Working Paper Series
ISSN 1170-487X

Applications of Machine Learning in Information Retrieval

**by Sally Jo Cunningham,
James Littin and Ian H. Witten**

Working Paper 97/6
February 1997

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

APPLICATIONS OF MACHINE LEARNING IN INFORMATION RETRIEVAL

Sally Jo Cunningham, James Littin and Ian H. Witten
Department of Computer Science
University of Waikato
Hamilton, New Zealand
email: {jlittin, sallyjo, ihw} @cs.waikato.ac.nz

1 Introduction

Information retrieval systems provide access to collections of thousands, or millions, of documents, from which, by providing an appropriate description, users can recover any one. Typically, users iteratively refine the descriptions they provide to satisfy their needs, and retrieval systems can utilize user feedback on selected documents to indicate the accuracy of the description at any stage. The style of description required from the user, and the way it is employed to search the document database, are consequences of the indexing method used for the collection. The index may take different forms, from storing keywords with links to individual documents, to clustering documents under related topics.

Much of the work in information retrieval can be automated. Processes such as document indexing and query refinement are usually accomplished by computer, while document classification and index term selection are more often performed manually. However, manual development and maintenance of document databases is time-consuming, tedious, and error-prone. Algorithms that "mine" documents for indexing information, and model user interests to help them formulate queries, reduce the workload and can ensure more consistent behavior. Such algorithms are based in machine learning, a dynamic, burgeoning area of computer science which is finding application in domains ranging from "expert systems," where learning algorithms supplement—or even supplant—domain experts for generating rules and explanations (Langley and Simon, 1995), to "intelligent agents," which learn to play particular, highly-specialized, support roles for individual people and are seen by some to herald a new renaissance of artificial intelligence in information technology (Hendler, 1997).

Information retrieval has many characteristics that are suitable for machine learning. Key information retrieval processes are classification tasks that are well suited to machine learning—in many cases, tasks that until recently had to be accomplished manually, if at all. Learning algorithms use examples, attributes and values, which information retrieval systems can supply in abundance. Most such systems have thousands of documents (examples), and the plethora of natural language features that can be extracted from documents provide a wealth of attributes. Thus there is much data available: indeed, it will become apparent that there are *too many* features and examples in most systems for machine learning schemes to process satisfactorily.

The remainder of this section summarizes the field of information retrieval, introducing the different phases of an interaction with a retrieval system and identifying the issues relating to each phase, and the field of machine learning, outlining the major paradigms and showing how they can be applied to information retrieval. The next section introduces the basic ideas that underpin applications of machine learning to information retrieval: different ways of representing text, different ways of extracting features from text, and the problems posed by multiple collections. Section 3 describes applications of machine learning to text categorization—a process that is expensive and time consuming when performed manually. Section 4 considers how machine learning can be applied to the query formulation process: the automatic creation of user models that provide a context within which to interpret users' queries, and the use of relevance feedback to support the query refinement process. Section 5 examines methods of document filtering, where the user specifies a query that is to be applied to an ever-changing set of documents. The final section summarizes the area and presents some conclusions.

1.1 **Information retrieval**

Information storage and retrieval systems make large volumes of text accessible to people with information needs (van Rijsbergen, 1979; Salton and McGill, 1983). A person approaches such a system with some idea of what they want to find out, and the goal of the system is to fulfill that need. The person (hereafter referred to as the *user*) provides an outline of their requirements—perhaps a list of keywords relating to the topic in question, or even an example document. The system searches its database for documents that are related to the user's query, and presents those that are most relevant. General texts relating to information retrieval and computer implementation of retrieval systems are Salton (1988), Frakes (1992a), Baeza-Yates (1992), and Witten *et al*. (1994).

The information retrieval process can be divided into four distinct phases: *indexing*, *querying*, *comparison*, and *feedback* (Lewis, 1991). The first refers to the way documents are managed in the collection. To make searching more efficient, a retrieval system stores documents in an abstract representation that can efficiently isolate those documents likely to relate to the users' needs. Many fielded systems employ a Boolean vector representation for documents. A set of keywords is stored, along with links to the documents in which each word appears. This structure for storing indexing information is called an "inverted file." Standard Boolean operators (AND, OR, NOT) are used to combine keywords into a query. Suppose a user wishes to retrieve documents about motor racing, but is not interested in trucks: a query like *((motor AND racing) OR motorsport) AND NOT (truck)* might be appropriate. The system looks up the words *motor*, *racing*, *motorsport*, and *truck* in the inverted file, and retrieves all documents that are indicated by the first three words and not also indicated by the fourth.

An extension of the Boolean model utilizes information about the distribution of words in each document, and throughout the document database as a whole. Using this information, a retrieval system is able to rank the documents that match and present the most relevant ones to the user first—a major advantage over the standard Boolean approach which does not order retrieved documents in any significant way. From a list of documents sorted by predicted relevance, a user can select several from the top as being most pertinent. In a system using the Boolean model, users would have to search through all the documents retrieved, determining each one's importance manually.

