grouping, sorting, or even pruning certain documents based on the relevance values. In this research, the documents are rankordered and all new documents are presented to the user.

Lewis (1992) observed that the term 'classification' is ambiguously used in information retrieval, applied statistics, psychology and other fields. However, according to Lewis, the term almost always refers to the task of aggregation of like-entities. In this research, classification refers to the particular task of grouping documents according to classes. We define classes as high-level concepts, each representing a sub-area or subdivision in a particular domain. The classes are best understood as top level subject headings or descriptors as found in conventional classification schemes. There are many classification schemes covering almost all the established disciplines. In this research, focusing on biomedical information, we used the Medical Subject Headings List[1]. Any established domain can be divided into significant subject subdivisions; a class, usually represented as a concise phrase, is the central or representative subject in each of the subdivisions. For example, in Fig. 1, we show that the domain of cell physiology contains 15 subject subdivisions. We treated each of these subdivisions as a class.

An important characteristic of classes (in the way we define them) is that they are sufficiently broad to *subsume* lower level semantic units. Any significant topic that falls under a class and

[1] The Medical Subject Headings List (MeSH) is created and maintained by the National Library of Medicine in the USA. In our research, we selected cell physiology as the domain. In MeSH, there are numerous lower level headings and sub-headings under the major heading cell physiology. We chose 15 headings immediately under cell physiology as our class set.

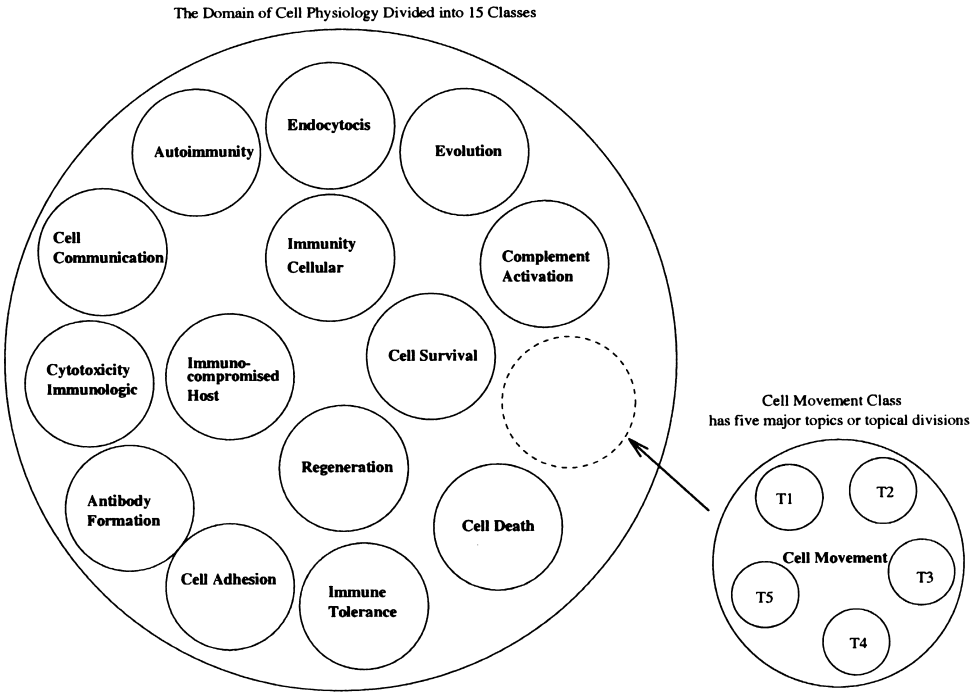

Fig. 1. The domain of cell physiology and its 15 classes selected for this research.

is directly related to a class can qualify as a unit. A secondary but critical corollary of this definition is that there is usually an n:1 relationship between topics that are significant in each class and the class itself. That is, the class can be treated as a *semantic distillation* of n terms into a single term, where n almost always is significantly higher than 1. An example of semantic distillation is shown in Fig. 2 for the class 'Cell Movement'.

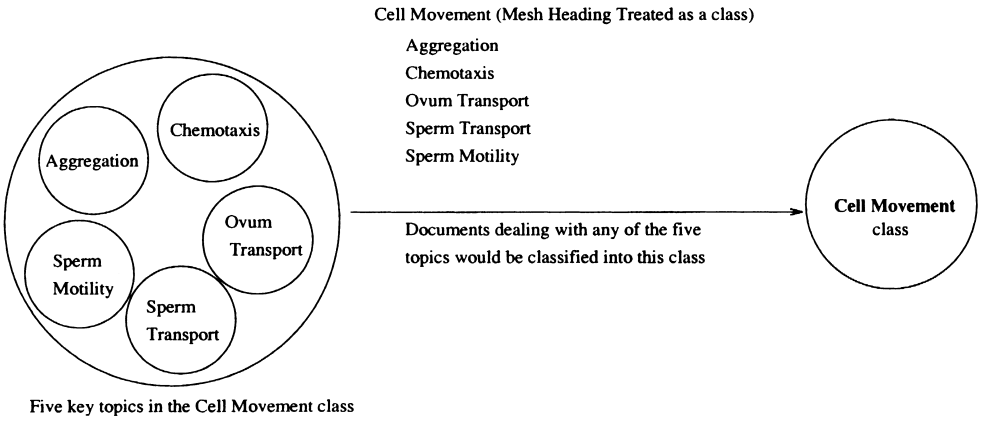

Fig. 2. Relationship between lower level terms and a class; the semantic distillation can be said to be 5:1 between the two levels.

2.1. How is classification related to filtering?

In our multilevel model, the content of each document is ultimately reduced to a class and filtering is conducted based on the document class (Fig. 3). Classification is conducted over document representations that contain topical information. When the system is invoked, the new documents are converted to numerical representations containing a fixed set of features. Each feature indicates the 'weight' of a particular topic in the document (the calculation of the weight will be described later). Then, the document representations are classified into classes. The classification module in the system places incoming document representations into one and only one class. The output of the module consists of a single class label for each document. This classification step is conducted prior to and independently of the procedure for identifying document relevance.

The profile data structure is an array with one element for each class and each element contains a numerical value representing the degree of user's interest in the corresponding class. Thus, the structure of the profile is directly determined by the number of classes the system maintains. After a class label for a document is established, the system can establish the relevance value of the document by looking up the interest value for that class in the profile (e.g. user has an interest of 0.7 in documents belonging to class 1). Subsequently, the documents are sorted according to their relevance values and presented to the user in that order. This is the basic process of filtering we apply in this research.

2.2. Advantages of a classification layer in filtering

In our approach to filtering, the profile structure is directly determined by classes. The intermediate classification layer and the application of classes provide the following primary advantages.

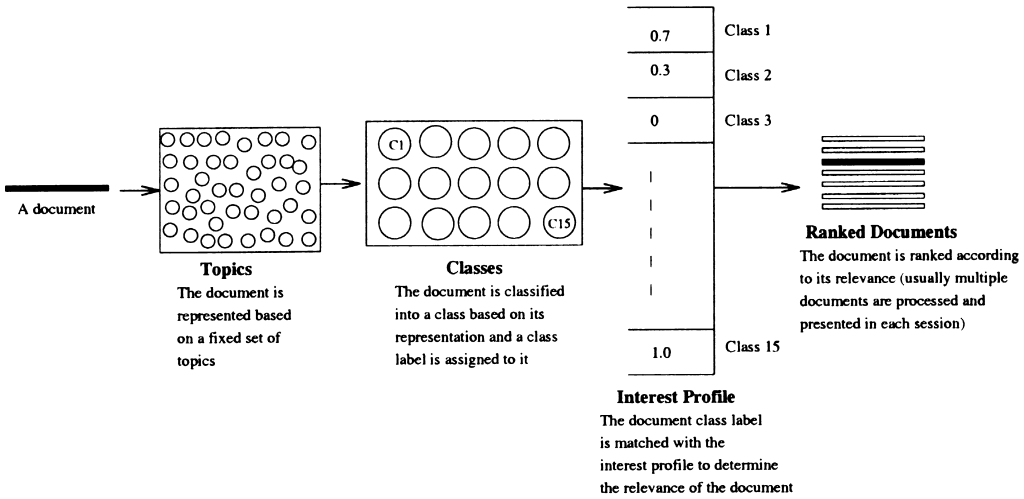

Fig. 3. Relationship between document classification and filtering.

- Classes allow us to use a relatively concise and simple representation for capturing user's interest.
- As classes (in the way we defined them) are high level and authoritative semantic units, they make the overall filtering operation more predictable and manageable.

A large amount of research evidence accumulated in the area of information retrieval demonstrates that critical operations involving topical vocabularies, for example, assignment of appropriate index terms and query formulation, can be extremely complex (Bates, 1998). Instead of using vocabularies extracted from documents to build the profile, the classification layer as implemented in this research introduces a 'control vocabulary' level that makes the filtering process more stable and ultimately less complex.

We also gained a tertiary benefit of selecting a well-known and widely used classification scheme. MeSH is used in the world's largest medical database known as Medline (9 million records). In automating the classification component, we selected a supervised learning approach. For this type of learning algorithm, a large amount of training data is usually called for. Here, Medline provided an abundant source of preclassified and reliable training data. In other words, it was a knowledge source that we could directly and immediately leverage. This considerably reduced the computational demands (and associated manual effort) necessary in creating an accurate set of training records and ultimately producing a trained classifier.