The form and descriptive power of queries reflect the method of indexing that the system employs. The response to a query is constructed using the indices and the operations provided by the underlying conceptual model. The user has to know something about this model in order to structure queries correctly and efficaciously. There is a trade-off between the indexing power of the system's model and the complexity of the queries required to search it. Experienced users of a system with limited scope, such as a database of technical reports, may be comfortable using a complex query interface with additional operators such as proximity. They will know the structure of the index and have a feeling for the specific information that characterizes the documents they seek. In situations where users vary in ability and experience, such as a public library, queries usually take the form of a single key word or phrase indicating the user's interest area. However, users may not know what indexing terms are used to describe the documents they want. In a library environment, the keywords describing the contents of a book are chosen by the people who maintain the library's catalogue. To locate a particular book, the keywords in a query must be identical to those in the book’s catalogue entry. This style of querying requires imagination and persistence on the part of the user, who must be prepared to interact with the system for some time.

Systems that employ simple query methods usually provide other facilities to aid users in their search. For example, a thesaurus can match words in a query with those in the index that have similar meaning. The most effective tool exploited by information retrieval systems is *user feedback*. Instead of having users re-engineer their queries in an *ad hoc* fashion, a system that measures their interest in retrieved documents can adjust the queries for them. By weighting terms in the query according to their performance in selecting appropriate documents, the system can refine a query to reflect the user’s needs more precisely. The user is required only to indicate the relevance of each document presented.

Finally, an information retrieval system’s conceptual model defines how documents in its database are compared. In a system using an inverted file of keywords, two documents may be considered “similar” if they if they have a certain number of keywords in common. A system that uses a term frequency vector model would consider two documents to be similar when their term vectors lie close together in the vector space.

1.2 **Machine learning**

One aim of machine learning is to devise algorithms that can supplement, or supplant, domain experts in knowledge engineering situations. Using learning algorithms to automate information retrieval processes such as document classification and user modeling can alleviate the workload of information workers and reduce inconsistency introduced by human error. General texts on machine learning paradigms and techniques include Carbonell (1990), Piatetsky-Shapiro (1991), and Langley (1996).

Langley and Simon (1995) identify five major paradigms in machine learning research, four of which have been applied in information retrieval studies. The paradigms are rule induction, instance-based learning, neural networks, genetic algorithms, and analytic learning. The first four learn from information with very simple structure—examples of the concept being learned, often described by lists of symbolic or numeric attributes. Heuristics are applied to generate structures that represent relationships implicit in the data. These four paradigms are used in so-called “intelligent” information retrieval systems where the examples (documents) are described by simple features such as word-frequency measures.

The fifth paradigm, analytic learning, is typically employed to learn proofs or explanations for example situations using background knowledge. By compiling groups of explanations a system may be able to reduce the amount of search required to solve similar problems in the future. The background knowledge and complex structure necessary to store explanations make analytic learning systems infeasible for large-scale information retrieval. These algorithms learn much from rigorous expert explanations and require few examples—in some ways the antithesis of the usual information retrieval scenario.

Genetic algorithms are the least frequently applied of the other four paradigms. As the name suggests, they mimic the behavior of biological genetic systems. Examples are represented as a string of values similar to genes in a chromosome. A population of examples is stored, and at each iteration operators such as *mutation* and *crossover* are applied. Mutation changes some of the values in an example randomly, whereas crossover combines different values from pairs of examples into a new instance. The population of examples is prevented from growing indefinitely by retaining only the "strongest" examples as determined by some fitness measure. Evolution is terminated when the user is satisfied with the strength of the surviving examples. In information retrieval, the values in each example might represent the presence or absence of words in documents—a vector of binary terms. The evolutionary process is halted when an example emerges that is representative of the documents being classified.

Why genetic algorithms have been ignored by information retrieval researchers is unclear. The random nature of the genetic operators, and the resulting non-deterministic behavior of the algorithms, may be a reason. Also, it can be difficult to set control parameters, such as the probabilities of crossover and mutation, to ensure good performance.

Decision tree and rule induction schemes are the most-studied and best-developed machine learning techniques. They generate an explicit description of the concept represented by the input data. As with all learning algorithms, the accuracy and appropriateness of the concept description reflect the quality of the data supplied. The algorithms have only the information present in the input to learn from, and will pick the most potent regularities to create the concept description. If strong patterns appear by chance, or the data is irrelevant to the classification task, the concept description will be inadequate. Algorithms use different techniques to determine which patterns in the data are appropriate for incorporating in the concept description, but they can be generally classed as either *covering* or *divide-and-conquer* algorithms.

A covering algorithm creates a set of rules for each class in a concept. Each rule covers many examples of the class in question (*positive* examples), yet selects few examples of other classes (*negative* examples). In the simplest case, terms are added to a rule until it covers only positive examples, which are removed from the dataset, and the process is repeated until every positive example is covered by at least one of the rules. The “best” term to add in each case is determined by a heuristic measure during a search of the space of possible terms.

Divide-and-conquer algorithms recursively split the dataset until the remaining subsets contain examples of a single class. The concept description they create is often expressed as a decision tree, with the ultimate subsets representing leaf nodes. This approach is fundamentally different from the covering method. Although the goal of both approaches is to produce a concise, accurate concept description, typical divide-and-conquer algorithms try to achieve accuracy by creating the smallest possible concept description. Covering algorithms attack the problem from the other direction—they try to produce a small concept description by using the most accurate terms at each step. There is no evidence to indicate that either approach is inherently superior to the other. However, a divide-and-conquer algorithm, C4.5 (Quinlan, 1993), has become the benchmark for inductive machine learning.

Instance-based learning algorithms do not create an explicit concept description that can be used to classify new examples. Instead, training examples are stored, and new examples are classified by comparing them with these. The representation of stored examples and the mechanisms for comparison differ between algorithms. The simplest is the *nearest-neighbor* technique, where training examples are stored verbatim in an *n*-dimensional space, *n* being the number of attributes describing the concept. Examples are compared using Euclidean distance, all attributes being given equal importance. The nearest-neighbor approach is most applicable to numeric or ordered data where the distance between two values has some meaning. With symbolic data the distance between any two different values is often deemed to be 1. Examples compared by a symbolic attribute will be maximally different if the values of that attribute are different.

Instance-based methods are, computationally, among the simplest learning schemes, and variations are often considered as models of human learning. Many theories from cognitive psychology are implemented as instance-based learners for evaluation purposes. Extensions to the standard nearest-neighbor algorithms include generalization of stored examples (Salzberg, 1990) and alternative similarity metrics (Cleary and Trigg, 1995).

Neural networks also lack an explicit concept description. They represent knowledge as weighted links between nodes in a multilayer network, with activation spreading from input to output nodes during the classification process. Learning involves altering the weights on the links until the output nodes give the correct response to the inputs. As the name suggests, neural networks are broadly intended to model physical aspects of the human brain, and in doing so, to model natural learning processes also. However, typical neural networks are minuscule in comparison to the brain, and the concepts they learn are correspondingly simple.

All these approaches to machine learning have been successfully applied to real-world problems—the choice of method in each case seems to be based largely on the experience and preference of the researchers involved. This success is also apparent in information retrieval applications, the different paradigms being utilized in similar proportions to other domains.

2 Machine Learning for Information Retrieval

Information retrieval has been automated for decades (Frakes, 1992a), but only since the late 1980’s has machine learning been applied to this area. The information retrieval process can be divided into four distinct phases: indexing, query formulation, comparison, and feedback (Lewis, 1991), all of which provide opportunities for learning. Researchers tend to focus on one of these subprocesses when trying to improve the performance of their retrieval systems. For example, clustering techniques (Martin, 1995) build links between related documents so that indexing becomes more effective, and user modeling techniques attempt to extract as much information as possible from user queries (Bhatia *et al.*, 1995; Krulwich, 1995a).

The unstructuredness and diversity of natural language text presents interesting challenges to the application of machine learning algorithms (Seshardi *et al.*, 1995). The sheer number of features that could be used for classification overwhelms most learning schemes, and selecting an auspicious subset of features is a difficult task. Diversity also causes problems when several collections are searched to satisfy a single query, for one cannot directly compare ranking scores across collections because each collection’s feature set, and the distribution of those features, is unique.

This section explores the basic ideas that underpin the application of machine learning to information retrieval. The importance of the language model used to represent text is paramount, and several different representations are introduced. Following this, methods for extracting valuable features from the text are presented. Then the problems of retrieval in multiple-collection systems are discussed, and two learning algorithms designed to overcome them are presented.

2.1 Models of text and text representation

For a retrieval system to categorize documents effectively, it must extract enough information from them to discriminate examples of each category. The system must create a model of the documents’ text that somehow captures the essence of what it is that they are about. Natural language has myriad properties that can be used to create such a model, but most are intangible, and without a reasonably solid working knowledge of the language in question a processor (human or computer) cannot take advantage of them. Moreover, speed of processing is an important factor in large-scale information retrieval, so in reality only a small set of easily extracted features can be used by any practical system.

Term vector models

The most popular text representations are based on the “bag of words” or *term vector* model, where a document is represented by a set of *terms* (Robertson and Sparck Jones, 1994; Lewis, 1991; Witten *et al*., 1994). Documents that contain a given set of terms each correspond to a certain point in *n*-dimensional space, where *n* is the number of terms. Terms are usually words, and a term vector is either a Boolean vector representing the set of words that appear in the document, or a numeric vector whose values are derived from the number of occurrences of each term in a document—the latter being based on the commonsense understanding that the more often a term is mentioned, the more likely it is to be central to the subject matter. All other information implicit in the text, such as the meaning and order of words, is lost once the document’s vector representation has been computed. However, the simplicity of this model makes it one of the most popular in the field of information retrieval.

If the term vector is Boolean, queries generally consist of a Boolean combination of terms and any document satisfying that expression is regarded as satisfying the query. If the term vector is numeric, queries are usually represented by the set of terms that they contain, possibly taking their frequencies into account just as is done for documents. In this case, the query is regarded as a Boolean OR query for documents containing any of the specified terms, but it is generally expected that the documents returned will be sorted into order reflecting the number of terms they contain and the frequencies with which they occur—this is called a “ranked” query. Many retrieval models implement a combination of Boolean and ranked query, where documents returned by the Boolean query are subjected to a further ranking process.