In summary, we treat filtering as a process of presenting documents to the user that are sorted according to an interest profile. Building and maintaining the profile is a major operation in the filtering system. We employ an intermediate step of grouping documents into a relatively small set of fixed classes. The explicit grouping process is a feature of our filtering model that we believe makes the overall filtering process less complex and more manageable. We use the term 'complexity' not in the strict computer science sense of time–space complexity. Rather, we use the term broadly to cover the implementation complexity associated with establishing a profile structure, acquiring interest information based on the profile, determining topics to include in the profile and also generating the classifier. Use of controlled vocabulary or classification labels in conventional online databases is often justified on the grounds of attaining more accurate and predictable search results. The classification layer in our model and the use of an established classification scheme are also motivated by similar concerns. We recognize that a controlled time–space complexity analysis of the classification layer would expand one's understanding of the efficiency gains achieved by the particular classification algorithm (as compared to another algorithm of the same type) and ultimately it would be a useful exercise. However, this research has a narrower scope: demonstrating and clarifying the link between a supervised classifier and filtering outcome exclusively in terms of effectiveness. Therefore, our measures for analyzing both classification and filtering concentrate on effectiveness instead of efficiency. We aim to analyze this link in terms of how classification quality may influence profile content and how that in turn may affect filtering service.

2.3. Multilevel filtering process

Formally, we express the filtering process as a mapping function $f:\mathscr{D}\rightarrow\mathbb{R}$, where $\mathscr{D}$ is the document set, $\mathbb{R}$ represents relevance assessment and $f(d)$ corresponds to the relevance of a

document d. Given that such a map is known, any document in $\mathscr{D}$ can be pruned, sorted, or grouped using the relevance assessment. To map documents directly to profiles requires extracting the individual components (e.g. morphological units such as words) of documents and maintaining a correspondingly complex interest profile. To make the process computationally more manageable, f is decomposed into two functions such that the first function is a map $f_1:\mathscr{D}\rightarrow\{C_1,\ldots,C_m\}$ and the second function is a map: $f_2:\{C_1,\ldots,C_m\}\rightarrow\mathbb{R}$. The function f_1 produces mutually exclusive groups of documents. This should be contrasted with the process of categorization in which documents can belong to more than one group (Jacob, 1991) — a process that cannot be accommodated in the present model. The principle behind the document grouping process is that the number of groups, as determined by the number of classes, should be relatively small and remain fixed during the online filtering mode.

In our work on IF, we developed the multilevel document filtering model to incorporate the document representation, classification and interest profile-management in a single architecture (Fig. 4). A particularly useful feature of the architecture is that it is highly modular and, therefore, virtually any technique applicable for representation, classification or profile-management can be integrated into a single system. In our earlier work, based on an implemented filtering system, we examined the utility of our IF model in a 'holistic' manner — concentrating on the overall filtering performance. However, in the course of conducting our work we found that automated document classification can have significant influence on the filtering outcome. This motivated a thread of recent research on machine learning techniques applied on document classification and their impact on filtering (Jacob, Mostafa & Quiroga, 1997; Mostafa, Quiroga & Palakal, 1998). In this article, we extend that work to an examination of the supervised learning approach based on neural networks.

3. Related research and problem definition

Below, we review literature from related areas of information filtering. The scope of the review, by necessity, is somewhat broad. The survey covers IR, IF, classification, user modeling and machine learning literature. An unfortunate consequence of the breadth of the coverage is that certain terms lose their precision, because their usage and meanings vary depending on the context of the research area. A particularly difficult term is 'classification'. The term may mean

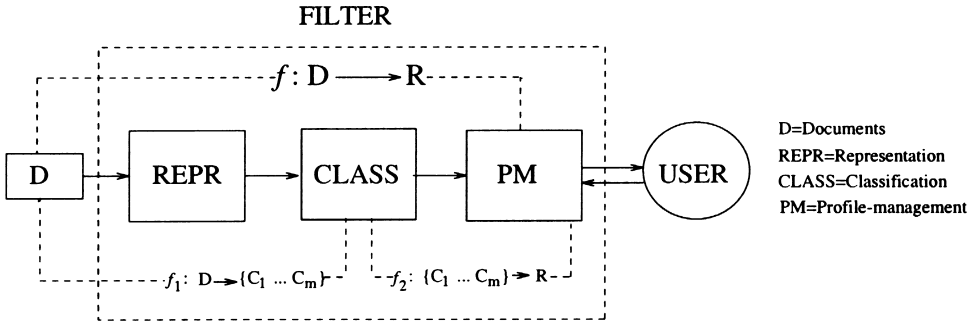

Fig. 4. Model of the filtering process.

a strictly topical grouping of documents (as we treat it), a grouping of documents into nontopical classes, or in a broader sense a grouping of like entities. Below, as part of the review, we will present relevant information in context to clarify the subtle differences in usage of this term. In the summary section, further clarification will be provided of the term 'classification'.

One way to differentiate among various streams of research in IF is to consider how interest profiles are captured, represented and updated by an IF system. One of the earliest web filtering system, SIFT, permitted profile construction using keywords specified by users (Yan & Garcia-Molina, 1995). These profiles were then matched against contents of USENET news articles to determine the relevant articles for individual users. SIFT permitted matching based on vector space similarity measurement and a more direct boolean comparison. The vector space comparison scheme also utilized relevance feedback to re-adjust term weights in the profiles for improving effectiveness. Foltz and Dumais (1992) compared two types of profile acquisition methods. In one approach, users were asked to directly contribute keywords for their profiles. In the second approach, a variant form of relevance feedback was applied indirectly to extract keywords from documents that the users rated as relevant. This research compared the effectiveness of the different profile representations in filtering technical documents generated at Bellcore over a six-month period. To match the profiles with documents, the researchers applied a straight vector space comparison and one based on a reduced set of Latent Semantic Indexing (LSI) dimensions. The LSI dimensions were produced as orthogonal factors by applying singular value decomposition on a large word-by-document matrix. Experimental data showed that profiles generated using the relevance feedback approach combined with LSI matching produced the best filtering results. Pazzani and Billsus (1997) described a system named Syskill and Webert that was capable of suggesting new web-sites of interest based on positive and negative feedback collected from users. A distinctive element of the Syskill and Webert system was that it treated identification of relevant documents as a classification problem involving establishing membership of documents into the two nontopical classes: hot (relevant) and cold (nonrelevant). In this research, part of profile acquisition process dealt with training a *classifier*. The researchers presented results based on several different classifiers including a Bayesian technique, two types of neural networks and decision trees. Overall results indicated that a linear classification approach implemented in the Bayesian classifier consistently produced superior filtering performance. For most experiments, a simple vector representation of documents was used whereby each element indicated the presence or absence of a significant term or a word. A more complex version of the system employed $tf \times idf$ (term frequency multiplied with inverse document frequency) weighting for each vector element. Experiments conducted using the more complex document representation found that the performance is as effective as those achieved using boolean features and the Bayesian classifier. Pazzani and Billsus (1997) also demonstrated a profile acquisition approach that combined directly solicited keywords (both hot and cold) from users with on-going profile revision based on user's feedback. Additional empirical evidence presented in their research showed that when the combined profile acquisition method was applied with the Bayesian classifier it produced the best overall filtering performance.

A distinct stream in filtering research concentrates on increasing the autonomy of the IF system to automate profile revisions (also called adaptation) and associated filtering operations.

Generally, researchers use the notion of 'agents', defined as intelligent software components (Jennings & Wooldridge, 1998), to refer to the implementation of autonomous operations in an IF system. One class of agents, developed by Maes and her group at MIT (Maes, 1994) could unobtrusively watch behaviors of e-mail system users and learn to associate various actions with the content of messages. These agents then automated tasks such as routing of messages to pre-designated folders or sorting of messages based on the sender. Maes (1995) stated that agents can also learn from other agents specialized in particular tasks and agents can be 'evolved' to improve their fitness in certain types of tasks using genetic algorithms. A system named Amalthaea (Moukas, 1997), also developed at MIT, supported filtering using two types of agents: IF and information discovery (ID). In Amalthaea, a core component of agents included weighted keyword vectors representing their specific content coverage. ID agents located documents by autonomously searching specific search-engines; they also monitored designated web sites for changes. IF agents acted as 'masks' that only allowed documents to pass through that were considered relevant. Both IF and ID agents were generated and evolved using a genetic algorithm (GA) that controlled an ecosystem of agents. The GA applied relevance feedback gathered from each cycle of use to adjust the number and type of agents in the ecosystem. Empirical data showed that in Amalthaea a user's interest was quickly learned with a fast rise in the fitness of appropriate agents. Additional evidence demonstrated that Amalthaea could also re-learn a user's interest when the interest changed or shifted. In the Browse system (Jennings & Higuchi, 1992), the agent built a user model based on a neural network. Here, the nodes of the neural network represented individual words and related words were connected with links. Weight values in each node captured the individual user's interest in particular concepts and values assigned to links stored the strength of word-associations. The authors argued that this nonlinear way of representing the user model allows for an in-depth and complex representation of personalized information need, as compared to what is possible using linear methods. The neural network user model was directly applied to establish relevance of USENET news feed for each day over a two-week period. Usability data showed that after a few days, with different user-adjustable relevance threshold settings, Browse successfully identified a significant proportion of relevant documents. However, a limitation of this system was the training time required, based on positive or negative relevance feedback, to achieve acceptable performance.