The term vector model disregards word order, and different documents can map onto the same vector simply because the same words appear in each—an effect that becomes more pronounced when the dimensionality of the vector space is reduced. Limiting the number of terms is sometimes necessary to make classification of large collections practical. In these cases words may be excluded because they convey little discriminatory meaning (stopwords), occur too infrequently, or are extensions of other words (stemming).

Most information retrieval systems that use symbolic learning algorithms are based on the term vector model of text, because it provides a finite set of attributes (Apte *et al.*, 1994a; Apte *et al.*, 1994b; Cohen, 1995a). The attributes have very simple structure—either a binary value indicating term appearance (for example, Crawford *et al.*, 1991), or a term frequency measure (Apte *et al.*, 1994a). A set of term vectors representing the contents of a retrieval system's database is no more than a table of numbers.

Ranking term vectors using tf×idf and the cosine rule

When document vectors reflect the frequencies with which terms appear, documents are considered similar if their term vectors lie close together in vector space. Before determining distance, the dimensions of the space should be normalized in a way that reflects the differing importance of different words. Importance is generally measured simply on the basis of word frequency, rare words being more salient than common ones. Each term is weighted by the number of documents in which it appears, so that a single appearance of *the* counts far less than a single appearance of, say, *Jezebel*. The components of the term vector are not simply term frequencies, but term frequencies divided by the number of documents in which that term appears. This is called *term-frequency times inverse document frequency weighting*, or *tf×idf.*

Euclidean distance in the vector space is not a good metric for assessing the similarity of documents because the length of the vector representing a document is determined by the length of the document itself, and length is normally a factor that one wishes to discount when measuring document similarity. Thus documents are normalized to unit length before considering their distance. Equivalently, one can measure the distance between vectors as the cosine of the angle between them. This is the standard *cosine measure*:

$$cosine(Q, D_d) = \frac{Q \quad D_d}{|Q| \; |D_d|}$$

where Q is the term vector for the query and D_d is the term vector for the document d. This can be calculated using the formula:

$$cosine(Q, D_d) = \frac{1}{W_q W_d} \sum_{t=1}^{n} w_{q,t} \quad w_{d,t}$$

where W_d is the Euclidean length of document d, and $w_{d,t}$ is the weight of term t in document d. The cosine measure calculates the cosine of the angle between two normalized vectors. A value near 1 indicates that two documents are similar, whereas a value close to 0 indicates that they are very dissimilar, the angle between them approaching 90º. The continuous nature of the function means that documents can be ranked in order of similarity to the query vector.

Latent semantic indexing

The cosine-rule approach to ranking suffers from many well-known problems. For example, there are different ways to refer to the same object or concept, which do not necessarily have words in common—the problem of *synonymy*. Conversely, many words mean quite different things in different contexts—the problem of *polysemy*. Finally, the cosine rule assumes that words are independent, whereas many words occur in key phrases so that the appearance of one in conjunction with the other should not be taken as independent evidence of the similarity between query and document.

In an attempt to circumvent these problems, some researchers re-map the term vectors into a different space. One way of doing this is *latent semantic indexing*, which factors the term-document matrix into three matrices using the singular-value decomposition (Deerwester *et al*., 1990). Figure 1a illustrates the factoring process. The "concept matrix" in the center of the right-hand side is a diagonal matrix, that is, the only non-zero entries lie on the major diagonal illustrated by the thick line. Moreover, elements are sorted into decreasing order along this diagonal, so that the most important ones lie to the top left, and the ones that

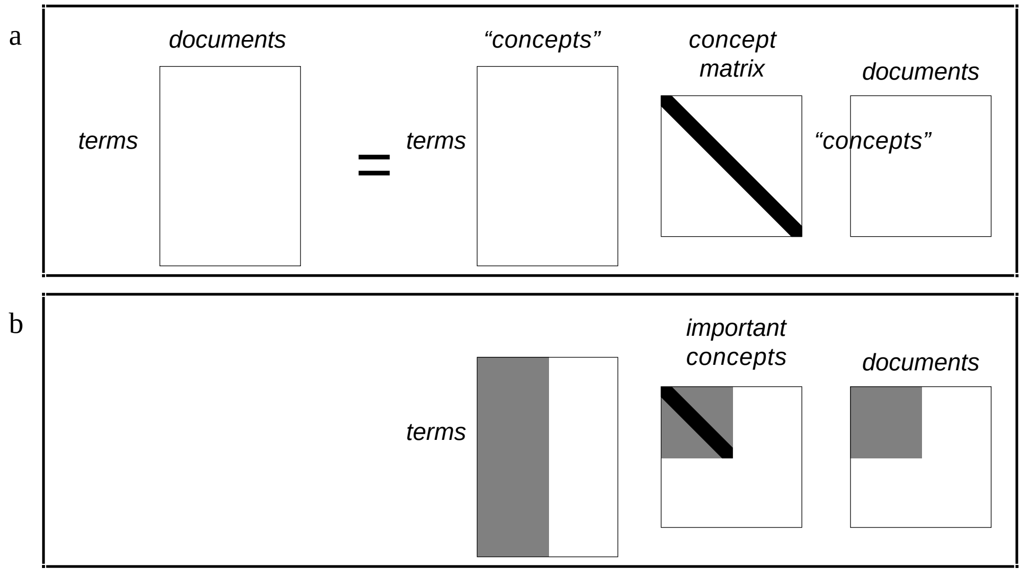

Figure 1 (a) Factoring the document-term matrix
(b) Approximating the concept matrix

occur by chance, corresponding to noise in the document collection, lie to the lower right. Thus the matrices can be altered, as in Figure 1b, by focusing on the important concepts and setting the non-shaded areas to zero. This provides a smaller, more tractable, representation of the original document-term matrix, and the product of these three matrices is now only an approximation to the original matrix.

This representation maps terms, queries and documents into vectors of *k* factor weights, where *k* is the dimensionality of the new concept matrix. An advantage of this approach is that the new space constructed has “semantic” qualities: terms that are used in similar contexts are associated, and a document may be located near relevant terms that do not actually appear in its text. A query is processed by representing its terms as a vector in the space, and then ranking documents by their proximity to the query vector. The representation can be used to compare documents with documents (document clustering) and queries with queries (query clustering). It would seem to form an ideal basis for the application of machine learning, although the representation is fairly new to information retrieval and it has only appeared in a few machine learning applications (Wiener *et al*., 1995; Moulinier, 1996; Hull *et al.*, 1995 and Schutze *et al.*, 1995). Its principal disadvantage is its cost: it is not clear whether the factoring operation can be applied to large-scale document collections.

Sequential models

In the domain of text compression, researchers define an accurate model of a document to be one that allows it to be compressed effectively: the more accurate the model the more effective the compression, and *vice versa*. Many compression methods take account of the order in which terms appear in the text, and use this information to group them into strings of arbitrary length, forming the *alphabet* of the document. An index identifying each string is used in place of the string in the compressed version of the document. To decode the compressed document, a table of strings is consulted, and indices are replaced with the appropriate piece of text.

The ZEROWORD and FIRSTWORD algorithms (Moffat, 1989; Bell *et al.*, 1990) use the sequential grouping of characters in a document as the basis for dividing the text into shorter strings. Sequences of characters delimited by non-letters are used in the alphabet—ideally these map to actual words in the text. To compress a document, known words are replaced by their index, and novel words are specified in full along with their index (which is used thereafter). Moffat shows that this type of compression is very effective for types of text files that are readily divisible into words.

ZEROWORD uses a context-free model, whereas FIRSTWORD uses the presence of one word to predict the next. The PPM algorithm (Bell *et al.*, 1990) also uses context information. PPM compresses at the character level, and uses some finite number of preceding characters to create the context for the next. Terms (characters) known in the context of the previous term are in the previous term's *first-order* context; terms known in the context of the previous-but-one term's context are in that term's *second-order* context, and so on. This process can continue indefinitely and is limited by time and space constraints. However, the more contextual information available, the better prediction will be.

Compression algorithms can be used to categorize documents in a very simple manner that invokes the minimum description length principle (Rissanen, 1985). A sample of text from each category is used to train the compression algorithm. The document to be classified is added to each training sample in turn and the sample that compresses the document into the fewest bits is judged to have the same category as the document. The relative degrees of compression achieved by each sample can also be used to rank the categories. Intuitively, one would expect a better compression algorithm to achieve greater classification accuracy, because the algorithm is able to use stronger patterns in the sample text to compress the unlabeled document (Lang, 1994).

Text compression methods capitalize on the redundancy inherent in word ordering. Most applications of machine learning to information retrieval other than ones using compression algorithms directly discard the ordering of the words in the text. One exception is Cohen (1995a) who, using the relational learning algorithm FOIL (Quinlan, 1990), was able to generate rule sets that utilized relationships between words such as *near* (1, 2, or 3 words apart), *after*, and *successor*. Word occurrence relations, analogous to the term appearance values in the vector model, were also used in the rules. For example the relation *learning*(d_3,2) would indicate that the word "learning" appeared in document d_3 as the second word. The non-word relations, such as *near*, provide background knowledge on word positions. For example, the definition *successor*(4,5) indicates that the fifth word of a document immediately follows the fourth.

Spreading-activation model

Associative retrieval takes advantage of the connectionist model to retrieve relevant documents that may not have many (or any!) terms in common with the user's query (Crouch *et al.*, 1994). In this scheme, activation spreads from the user's query vector, via a term layer containing nodes representing all terms contained in all document descriptions, to a document layer. At this point, the activation level of a document node can be construed as indicating the relevance of that document to the query. Activation then spreads backward to the term layer, reinforcing query terms and adding activation to new terms in relevant documents, and then forwards again to the document layer, where additional documents may be activated by the new terms. At this point the cycle is halted and the documents are ranked for retrieval according to their activation level. Additional cycles through the network may randomize the document ranking (Wilkinson and Hingston, 1991, 1992). Tests on actual text collections indicate that associative retrieval can increase precision, but may decrease retrieval performance for queries that have few or no relevant documents in the collection. In this latter case, the query terms will be augmented by terms from irrelevant documents, and these new terms will in turn activate still more irrelevant documents.