A different approach was taken by Payne, Edwards and Green (1997), in their agent-based filtering system. They created two types of agents: one used a rule induction algorithm called CN2 and another used a *K*-nearest neighbor algorithm called IBPL. These agents have been applied in two domains: mail messages and USENET news. In the e-mail system, agents were required to suggest a potential list of desired actions based on the content of messages and in the news system, agents were responsible for pruning uninteresting articles. Both types of agents represented content using the frequency of a select group of keywords extracted from documents. In the e-mail application, higher average accuracy was attained by CN2 (65%) than the IBPL (57%). In the news domain, it was found that with only two levels of filtering (interesting or dull), CN2 (59%) produced slightly higher accuracy rate than the IBPL (51%). However, when filtering levels were increased to six, performance of both CN2 (27%) and IBPL (25%) dropped drastically. Additional experimental data related to filtering conducted on USENET news showed that when content coverage of agents was further refined based on

relevant keywords directly acquired from users, it produced inconclusive results — for certain newsgroups accuracy rate went up, while for others it led to deteriorated performance. Certain researchers have observed that users may consider themselves lacking sufficient information about the domain of documents or simply may be unmotivated to provide explicit interest-oriented information. As an alternative means of collecting this information, researchers suggested gathering recommendations from other members of the user-community or a subset of the user population. This approach has led to the development of another stream of filtering research usually referred to as collaborative filtering. Examples of some collaborative filtering systems reported in the literature include the PHOAKS (People Helping One Another Know Stuff) system developed at AT&T (Terveen, Hill, Amento, McDonald & Creter, 1997), GroupLens (Konstan et al., 1997) and Siteseer (Rucker & Polanco, 1997). PHOAKS used recommendations posted in USENET news groups to suggest web sites, GroupLens collected ratings on news articles to rank them and Siteseer employed bookmarks to link users and suggest new sites. More information on collaborative filtering can be found in the special issue of the ACM Communications (Resnick & Varian, 1997).

So far in this review, we primarily concentrated on the interest-profile and various functions associated with IF. Two other significant strands of research are also relevant to IF. These are topical classification and routing.

According to Lewis (1995) one of the key classification processes is determining topical labels for individual documents. More broadly, this means establishing for each document a topical area or an identifier that represents the semantic content of the document. This sense of classification is more conventional, as typically understood in library or indexing practices. Pharos (Dolin, Agrawal, El Abbadi & Pearlman, 1998) is a prototypical example of topical classification system. The goal of this system, created by the Alexandria Digital Library team, was to allow users to identify diverse contents on the web based on keywords they enter into the system. In a demonstration system, users could enter keywords that were matched with the Library of Congress Classification Scheme (LCC). The user then could select a list of newsgroups that were linked to the online LCC version. In this system, LC classes were represented as reduced LSI dimensions and standard similarity measures were used to match these vector representations with queries and newsgroup articles. Similar document classification approaches have also been explored by Larson (1992), employing the LCC and the LC subject headings and by Cheng and Wu (1995), using the Dewey Decimal Classification scheme. The Construe system (Hayes, 1992) supplemented a newswire database environment by automatically determining topical labels for Reuters news stories. In experiments conducted with 674 labels and 723 stories it was found that Construe could accurately place a document in its category at least 94% of the time (described by the authors as *Recall*). Construe was a rule-based system and as such it was strongly tied to the domain of documents. One of the few recent examples of neural network application in topical classification of documents was presented by Lin (1997). In this research, a Kohonen feature map algorithm was used to partition document-collections into topical 'regions' for the purpose of visual display of the whole collection. The vector-space model proposed by Salton (1989) was used for document representation. A basic advantage of the Kohonen approach is that it requires little prior user intervention to successfully learn the distinct topical areas. Lin demonstrated the utility of this algorithm across different document collections.

Given a set of training documents, information need descriptions (topics) and relevance judgments, the goal in routing is to develop standing queries (similar to profiles) that can predict the relevance of new documents in the stream. Routing has been compared with filtering by Hull (1998) and in terms of techniques applied there are strong overlaps. A comprehensive comparison of routing systems has been annually conducted through TREC (Text REtrieval Conference) initiatives (Harman, 1998). The best overall performance in routing tasks (47 topics) in the recent TREC-6 experiments[2] achieved an average precision of 42% (Singhal, 1998). A variety of techniques is usually employed to generate routing queries. Generally, this problem is treated as a classification problem, whereby the goal is to assess the 'degree of membership' of documents in the standing queries. Schütze, Hull and Pedersen (1995)have compared the utility of four different techniques in routing: linear discriminant analysis, logistic regression, a two-level linear neural network and a three-level nonlinear neural network. These techniques were compared to a simple baseline system developed using an algorithm similar to the Rocchio (1971)query expansion. For feature representation in the experimental levels they applied LSI, an optimal term selection method using χ^2-test and a combination of these two approaches. This research found that with reduced dimensions using the LSI, the linear discriminant analysis produced the best results. The neural network levels were able to closely approximate the linear discriminant analysis performance when the LSI representation was used. However, the neural network levels produced slightly better overall results with the other representations. Little difference was observed between the linear and the nonlinear neural network methods, which according to the authors, was possibly due to the lack of training data to learn complex models.

3.1. Summary and research problem

We found a number of IF services represented profiles as conventional search-queries and they used the same type of representation for profiles and documents. We have also found that IF services conducted classification in a wide variety of ways. For example, classification involved identifying for each document one of two nontopical binary classes: relevant or nonrelevant. In more sophisticated systems classification also involved establishing membership in multiple nontopical classes, whereby the classes were document folders (e.g. review immediately, see later, delete) or a set of actions the system took on behalf of the user. In these systems, sometimes topical classification played an implicit role, but more frequently the nontopical classification was the main and ultimate focus. Certain systems explicitly and exclusively attempted topical classification of documents (e.g. Construe). Generally, they implemented automatic semantic tagging as a way to make indexing efficient, extract 'new' information from incoming streams, or support direct queries by users based on assigned classification labels. Although our approach shares certain similarities to the diverse set of approaches reviewed here, there are some subtle and important differences when they are

[2] It should be noted that the recent TREC-6 included a special track called filtering. Filtering was treated similarly as routing, except the output of the matching process were binary retrieval decisions (yes/no), not a ranked list of documents. Among the 10 participants, the AT&T team, using a variant of their routing algorithm, produced the best results (Singhal, 1998). A detailed and complete analysis of the results can be found in Hull (1998).

compared to our assumptions, the architecture of our filtering model and ultimately the design of our system. In our approach to filtering, classification is not the terminal operation but merely an intermediate step. We do not equate classification with filtering; rather, in our filtering model, we use classification in two particular ways: (1) to group documents into a fixed set of classes before they are sorted and (2) to fix the structure of the interest profile based on the set of classes making up the classification scheme. In this research, classification is not a grouping or clustering of like-entities in the broad-sense; instead we take a narrow view of classification. To us it is *grouping of document entities according to externally defined topical divisions*. This is admittedly a conventional or 'library-oriented' use of the term and as such it should be understood as independent of other operations that may be conducted beyond this basic grouping event. The resulting filtering model separates the classification layer from the other components providing certain advantages, including leveraging of established classification vocabularies.

To conclude, IF appears to be a vibrant and growing area of research. In reviewing the literature, interesting links among representation, classification and interest profiles have emerged. To gain a broader perspective of filtering than what is offered here, the reader may consult (Oard, 1997) and the related web site that tracks latest development in the area (www.clis.umd.edu/dlrg/filter). In this research, our focus is on the classification module, concerning its influence on the overall dynamics of a multilevel filtering environment. Particularly, given the choices among different classification techniques, we are interested to know how a supervised learning method performs in comparison to a method directly based on human-generated classification information. Subsequently, our goal is to clarify the relationship between document classification performance and the utility of profiles. We especially wish to determine how document classification accuracy can influence profile quality and how profile quality in turn can affect filtering performance.

4. SIFTER system and supervised classification

In SIFTER the major components are implemented as separate modules. The modularity of the system supports convenient substitution of different representation, classification and profile-management techniques and it permits analysis of their individual or combined influence on the filtering performance. In several previous articles, description of the major features of SIFTER along with performance analysis have been presented. An in-depth treatment of the profile management component can be found in Mukhopadhyay et al. (1996) and a sub-component that detects abrupt shifts in user's interest is discussed in Lam, Mukhopadhyay, Mostafa and Palakal (1996). The representation and classification components receive primary emphasis in Jacob et al. (1997); Mostafa et al. (1998). In this section, therefore, only a brief overview of the system is offered. It is followed by a description of the neural network and transaction-oriented components of SIFTER.

4.1. Representation, classification and profile learning

There are three modules in SIFTER responsible for performing the primary functions

associated with filtering. Initially, when the system is invoked, new documents are read in and converted to more compact structures to permit more efficient interpretation of their content. This process is called representation. Subsequently, each document is assigned a class label by the second module known as the classifier. Based on the classification, documents are then ranked and presented to the user. This function is performed by the profile learning module, which is responsible for maintaining the profile up-to-date and applying the profile to rank incoming documents. For comparative analysis, two versions of SIFTER were developed: a baseline version and a neural network version. The major difference between the two versions was in the classification module. Below, the three modules are described. Differences between the two versions are pointed out whenever appropriate.

4.1.1. Representation

In both SIFTER versions, the representation process involves converting documents to more efficiently parsable structures that are treated as input by the document classification module. A document structure is a vector $V = (W_1, W_2, \ldots, W_n)$, where each element W_k represents a weight associated with a discriminatory concept found in the document. To convert a document to a vector V, initially a thesaurus is applied. The thesaurus is an array of elements, with each element containing a value-pair: an atomic token (a single term or word) and a unique numeric identifier. Each document vector length is equivalent to the total number of unique tokens in the thesaurus and each element in the document vector is used to store a weight value associated with the corresponding token. The $tf \times idf$ technique for document representation (Salton & McGill, 1983) is integrated into the representation stage due to its successful and wide-scale application in the past. This required the frequencies of significant terms in documents, as identified by matching with the thesaurus, to be stored and eventually modulated to generate the appropriate weight values. Each term frequency, T_{ik}, for the document i and the term k, is computed during the online filtering mode and stored. The modulation of the frequency involved multiplying with a 'correcting' factor I_k, the inverse document frequency. It is calculated using $\log(N/n_k)$, where N is the total number of documents in a representative document base in which n documents contain the term k. The correcting factors corresponding to each term of the thesaurus is produced beforehand in a batch operation using a large training set of documents collected from the source. Applying a large document base to generate the correcting factors makes the representation process less vulnerable to drastic changes in size and content of the document stream.

4.1.2. Classification

The next stage involves classification of document vectors. In both SIFTER versions, a classification procedure is executed to identify for each vector V a single class from a pre-established set of classes. The size of the set of pre-established classes remains fixed during the online filtering mode, thus affording predictability and stability in the process. Establishing an appropriate set of classes (to be used by the IF systems) is a separate step from the actual document classification step. The class set establishment step is conducted as an offline procedure, while by necessity the classification step is an online operation.

Both versions of SIFTER use a set of class vectors to support classification operations. The class vectors are similar in representation as documents. Semantically, however, each class

vector C_j should be viewed as a high-level grouping of concepts so that they form sub-areas or classes in the domain defined by the thesaurus. In both SIFTER versions, each element in the vector C_j represents a particular token identifier in the thesaurus and the dimension of C_j is equal to the number of unique tokens in the thesaurus. For the class 'Cell Death', for example, the elements corresponding to these tokens in the class vector would have suitable high weights and the rest of the elements would be set to zero. This is a manual way of producing classes and it was applied in this research. A subset of the Medical Subject Headings (MeSH) produced by the National Library of Medicine was applied to create a set of class vectors for both versions of SIFTER.

In the baseline SIFTER version, during the online filtering mode, the classification was conducted by using the distance measure: $1 - \sum_{k=1}^{t}(V_k C_k)/\sqrt{(\sum_{k=1}^{t} V_k^2)(\sum_{k=1}^{t} C_k^2)}$. This equation is derived from the cosine similarity measure proposed by Salton and McGill (1983). In it V_k represents a document vector and C_k is a class vector from the class set and the class that generates the smallest distance is assigned to the document. To support the neural network version, the class set was used as the target class set to train a neural network document classifier. The training was performed as an offline procedure. A SIFTER version, with the trained neural network integrated into it was subsequently used for conducting the filtering sessions. We elaborate on the neural network component in Section 4.2.

4.1.3. Profile learning

The purpose of the third and the final module, called the profile learning module, is threefold: rank documents for presentation, collect feedback and adjust the profile. This module was identical in both versions of SIFTER. The profile learning module has exclusive access to the interest information. The interest information consists of two vectors of the same length as the fixed number of classes used by SIFTER. The first vector is an estimated profile, formally referred to as an estimated interest probability vector: $\hat{\Theta} = (e_1, \ldots, e_m)$. In this vector, each e_c is the estimated interest value in the class c and there is a maximum of m classes.

Generally, the ranking of documents is very much dependent on the profile vector (Θ). In the first few sessions, however, there is little information about the user's interest. During this period if the user follows the expected behavior of browsing only the first few documents in the list, the profile would be skewed toward arrival order of documents (as only the first few documents would tend to receive feedback). To equalize the chance for all the classes to be exposed or browsed, at the early period, another vector called the action probability vector is used. The action probability vector is of the form: $p = (a_1, \ldots, a_m)$. In this vector, each a_c is the probability that the class c should be ranked as the first class while determining the ranking of documents based on their class membership. This second vector is of the same length as the maximum number of classes m. In the first session, all the elements in the p vector are set to be equal to each other, i.e. each a_i, ($i = 1, \ldots, m$), is set to $1/m$ and all the elements of Θ are set to zero. The class ranked first is always selected based on the p vector. Subsequent classes are selected based on the Θ vector. As the probabilities in the p vector are equal at the beginning, all the classes have equal chance of being ranked first.

The two vectors are updated at the end of each session based on the user's feedback. The p vector eventually converges to a top class. This top class in p ultimately corresponds to the top

class in the Θ, hence, the class information in p becomes redundant. The update algorithm will be presented shortly. Below, we describe the ranking algorithm.

The goal of the ranking algorithm is to sort documents according to classes, whereby the order of the classes is directly established based on the interest information. The steps are straightforward: determine the top class from p and the order of the rest of the classes in Θ and then sort the documents accordingly.

1. The top class from p is selected probabilistically. The values in the p vector, each representing a class, are treated as a probability distribution. For an example, assume the p vector is [0.2, 0.4, 0.1, 0.3]. In each session a pseudorandom number, PRN, of uniform distribution is generated between 0 and 1.0. Class 1 (represented by the first element in p) will be selected if PRN is between 0 and 0.2, class 2 will be selected if PRN is between 0.2 and 0.6 (as the upper probability range is $0.2+0.4=0.6$), class 3 will be selected if PRN is between 0.6 and 0.7 and finally class 4 will be selected in the case of all other values of PRN.
2. Then a simple ordering vector, $o(s)$, with the same number of elements as the number of classes is generated in each session s. In this vector the first element contains the top class selected in step 1. The rest of the classes are selected from the vector Θ, according to their magnitude in Θ. Specifically, if the class has not been selected already, the second element selected for $o(s)$ is the largest class in Θ, the third element selected is the second largest class in Θ and so on.

Assuming, in a session s there are five documents: $d_1(s),\ldots,d_5(s)$. These documents are sorted and presented according to the order of the classes in the ordering vector $o(s)$. So, documents from the first class in $o(s)$ would be presented first, documents from the second class in $o(s)$ would be presented next and so on.

At the end of each session, the two vectors, $\hat{\Theta}=(e_1,\ldots,e_m)$ and $p=(a_1,\ldots,a_m)$ are updated based on the user's feedback. A feedback collected for a document is applied to the class assigned to that document. The steps below are followed:

1. Suppose a feedback f_c for a class c is received on session s, the element e_c in the vector $\Theta(s)$ is updated as follows: $e_c(s+1)=((e_c(s)\times s)+f_c)/(s+1)$, where f_c is 1 or 0 denoting a positive and a negative feedback respectively. Essentially it maintains the interest value for a class as the running average of the feedback given by the user to that class.
2. Let c'(s) be the class corresponding to the maximum element in the vector $\Theta(s)$ in session s. The elements a_c of the action probability vector $p(s)$ are updated as follows:

$$\begin{aligned} a_c(s+1) &= a_c(s)+\lambda(1-a_c(s)) && \text{if the } c^{\text{th}}\text{class} = c'(s),\\ &= a_c(s)-\lambda a_c(s) && \text{otherwise} \end{aligned}$$

where λ is a suitably chosen learning rate and $0<\lambda<1$. This way of updating has the property that the p vector is moved by a small distance towards the optimal unit vector, where value of one class emerges as the highest value close to 1 and rest of the class values tend toward 0. Update of p above also shows that it is conducted based on the interest value estimates in Θ and typically the highest values in both vectors would correspond to the same class after a period of time.

4.2. Document classification using a neural network

Extensive literature exists on how neural network algorithms work and their general utility. Weiss and Kulikowski's (1991) book is a good source on classification using neural networks and associated machine learning approaches. For a thorough mathematical treatment of neural networks, the reader may consult Hertz, Krogh and Palmer (1991). Here, we do not offer general background information on neural networks. Instead, we present a description of the specific technique as applied in this research.

The neural network designed for our task is a basic multilayer feed-forward network with adaptation to suit the text classification problem. Our network consists of three layers, namely, an input layer, a hidden layer and an output layer. The input layer captures a representation of a document. The output layer captures an assignment of a specific class to the document. Each document is denoted by a document vector whose elements are some chosen features. To control the amount of training period, we make use of the tuning set approach. The labeled training documents are divided into two parts, namely the *network training set* and the *tuning set*. We remove the class labels from the actual documents and create separate class label vectors. The idea here is to make the neural network learn to associate unclassified documents with their correct class labels. Therefore, each labeled document is represented by a unique document vector and it has one corresponding class label vector. The neural network is presented with the documents in the training set. Each document vector is applied to the input layer of the network. The input activation of each input unit is propagated forward through the network and produce an output vector. By comparing the output vector with the class label vector, we can compute the error at the output layer. Back-propagation technique is employed to adjust the weights of the network so as to reduce the error. After all the training documents have been presented to the network, the network is then evaluated by the labeled documents in the tuning set. Termination of the training is determined by the behavior of the classification performance on the tuning set. The details of our neural network approach is presented below.

4.2.1. Network architecture

Suppose each document is represented by n_1 features. The input layer of the neural network has n_1 nodes, namely, $U_1,\ldots,U_{n_1}$, with the node values $u_1,\ldots,u_{n_1}$ respectively. Each node corresponds to a feature weight and takes on a real number value. The hidden layer has n_2 nodes and the nodes are labeled as $U_{n_1+1},\ldots,U_{n_1+n_2}$, containing the node values $u_{n_1+1},\ldots,u_{n_1+n_2}$ respectively. Suppose there are m possible classes. Let $C_1,\ldots,C_m$ denote the class label vectors. The output layer has m nodes labeled as $U_{n_1+n_2+1},\ldots,U_{n_1+n_2+m}$, with the node values $u_{n_1+n_2+1},\ldots,u_{n_1+n_2+m}$ respectively, where the node $U_{n_1+n_2+i}$ corresponds to the class C_i. There are links connecting each input node to each hidden node and each hidden node to each output node. A weight, W_{ij}, is associated with each link connecting the node U_i and U_j, where $i > j$. In our experiments, n_1, n_2 and m are set to 43, 85 and 15 respectively.