2.2 **Feature extraction**

The text model adopted by an information retrieval system defines the *features,* or types of information, to extract from the text. It is these features that the learning component uses to find regularities from which it can create classifiers. The better a system is at selecting auspicious features, the better its chance of generating an accurate concept description. It is commonly accepted that feature extraction is the most critical stage of the learning process in almost any domain (Holte, 1993; Lewis and Ringuette, 1994; Krulwich, 1995a).

Problems in feature extraction arise for several reasons (Martin, 1995; Lewis and Croft, 1990). First, the large number of synonyms in English—or any other natural language—allow a similar idea to be expressed by different words in different documents. With any of the text models described in the previous section except those based on latent semantic indexing or spreading activation, related documents would have to use enough of the same words for the similarity to be recognized. Second, many words will appear in both documents simply because they are in the same language; such words are distributed throughout most documents and have no discriminatory power. Third, due to polysemy the same words can appear in different documents with quite different meanings.

Two documents may even be considered similar because they both omit a number of important words. This is necessarily the case in two-class categorization situations where the documents in the negative class are similar only in that they are not related to documents in the positive class. In multi-class clustering situations, however, it is necessary to further differentiate the documents, and then the significance of missing words is not so plain. Usually this issue is ignored, but Martin’s (1995) text clustering system deals with it in the following way. When comparing documents, the absence of a keyword from both term vectors indicates that the documents *could* both contain the word, but the information is not available. If, however, one vector contains the word but the other does not, the documents are assumed to differ with respect to that word.

Weighting terms

One common technique for indicating the degree to which a document is “about” a term is to weight the term. As noted above, this is generally done by dividing the *term frequency*, the number of times that the term appears in the document, by the *document frequency*, the number of documents in which it appears—in other words, *tf*×*idf*. Fuhr and Buckley (1991) adopt a probabilistic approach to assigning weights to index terms. They view the result of processing a set of queries as a space of query-document pairs with attached relevance judgments (relevant or not relevant). Given an experimental sampling of this space, the probability of relevance of a document given each index term can be estimated. This estimate is then used predictively in retrieval to estimate a document’s relevance to a new query. The probability estimates can be revised in the light of additional relevance data. An interesting feature of this approach is that the system can refine its retrieval capability incrementally, based upon usage.

Another consideration in determining a term’s weight may be the position of that term in the document. For example, words appearing in a title or abstract are more likely to express a concept central to the document than terms buried in a middle section. Documents encoded in a mark-up language such as HTML or SGML provide a particularly rich source of structural information that can be used to fine-tune a weighting, and the link structure of Web documents can be similarly useful in indicating significant terms—for example, those that appear in the anchor phrase for a link (Boyan *et al*., 1996).

Using a thesaurus

Another standard technique is to employ a thesaurus to match synonyms (Martin, 1995). The thesaurus can be constructed automatically, most commonly by detecting related terms through their co-incidence in document pairs (Salton and McGill, 1983). Wong *et al.* (1993) use a neural network to compute term associations and their weightings. Grefenstette (1994) uses sophisticated, but practical and robust, techniques of natural language processing to derive appropriate word context from free text. Furnas (1985) presents an adaptive indexing scheme that learns semantic connections between individual terms by monitoring user interactions. The user is assumed to issue a series of queries in a single session that are related to the same information need. Once a relevant document is located, the system updates the weights for terms in earlier, unsuccessful queries, on the basis that these terms are useful synonyms.

Fully automated thesaurus construction techniques produce term relationship networks that can be used to augment queries automatically, but the networks themselves generally cannot be interpreted by people. Guntzer *et al*. (1989) use a semi-automated technique for constructing a semantically meaningful thesaurus. User interactions are monitored, and a knowledge acquisition subsystem attempts to infer concept relationships between query terms by matching query structures to previously-defined production rules. In most cases these relationships are somewhat uncertain and must be confirmed by the user before being committed to the thesaurus.

Identifying phrases

Other approaches to extracting the most distinctive parts of documents transcend the use of single words. Krulwich's (1995a) system, operating in a bulletin-board environment, identifies semantically significant *phrases* in documents. Phrases can consist of words in the keyword, name, title, and subject fields, and certain patterns in the text fields of documents. For example, a single word that is fully capitalized, or a short phrase in a different format—such as bullets, diagram labels, or section headings—is usually regarded as significant. Each document is rechecked for the complete set of phrases, and the system learns a subset of phrases that best describes the category. The induction algorithm used is simple, because the purpose of the study was to develop more sophisticated feature extraction heuristics. Comparison of a number of different learning algorithms, from neural networks to standard decision-tree learners, underscored the point that feature selection is the most critical part of the learning process.

Conrad and Utt (1994) use a similar technique to extract names of companies and people from *Wall Street Journal* articles. Company names are recognizable when they include words such as "Corporation" and "Inc," and personal names are capitalized and may be preceded by a title such as "Ms." Tables of common personal names, titles, and so on, are stored by the system and used to identify important objects in the articles. The system is being extended to recognize synonyms and abbreviations such as "Digital Equipment Corporation/DEC." Although these techniques are very simple, the researchers show figures of 89% for precision and 79% for recall on a test database of 139 articles.