4.2.2. Training algorithm

1. Weight Initialization — Every weight, W_{ij} is initialized randomly by a small real number.
2. Node Assignment — For each training document, perform steps 2–5 (inclusive) until all training documents are processed. Suppose the next training document has a representation $(d_1,\ldots,d_{n_1})$. An input node U_i is initialized with the value of d_i in the document vector. If a document belongs to a particular class, say the i-th class, it is modeled by setting C_i to 0.9 and all the remaining elements in the class vector are set to 0.1.
3. Forward Propagation — We employ the sigmoid function as the activation function of each node in the network. Starting with the hidden layer, we compute the weighted sum, S_i and activation, u_i for each node in the hidden and output layer as follows:

$$S_i = \sum_{j:j<i} W_{ij}u_j$$

$$u_i = \frac{1}{1+\mathrm{e}^{-S_i}}$$

4. Backward Propagation — Starting with the output layer, we visit each node in the hidden and input layers and compute δ. For a node, U_j, in the output layer ($j = n_1 + n_2 + i$), δ_j is determined by:

$$\delta_j = (C_i - u_j)u_j(1-u_j)$$

For a node, U_k, in the hidden layer, δ_k is determined by:

$$\delta_k = \left(\sum_{j:j>k} W_{jk}\delta_j\right)u_k(1-u_k)$$

5. Weight Updating — Each weight, W_{ij}, is updated as follows:

$$W_{ij} = W_{ij} + \rho\delta_i u_j$$

where ρ is the learning rate. We set ρ to be 0.01 in our experiments.

6. Error Rate Determination — After completing steps 2–5 for all the training documents, the current trained network is used to classify the documents in the tuning set. The classification error rate, defined as the ratio of the number of incorrectly classified documents to the number of total tuning documents is calculated. One iteration of the steps 2–6 (inclusive) is considered as one epoch. Then the algorithm goes to Step 2 for the execution of another epoch.

The tuning set classification error rate was recorded at each epoch. In the early period of training, the classification error rate gradually decreased and then it remained stable for some time. Later in the training, when noticeable rise in the error rate was detected the training phase was terminated in order to avoid over-fitting. After the training phase was finished, the weights on the network were stored, to be applied during online classification. In other words, this trained network was interpreted as the document classifier.

The number of documents in the training and tuning set was 4000 and 2000 respectively. The feature set was based on a thesaurus that consisted of 43 unique tokens derived from cell biology terms in the MeSH headings. To evaluate the performance of the network, we conducted classification on documents from a *testing set* that are different from the training documents (described later). Each test document vector was applied to the input layer. The class of the document was determined by selecting the one corresponding to the maximum output unit. Specific metrics, including classification recall and precision, were used to measure the performance of the neural network approach.

The neural network approach applied here requires a priori availability of a training set with known classes for each case. This approach should be contrasted with two other established approaches: reinforcement and unsupervised learning (Bigus, 1996). In certain applications little or no immediate information is available a priori about the relationship between cases and the prospective target classes. However, such information may be 'emergent' or may become available over time. For example, in SIFTER, as document relevance can only be judged after documents are reviewed by the user, the relevance feedback is only available a posteriori and therefore is more suitable as a reinforcement signal. In certain other situations, however, the target class information may be not be available a priori or even during the operational mode. An example may be an application that classifies images of some unknown terrains or landscapes. By applying unsupervised learning, a clustering procedure can group such data into homogeneous regions. A particular advantage of the unsupervised learning procedure in document classification is that it requires comparatively little human intervention as pre-classified documents are not needed. In our most recent work in this area, an unsupervised learning approach was applied using the *Maximin Distance* clustering technique (Tou & Gonzalez, 1974) as an implementation for the document classification module. This method, however, produced mixed classification results and degraded filtering performance. In searching for a better approach, we noted that immense amount of monetary and intellectual effort already has been invested in developing classification schemes and in manually classifying millions of documents (e.g. the Medline database). We also found that the underlying model of SIFTER does permit offline learning of document classification and convenient integration of the learned network for online classification. This motivated us to take advantage of the pre-classified medical documents as an authoritative training source for the supervised learning procedure applied here.

4.3. SIFTER operations and evaluation

SIFTER is a fully implemented filtering system. The core set of modules, the filtering engine, was written in C. The graphical user interface (GUI) was written in TCL/TK. SIFTER has been tested in the Sun Solaris and the Hewlett Packard UNIX environments. SIFTER uses a message-processing convention to provide filtering service. Given a designated document 'bin' on the system, invocation of SIFTER results in presentation of newly added documents sorted according to the profile. The SIFTER GUI allows the user to view document headers, open a different window to browse the document content, provide relevance feedback, generate a graph showing the learning state of the system and display the current interest profile. SIFTER also supports an autonomous filtering mode, whereby filtering can be conducted without

ongoing and explicit user intervention. In this mode, an interest profile (Θ) can be directly entered into the system and several relevant parameters (e.g. interest in various areas) can be preset. The user-supplied profile is created by assigning to each class in Θ a value between 0–1.0 (inclusive) to signify the level of interest. In this mode, determination of document relevance and generation of feedback are directly derived from the profile Θ. This is conducted probabilistically so that documents belonging to classes with low interest value (near 0) in Θ would be much less likely to be identified as relevant than would documents in classes with higher interest value (close to 1.0). The internal profile, Θ is still maintained in the autonomous mode, as it was designed to support adaptation to changing filtering demands. SIFTER is also capable of logging numerous types of transaction data. For example, it can log document identifiers, the class label and the final ranking of documents. It can save the statistical data relevant to each invocation as formatted text files in a pre-designated directory. A more detailed description of SIFTER's major features and a usability analysis can be found in (Mostafa, Mukhopadhyay, Lam & Palakal, 1997).

5. Research methodology

Broadly, our research goal was to analyze how the quality of document classification may influence filtering outcome. Expressed in an empirical framework, in this research the independent variable was classification quality and the dependent variable was filtering outcome. Several measures were used to calculate the independent and dependent variables. As a way to asses the performance of a system employing the neural network approach, it was compared with a baseline system. Hence, measures associated with classification and filtering variables were applied to analyze the performance of both types of systems and results were comparatively analyzed whenever appropriate.

5.1. Classification quality and filtering performance

Several measures have been proposed in the literature for calculating classification effectiveness (Lewis, 1995; Hull, 1998). In this research, we used three measures proposed by Lewis (1995) for calculating classification effectiveness. We equated classification effectiveness with classification quality. The proposed measures are mainly based on classical information retrieval performance measures. All three measures assume that document classification has been judged by an expert, according to which the system's classification is then measured. In this research we treated original classification labels assigned by Medline classifiers as equivalent to an expert's classification judgment for each document. The following parameters are relevant to the three measures:

- a = total number of documents classified into a class that agree with expert's judgment
- b = total number of documents classified into a class that do not agree with expert's judgment
- c = total number of documents *not* classified into a class that according to the expert should belong to the class

- d = total number of documents *not* classified into a class that according to the expert do not belong to the class

The first measure is: $recall = a/(a + c)$; it is the proportion of class members as determined by the expert that the system accurately placed in the class. The second measure is: $precision = a/(a + b)$; it is the proportion of the documents placed in a system's class that are accurately placed. The final measure incorporates both error of commission and error of omission into a single error measure. It is: $error\ rate = (b + c)/(a + b + c + d)$; the b parameter stands for those documents that the system mis-classified into its class and the c stands for those documents that the system missed. In our research, the denominator stands for all the documents that the system had to classify and it plays a 'normalizing' role to make possible the comparison of error rates across classes.

We analyzed classification performance of all classes used by the NN classifier. This allowed us to derive an overall classifier-level assessment. However, to identify relationship between the independent variable (classification quality) and the dependent variable (filtering outcome), we concentrated on the performance of individual classes. More specifically, we concentrated on the error rate, as it combines classification performance into a single value and we ranked the NN classes according to this measure. Based on the ranking, we created three profiles using those classes that had the lowest error rate and three other profiles using the classes that had the highest error rate. Subsequently, we conducted filtering sessions using these profiles to establish their influence on the filtering performance.

Although all the documents are presented in every session in a ranked manner[3], the assumption is that a user would generally give more 'attention' to the documents ranked at the top and would expect the documents relevant to their interest to be typically highly ranked. For the purpose of quantifying the quality of filtering, we assumed that a user would generally look at the top 10 documents; hence, the 10th document was set as a cutoff point for calculation purposes. As we presented earlier, the determination of relevant documents is based on the profile information, Θ, provided by the user. As part of transaction-logging, SIFTER kept a count of total number of relevant documents it placed among documents ranked between 1–10 (inclusive) out of all the documents presented in every session. This value, called Filtered Documents until Cutoff (FDC) was collected for each session and it was used as one of our dependent variables. To arrive at a cumulative performance estimate at every session, another related parameter called Filtering Precision (FP) was used. FP is the ratio of total number of relevant documents to the total number of top-10 documents presented over n sessions. Specifically, FP was calculated as a running average of FDC in every session using the formula: $(\sum_{i=1}^{n} FDC_i)/(10 \times n)$, where n is the session number.

To conduct the experiments, a document set was created containing 7500 items. These were bibliographic records each consisting of document title, author, abstract and classification labels. They were downloaded from the Medline database (www.ncbi.nlm.nih.gov/PubMed/) and they dealt with the area of cell biology. Among the numerous MeSH headings concerning

[3] The ranking of documents is always based on the profile information and this way of ranking is treated as *filtering* in this research. That is, filtering, did not involve exclusion or pruning of documents, rather it involved sorting of incoming documents based on the profile. The document ranking algorithm is described in Section 4.1.

cell biology, the document set was limited to 15 main areas (Table 1). To have equal representation for all the classes in the document set, Medline was searched using the headings and for each class 500 documents were retrieved. It should be noted that, formally, MeSH is a categorization system as Medline records generally are assigned multiple MeSH headings. However, for the purpose of this research only a single MeSH heading, relevant to one of the 15 cell biology areas, was maintained per document. Thus, each heading was treated as a class and the 15 headings were treated as the classification scheme. The 7500 document set was then divided into two groups: (1) a 6000-document (400 documents/class) training and tuning set and (2) a 1500-document evaluation set (100 documents/class). The first set was used for neural network class acquisition and it was also treated as the base file in calculating the IDF parameter for use during the online filtering mode.

Two versions of SIFTER were prepared for conducting online evaluations. One version used the longer 43-token thesaurus. This thesaurus was created by using variants of all the terms in the 15 headings and significant lowerlevel (more specific) headings in MeSH. This SIFTER version also had integrated into it the trained neural network to conduct classification (henceforth referred to as SIFTER-NN). For the baseline version (henceforth SIFTER-BASE), we created a 21-token thesaurus containing only the terms found in the 15 headings (Table 1). A corresponding group of 15 class vectors was then created with the appropriate elements set to high weights and these class vectors were entered into the SIFTER-BASE. The distance measure described before, $1 - \sum_{k=1}^{t}(V_k C_k) \,/\sqrt{(\sum_{k=1}^{t} V_k^2)(\sum_{k=1}^{t} C_k^2)}$, was used in SIFTER-BASE to identify the appropriate class for each document. For compatibility with the type of classification desired, we also modified the evaluation data set used as input for the two different SIFTER versions. The input evaluation set of the neural network version contained only document number, title, abstract and author information. Whereas, the input evaluation set of the baseline version contained only a document number and the associated class. Apart from these basic differences both SIFTER versions performed all their other functions in an

Table 1
MeSH headings

CELL ADHESION
CELL COMMUNICATION
CELL DEATH
CELL MOVEMENT
CELL SURVIVAL
ENDOCYTOSIS
ANTIBODY FORMATION
AUTOIMMUNITY
IMMUNOCOMPROMISED HOST
CYTOTOXICITY IMMUNOLOGIC
IMMUNE TOLERANCE
IMMUNITY CELLULAR
REGENERATION
EVOLUTION
COMPLEMENT ACTIVATION

identical fashion. During the filtering mode, the same number and same sequence of documents were processed by both versions of SIFTER.

6. Research results and analysis

We wished to conduct several filtering sessions using the evaluation document set. Hence, for each session the maximum number of new documents presented was set at 30 documents. This allowed for a total of 48 sessions to be executed, using up to 1440 documents from a total of 1500 in the original set. This 1440 document-set was used for both classification and filtering analysis. We begin by describing and comparing the classification results of the neural network version SIFTER-NN and the baseline version SIFTER-BASE. It is followed up with results of filtering performance of SIFTER-NN. This performance is then analyzed in relation to the filtering performance attained from the SIFTER-BASE.

6.1. Classification Experiments

As designed to do so, the baseline version, SIFTER-BASE, accurately classified all the documents. The neural network classification results varied across classes. The average number of documents classified per class by SIFTER-NN was 96, with a relatively high standard deviation of 33.8. Whereas, in the SIFTER-BASE, the classification average was 96, with a standard deviation of 3.7. The average recall across SIFTER-NN classes was 0.7 and the average precision was 0.74. For certain SIFTER-NN classes relatively large difference was observed between recall and precision. For example, for the SIFTER-NN class IMMUNOCOMPROMISED HOST, the recall was 0.84, but its precision was 0.44. Another example concerns the SIFTER-NN class IMMUNE TOLERANCE, which achieved a precision of 0.77 and a recall of 0.51. The error rate results revealed the variability across SIFTER-NN classes more clearly. The average error rate of SIFTER-NN was 3.89%. The top-performing seven SIFTER-NN classes, with an average error rate of 1.6% (Table 2), correctly classified 584 documents out of 672 documents that belong to those classes (87%). On average, 84 documents were correctly classified into the topperforming seven SIFTER-NN classes, with a standard deviation of 7.3. The bottom eight classes in SIFTER-NN had an average error rate of 5.87% and this higher error rate was reflected in their classification results. The bottom eight SIFTER-NN classes correctly classified 436 documents out of 768 that belong to those classes (57%). On average, 55 documents were correctly classified into the bottom eight SIFTER-NN classes, with a standard deviation of 13.27. Low error rate of classes is indicative of more consistent and reliable performance of classes and conversely high error rate of classes can lead to less consistent and unreliable performance. Therefore, error rate is a relatively strong indicator of classification performance across classes.

6.2. Filtering experiments

To perform the filtering experiments, two groups of profiles were created. The high accuracy profile group, HIGH, was based on four classes demonstrating the highest classification

Table 2
Documents classification results (ranking according to NN error rate)

NN rank	Class	Base total	NN total	NN recall	NN precision	NN error (%)
1	COMPLEMENT ACTIVATION	99	100	0.95	0.95	0.63
2	ENDOCYTOSIS	99	91	0.90	0.98	0.69
3	EVOLUTION	99	90	0.86	0.95	1.18
4	REGENERATION	95	76	0.77	0.97	1.6
5	CELL DEATH	94	99	0.85	0.80	2.29
6	AUTOIMMUNITY	96	95	0.81	0.82	2.43
7	CELL ADHESION	90	109	0.9	0.74	2.57
8	ANTIBODY FORMATION	86	68	0.54	0.69	4.17
9	IMMUNE TOLERANCE	99	66	0.51	0.77	4.38
10	CYTOTOXICITY IMMUNOLOGIC	96	68	0.46	0.66	5.14
11	CELL COMMUNICATION	96	67	0.45	0.65	5.21
12	CELL MOVEMENT	98	71	0.45	0.63	5.49
13	IMMUNITY CELLULAR	99	100	0.57	0.57	5.90
14	IMMUNOCOMPROMISED HOST	97	184	0.84	0.44	8.13
15	CELL SURVIVAL	97	156	0.67	0.41	8.54
	Average Error Rate					3.89

accuracy in SIFTER-NN: COMPLEMENT ACTIVATION (ranked 1), ENDOCYTOSIS (2), EVOLUTION (3) and REGENERATION (4). The average error rate of the classes in the HIGH profile group was 1%. The low-homogeneity profile group, LOW, was based on four classes with the lowest classification accuracy rates in SIFTER-NN: CELL SURVIVAL (ranked 15), IMMUNOCOMPROMISED HOST (14), IMMUNITY CELLULAR (13) and CELL MOVEMENT (12). The average error rate of the classes in the LOW profile group was 7%. In each profile, the four classes were assigned different values to reflect discriminatory, moderately discriminatory and nondiscriminatory levels of interest. The rest of the classes in each profile were assigned a value of 0. In this way, for each profile group three different profiles were developed. The three profiles in the two groups and their corresponding interest values are shown in Table 3. Filtering results of the three profiles in each group were averaged to generate more realistic performance figures.

For comparative analysis, these profiles were entered into both SIFTER-NN and SIFTER-BASE as Θ and 48 filtering sessions were conducted for each profile. It should be noted that as the SIFTER-NN and SIFTER-BASE shared the same class set, the corresponding profiles used

Table 3
Profiles used in filtering experiments

	High accuracy group				Low accuracy group			
Class ranking	1	2	3	4	15	14	13	12
Discriminatory profile	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Moderately-disc. profile	1.0	1.0	0.7	0.7	1.0	0.7	0.7	1.0
Non-disc. profile	0.65	0.6	0.7	0.6	0.65	0.7	0.6	0.6

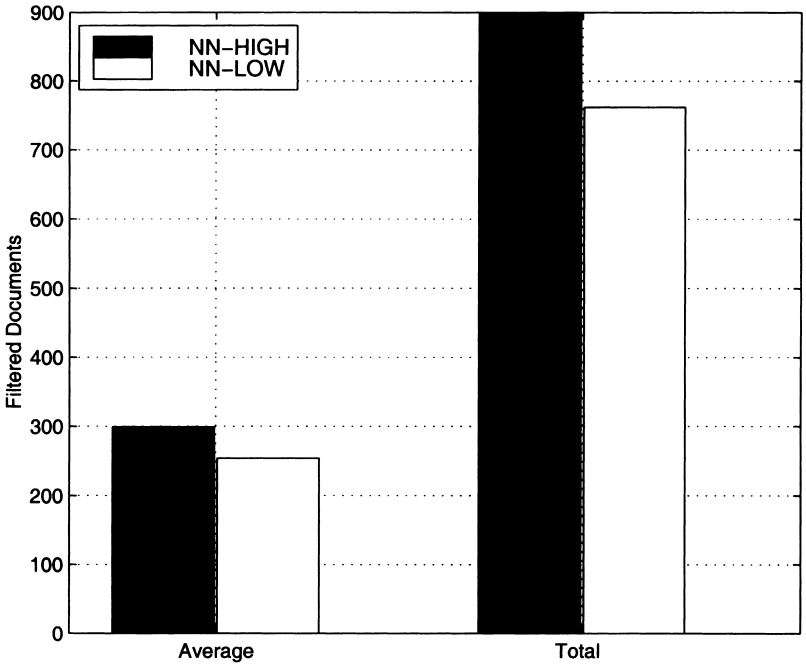

Fig. 5. Filtered documents (FDC) in SIFTER-NN.

with these two system versions were identical in content. Also, in SIFTER-BASE, there was no difference in error rate among profiles as it applied the baseline classification scheme and it directly used classification information maintained in documents. We begin with a presentation of filtering performance of SIFTER-NN comparing the results of the two profile groups, HIGH with LOW, in this system. Later SIFTER-NN filtering is analyzed in relation to the baseline system.

In SIFTER-NN the HIGH profile group outperformed the LOW profile group in filtering documents. This result was evident in terms of both the average FDC[4] of all three profiles in each group and the total FDC in each group (Fig. 5). The total FDC for the HIGH profile group was 898, whereas the total FDC for the LOW profile group was 762. This means that 136 relevant documents were not detected when the LOW profiles were used. The superiority of the HIGH profiles was also evident when the FP[5] results were considered (Fig. 6). The FP was averaged across the three type of profiles for the two groups. In the initial 1–20 sessions, there is a steady rise in performance by the HIGH group reaching to 0.6 precision by about 25th session. Whereas, in the LOW group there is considerable oscillation at the early period and it settles at a high of 0.53 precision at approximately the 30th session. Generally then it can be said that the HIGH group provided a better and more predictable performance.