These systems use formatting information to determine the extent of phrases, as well as their relative importance. If a system has access only to the plain text of documents, the problem arises of determining the appropriate granularity with which to divide the text. The standard grain size is the word. Words are relatively easy to distill from text, and are obvious distinct units of meaning—although even here the question of word boundaries, and stemming or affix stripping, is a non-trivial issue (Frakes, 1992b). But the difficulties inherent in language outlined above have led researchers to examine larger sections of text such as phrases, sentences, paragraphs, and groups of paragraphs.

A number of studies have shown that simple template- and parser-based methods, and statistical phrase-formation algorithms, are able to find important phrases. In a study of phrase clustering, Lewis and Croft (1990) used a simple syntactic method. They defined a phrase to be any pair of non-function words that headed a syntactic structure connected by a grammatical relation. For example, a verb and the first noun of the following noun phrase form a phrase. Extracted phrases are clustered into small groups containing a seed, along with the phrases most similar to it. It was shown that the use of phrases instead of single words slightly improved both recall and precision.

More recently, Cohen and Singer (1996a) investigated the use of “sparse phrases” consisting of a small subset of nearby, though not necessarily consecutive, terms in a fixed order. The sparse phrases present in any of the training documents are extracted and used as simple classifiers or “experts,” each one assigning a single class if its associated sparse phrase appears in a document. Initial weights were assigned to these experts and adjusted during a training phase by comparing the experts’ predictions to the “true” classification value for each training document. When categorizing an unknown document, the predictions of these experts are combined by using the weights. In general, most of the experts may be “sleeping,” in the sense that the phrases they represent do not occur in the document and therefore cannot participate in prediction; hence this method is dubbed the *sleeping experts* algorithm. The key, of course, is to decide on which sparse phrases to use, and Cohen and Singer (1996a) chose all possible phrases of a certain length (four words)—an enormous set. They claim that the resources required to do this are “quite modest” because an overwhelming majority (over 99%) of phrases appear just once in the corpus and information is only retained on experts that appear at least twice.

Detecting subtopics

Hearst’s (1994) TEXTTILING system uses linguistic analysis to determine where subtopic changes occur in lengthy texts. These boundaries, which generally occur at paragraph breaks, are used to divide the text into subsections. Each subsection has its own vocabulary of words relevant to its topic, and the distribution of these words is skewed towards related sections. A global word-based approach to comparing two documents might miss relationships between individual sections because the distribution of the relevant words across the whole document is relatively sparse. Salton *et al.* (1994) also match sections of full text to determine document similarity.

2.3 Collection fusion

Information retrieval systems generally have a single text database that is consistent in terms of indexing and content. However, some systems are required to search a number of databases to satisfy a user's query (Towell *et al.*, 1995). For example, separate collections might be used for *football* and *motor racing* documents, and a query about *world champions* would find relevant documents in both collections. Problems arise because the individual collections inevitably have different term distributions. If the same document were in both collections it would receive different scores for the same query, and thus scores from different collections cannot be directly compared. The effect is exacerbated when collections have their own servers, and use different ranking and retrieval strategies.

A typical query to a multiple-collection system asks for a certain number of documents, *N*, selected from *C* collections, and ranked in order of relevance. The problem of merging the documents in a valid order is called *collection fusion* (Voorhees *et al*., 1995). Simple solutions such as selecting the top *N/C* documents from each collection, or assuming that relevance scores are compatible across collections and selecting the *N* highest-scoring documents overall, perform poorly. Towell *et al.* (1995) introduce a collection fusion scheme that uses the document distributions of training queries to learn the appropriate number of relevant documents from each collection to retrieve for a new query. The algorithm models the distribution of a query by averaging the distributions of its *k* nearest neighbors in the training data. The number of documents to retrieve from each collection is calculated using a maximization procedure (not described in the article), and the order of presentation is determined by a random process biased toward collections with the greatest number of articles remaining.

A second learning scheme described by Towell *et al.* (1995) clusters training queries that retrieve several common documents from each collection. The average of the query vectors in each cluster is the system's representation of the topic covered by those queries. For each collection, each cluster is given a weighting which is proportional to the number of relevant documents retrieved from that collection. After training, new queries are matched with the most similar cluster and that cluster's weighting model is used to determine the proportion of documents to retrieve from each collection.

In terms of precision given a fixed recall figure, both these learning algorithms perform significantly better than simple non-adaptive schemes such as that described above. Of the two, the first comes closest to the results obtained by combining the collections and using a single server, achieving precision figures within 10% for recall of 10 to 200 documents.

3 Machine Learning for Text Categorization

Text categorization is the process of classifying documents into one of a number of categories. Document classification is required at several stages of the information retrieval process. A user's queries must be classified to determine which documents in the collection are relevant. A query may be treated as a document in its own right, or as a description of the desired concept. The category represented by the query may not be represented explicitly in the database—it is the system's indexing mechanism that determines how documents that fit the category description are located.