[4] FDC, (Filtered Documents until Cutoff), was described in Section 5.1 as the count of relevant documents appearing in top10 in each session.

[5] FP (Filtering Precision), calculated as the running average of FDC (Section 5.1).

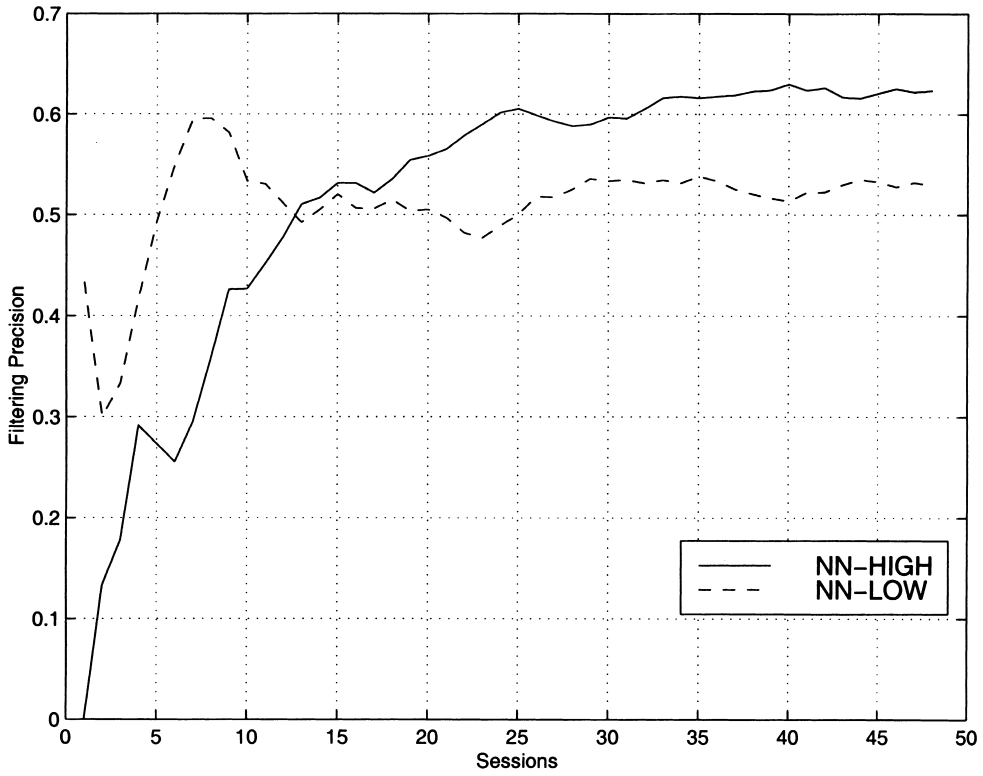

Fig. 6. Average filtering precision (FP) in SIFTER-NN.

To arrive at an overall assessment of the neural network approach, the filtering performance of the SIFTER-NN system was compared with the SIFTER-BASE system. It was found that SIFTER-NN closely approximated the performance of the baseline system. The grand total FDC of SIFTER-NN, calculated by summing the FDC of all the profiles, was 1660, whereas the grand total FDC of SIFTER-BASE was 1682 (Fig. 7). The average FDC/profile for SIFTER-NN was 277 and for SIFTERBASE was 281. SIFTER-BASE, therefore, showed only a slight edge in performance over all six profiles. The difference in FP between the two systems was also very narrow, with SIFTER-BASE again showing only a slim margin of superiority (Fig. 8). A Pearson product moment correlation (r) analysis was conducted to more firmly establish the degree of association between the two average FP trends. The resulting r of 0.98 showed that there was a strong positive correlation and it was significant ($p < 0.01$).

6.3. Discussion

The classification accuracy achieved through the neural network approach can be characterized as adequate with some room for improvement. Generally, SIFTER-NN's classification performance demonstrated high variance. Performance varied across classes in terms of recall, precision and error rate. The three-pronged approach to measuring classification was found to be useful. The difference between recall and precision, for individual

classes, ranged from 0 to as high as 0.4, demonstrating that a single measure alone may not capture the true 'classification capacity' of a class. The recall may be high but the precision may be relatively low and vice versa. The capacity of a class to simultaneously attract correct documents and avoid incorrect documents, improves both its recall and precision. This capacity is numerically captured in the error rate measure. Therefore, the classes with low error rate generally demonstrated higher and more similar recall and precision (e.g. top classes in Table 2). Conversely, classes with high error rate had lower and more dissimilar recall and precision.

The variability in classification performance influenced filtering in predictable ways. It was clearly established that higher classification accuracy produces superior filtering performance and conversely lower classification accuracy can lead to degraded performance. In SIFTER-NN, profiles based on the HIGH accuracy classes outperformed the profiles that were created using the LOW accuracy classes. The inability of the system to accurately match documents to its classes lead to misranking of documents, thus lowering the filtering performance. Overall, however, SIFTER-NN was able to closely approximate the performance of the baseline system. These findings have several general implications for IF system design. It seems feasible that an IF system based on the neural network approach can provide real-time and effective service with the special caveat that the system should make explicit the performance potential of the profiles in use. Accuracy rates of classes should be displayed in the user interface in an ongoing fashion to allow the user to assess the quality and the ultimate utility of their profiles.

When the filtering results attained by the supervised approach in this research is compared to the results attained based on an unsupervised approach in a previous research (Mostafa et

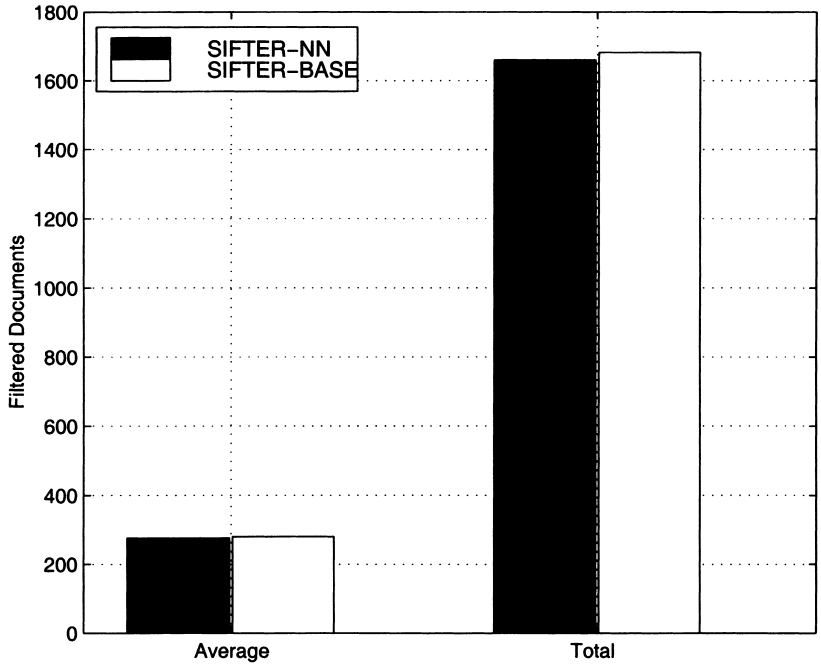

Fig. 7. Filtered documents (FDC) in SIFTER-NN and SIFTER-BASE.

al., 1998), it was found that the supervised approach can offer certain basic advantages. The unsupervised approach to classification is generally more openended in terms of type of classes the process generates. For example, with minimum control over the class establishment process, we found that the Maximin clustering algorithm produced classes that varied greatly in terms of their topical scope — from narrow to extremely broad (Mostafa et al., 1998). In contrast, in the supervised learning approach the class space can be fixed by the developer, with more consistent control over class scope. In SIFTER-NN the class space actually was identical to the humanestablished MeSH headings. The guarantee of strong semantic overlap with humanestablished schemes can make the classes more transparent to users, who rely on them to select and manipulate their profiles. Also, an established scheme and the availability of a large set of preclassified documents, make generation of classifiers that match the need of the user population more feasible.

The extremely rapid and frequent updates to the web document universe imply new solutions are necessary. It is possible to devise solutions that employ both automated and human classification in a *complementary* fashion. The documents processed by the neural network version of SIFTER did not have manually assigned class information. In other words the documents were treated as the vast majority of web documents that do not contain such information. Establishing manual document classification would take more time, thus

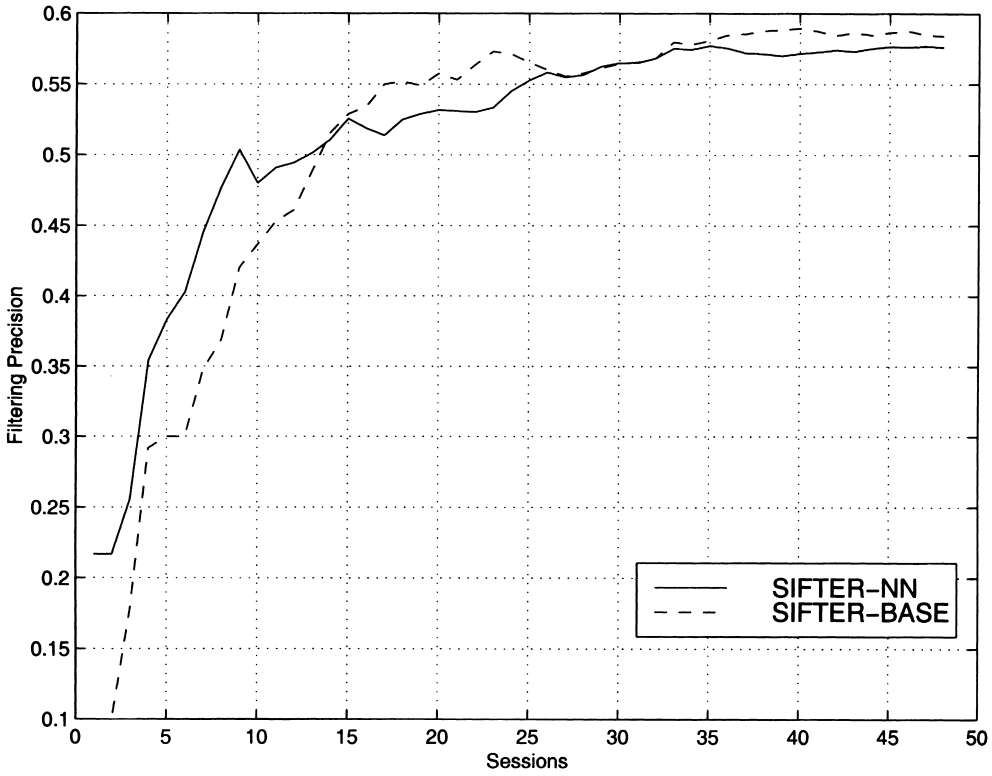

Fig. 8. Average filtering precision (FP) in SIFTERNN and SIFTERBASE.

increasing service latency. But, users may not wish to abandon manual classification altogether as it can serve an authoritative validation function. A sophisticated IF system perhaps can offer two levels of filtering service: coarse and refined. The coarse level by employing the neural network method can provide immediate and fast filtering service, with some loss in accuracy. However, over a period of time, after the documents are subjected to manual analysis and class assignment, a more refined set of filtered documents can be requested by the user. Outcome of the manual classification can also be used for ongoing monitoring and modulation of the automated level.

7. Conclusion and future work

In this research, we attempted to clarify the role of a document classification approach in filtering. It was found that, overall, classification using supervised learning produced satisfactory results, but classification accuracy rates were inconsistent across classes. To relate classification with filtering, different types of profiles were created based on classification performance distributions. It was demonstrated that the variability in classification performance does matter, as it was found that profiles of higher accuracy rates outperformed profiles of lower accuracy rates in filtering. However, the overall closeness in filtering performance of the supervised version with that of the baseline version suggested that the supervised version deserves more serious consideration.

In this research, the documents were selected from a specialized domain and contained limited information. Further, the user input into the classification process was not considered. As an extension of this study, we plan to more closely examine the cost in training (both manual and computational) across different types of domains and document formats when a supervised learning classification method is applied in filtering. The users can be involved in adjusting the classification learning process if they are allowed to suggest new classes or modify the thesaurus (feature-set). It would be interesting to establish the effect on classification and indirectly on filtering, when such user involvement is supported. Another extension that appeals to us is the development and evaluation of a hybrid filtering tool that can take advantage of both automatic and manual document classification.

Acknowledgements

The Purdue School of Science Computer Science Department and the Indiana University School of Library and Information Science supported the SIFTER project from its inception. Our special thanks go to Snehasis Mukhopadhyay for his help with algorithm and software development. This paper was completed at the Systems Engineering Department of the Chinese University of Hong Kong (CUHK), as part of a visiting scholar grant extended to the first author. We extend our thanks to the CUHK for its support of this collaborative project. Our thanks also go to Sigma Mostafa of Northwestern University, whose background in cell biology was especially helpful in vocabulary selection and in data analysis. Finally, both authors are grateful to the anonymous referee for constructive comments about the paper.

References

Bates, M. J. (1998). Indexing and access for digital libraries and the Internet: human, database and domain factors. *Journal of the American Society for Information Science*, *49*(13), 1185–1205.

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

Bigus, J. P. (1996). *Data mining with neural networks*. New York: McGraw-Hill.

Cheng, P. T. K., & Wu, A. K. W. (1995). ACS: an automatic classification system. *Journal of Information Science*, *21*(4), 289–299.

Dolin, R., Agrawal, D., El Abbadi, A., & Pearlman, J. (1998). Using automated classification for summarizing and selecting heterogeneous information sources. *DLib Magazine* (January). http://www.dlib.org/dlib/january98/dolin/01dolin.html.

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

Harman, D. (1998). The Text REtrieval Conferences (TRECs): providing a testbed for information retrieval systems. *Bulletin of the American Society for Information Science*, *24*(4), 11–13.

Hayes, P. J. (1992). Intelligent high-volume text processing using shallow, domain-specific techniques. In P. S. Jacobs, *Text-based intelligent systems: current research and practice in information extraction and retrieval* (pp. 227–241). Hillsdale, NJ: Lawrence Erlbaum.

Hertz, J., Krogh, A., & Palmer, R. G. (1991). *Introduction to the theory of neural computation*. Redwood City, CA: Addison-Wesley.

Hull, D. A. (1998). The TREC-6 filtering track: Description and analysis. Paper presented at the Sixth Text Retrieval Conference, Gaithersburg, MD. http://trec.nist.gov/pubs.html.

Jacob, E. K. (1991). Classification and categorization: drawing the line. In B. H. Kwasnik, & R. Fidel, *Advances in classification research*, *vol. 2* (pp. 67–83). Washington DC: Information Today.

Jacob, E. K., Mostafa, J., & Quiroga, L. M. (1997). An approach to the evaluation of automatically generated classification schemes. In P. Solomon, *Advances in Classification Research*, *7* (pp. 78–98). Medford, NJ: Information Today.

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

Jennings, N. R., & Wooldridge, M. J. (1998). Applications of Intelligent Agents. In N. R. Jennings, & M. J. Wooldridge, *Agent Technology* (pp. 3–28). Berlin, Germany: Springer-Verlag Chap. 1.

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM*, *40*(3), 77–87.

Lam, W., Mukhopadhyay, S., Mostafa, J., & Palakal, M. (1996). Detection of shifts in user interests for personalized information filtering. Paper presented at the 19th ACM International Conference on Research and Development in Information Retrieval, Zurich, Switzerland.

Larson, R. R. (1992). Experiments in automatic Library of Congress classification. *Journal of the American Society for Information Science*, *43*(2), 130–148.

Lawrence, S., & Giles, C. L. (1998). Searching the World Wide Web. *Science*, *280*(5360), 98–100.

Lewis, D. D. (1992). Text representation for intelligent text retrieval: a classification-oriented view. In P. S. Jacobs, *Text-based intelligent systems: current research and practice in information extraction and retrieval* (pp. 179–197). Hillsdale, NJ: Lawrence Erlbaum.

Lewis, D. D. (1995). Evaluating and optimizing autonomous text classification systems. Paper presented at the 18th ACM International Conference on Research and Development in Information Retrieval, Seattle, WA.

Lin, X. (1997). Map displays for information retrieval. *Journal of the American Society for Information Science*, *48*(1), 40–54.

Maes, P. (1994). Agents that reduce work and information overload. *Communications of the ACM*, *37*(7), 31–40.

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

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

Mostafa, J., Quiroga, L., & Palakal, M. (1998). Filtering medical documents using automated and human classification methods. *Journal of the American Society for Information Science*, *49*(14), 1304–1318.

Moukas, A. (1997). Amalthaea: Information discovery and filtering using a multiagent evolving ecosystem. *Applied Artificial Intelligence*, *11*, 437–457.

Mukhopadhyay, S., Mostafa, J., Palakal, M., Lam, W., Xue, L., & Hudli, A. (1996). An adaptive multilevel information filtering system. Paper presented at the 5th International Conference on User Modeling, KailuaKona, HI.

National Science Foundation (1988). Call for Proposal: Knowledge and Distributed Intelligence. http://www.ehr.nsf.gov/kdi/.

National Science Foundation (1988). Call for Proposal: Digital Libraries Phase 2. Main: http://www.nsf.gov/pubs/1998/nsf9863/nsf9863.htm. Medical Facet: http://dli2.nlm.nih.gov/.

Oard, D. W. (1997). The state of the art in text filtering. *User Modeling and User-Adapted Interaction*, *7*, 141–178.

Payne, T. R., Edwards, P., & Green, C. L. (1997). Experience with rule induction and knearest neighbor methods for interface agents that learn. *IEEE Transactions on Knowledge and Data Engineering*, *9*(2), 329–335.

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

Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, *40*(3), 56–58.

Robertson, N. (1997). A personalized web. *Internet World* (April), 32–34.

Rocchio, J. (1971). Relevance feedback information retrieval. In G. Salton, *The SMART retrieval system experiments in automated document processing* (pp. 313–323). Englewood Cliffs, NJ: Prentice-Hall.

Rucker, J., & Polanco, M. J. (1997). Siteseer: Personalized navigation for the web. *Communications of the ACM*, *40*(3), 73–75.

Salton, G. (1989). *Automatic text processing*. Reading, MA: Addison-Wesley.

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

Schütze, H., Hull, D. A., & Pedersen, J. O. (1995). A comparison of classifiers and document representations for the routing problem. Paper presented at the 18th ACM International Conference on Research and Development in Information Retrieval, Seattle, WA.

Singhal, A. (1998). AT&T at TREC-6. Paper presented at the Sixth Text Retrieval Conference, Gaithersburg, MD. http://trec.nist.gov/pubs.html.

Terveen, L., Hill, W., Amento, B., McDonald, D., & Creter, J. (1997). PHOAKS: a system for sharing recommendations. *Communications of the ACM*, *40*(3), 59–62.

Tou, J. T., & Gonzalez, R. C. (1974). *Pattern recognition principles*. Reading, MA: Addison-Wesley.

Weiss, S. M., & Kulikowski, C. A. (1991). *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning and expert systems*. San Francisco, CA: Morgan Kaufmann.

Yan, T. W., & Garcia-Molina, H. (1995). SIFT — A tool for widearea information dissemination. Paper presented at the 1995 USENIX Winter Technical Conference, New Orleans, LA.