Other applications of text categorization occur in library cataloguing systems and newswire stories. Predicting the category of new examples by learning a concept description from a set of training examples is a standard machine learning task, although in document classification the high dimensionality of the data demands special consideration.

Automating the labeling process has two advantages. The volume of work involved in text classification tasks is enormous, and manual classification is time consuming and expensive. Once an automated system has achieved a satisfactory level of accuracy, it can perform the task in a fraction of the time. In one particular study, described in Section 3.2, a technique based on machine learning for automatic labeling of documents yielded a 500-fold reduction in the number of documents requiring manual classification.

The second benefit is consistency. Human experts often disagree on classification of documents, and even a single person may classify documents inconsistently (Apte *et al.*, 1994a). Differences in background knowledge and ability, as well as the genuinely fuzzy semantic boundaries of concepts in a classification scheme, are responsible for inconsistent human classification.

Human fallibility also creates problems for automatic classification systems. Human performance is regarded as ideal in classification tasks, and systems strive to attain this level. But classification accuracy of 100% is invariably unattainable (Apte *et al.*, 1994a). All documents used to train a classification system must first be labeled by a human expert. Even with 100% accuracy on this training data, a system will have incorporated the expert's errors and inconsistencies into its category descriptions. Had another person classified the documents, the concepts would have been described differently.

There is an extensive literature on unsupervised document clustering. Generally, nearest-neighbor techniques are used to categorize documents into naturally-occurring groups which may or may not represent semantically meaningful categories. Many algorithms force a document to belong to only one cluster, which is unduly Procrustean given that in practice most pieces of text can be usefully described by several different classification headings. Unsupervised document clustering lies outside the scope of this article; the reader is directed to Willett (1988) for a survey of document clustering methods.

Many machine learning techniques have been applied to text categorization problems. The next section describes algorithms that have been used and the concept descriptions they create. Although—perhaps because absolute measures of performance are lacking—no particularly striking successes have been claimed, none of the systems discussed are considered *ineffective* by their developers.

3.1 **Algorithms, concept descriptions and data structures**

The use of learning systems in information retrieval typically involves either applying an existing "off the shelf" algorithm in some manner, or developing a new learning algorithm specifically for this problem. The choice of approach depends on the intended use of the system, and on the experience and interests of the developers.

Standard algorithms

Many standard machine learning algorithms have been used in information retrieval systems. In general, simple, robust classification techniques like ID3 (Quinlan, 1986) are used for text classification problems, such as identifying text features that catalyze learning (Krulwich, 1995a; Lehnert *et al.*, 1995; Soderland and Lehnert, 1995). Chen (1995) used ID3 as a control for an experiment with ID5R (Utgoff, 1989), an incremental extension of ID3. Somewhat more elaborate schemes such as ID5R, C4.5 (Quinlan, 1993) and FOIL (Quinlan, 1990) are usually relegated to feasibility experiments (for example, Chen, 1995; Cohen, 1995a; Cunningham and Summers, 1995; Bloedorn *et al*., 1996). Few studies have conducted systematic comparisons of learning schemes over document collections of any significant size. Moulinier (1996) analyzes the relative performance of ID3 (Quinlan, 1986), the production rule learners CHARADE (Ganascia, 1991) and SWAP-1 (Apte *et al*., 1994a), the instance-based scheme IB (Aha, 1992), and neural networks, over a document collection of Reuters news articles. Lewis and Ringuette (1994) compare Bayesian classification with the decision tree learner DT-min10 (Buntine, 1990) on two collections, one of which was the Reuters collection.

One defining feature of a learning algorithm is the form in which it represents the concepts it learns. In information retrieval this concept description is used to classify documents, or to explain the existing labels of documents. Inductive learning schemes, such as ID3, FOIL, and INDUCT (Gaines, 1991), are useful for these tasks because they produce simple, easily-understood rules or decision trees (Lewis and Ringuette, 1994). A supervisor can evaluate and augment these structures for classification purposes, or use them to gain insight into human-categorized databases.

Production rules are the most common form for concept descriptions in document categorization applications. Categorization has been undertaken with propositional rules such as those generated by INDUCT (Cunningham and Summers, 1995) and SWAP-1 (Apte *et al*., 1994a), with rules refined from decision trees (Crawford *et al*., 1991; Cunningham and Summers, 1995; Bloedorn *et al*., 1996); and with relational rules generated by first-order learners like FOIL (Cohen, 1995a). An example of propositional rules generated from 730 Reuters news articles appears in Figure 2. Synthesized from a decision tree created by CART (Breiman *et al*., 1984), these rules test the presence or absence of a particular word in a document in order to predict whether or not articles are about terrorism. Other algorithms use rules involving numerical thresholds on the number of occurrences of words.

Relational terms add an extra dimension to a rule set. Cohen (1995a) defined positional relations that can be used in addition to word-occurrence information, enabling rules to specify relationships *between* words in a document. Figure 3 shows rules intended to classify documents on machine learning. Instead of merely appearing in the same document, the words “decision” and “tree” in the second rule of Figure 3 must also satisfy the *successor* relation—that is, “tree” must directly follow “decision” at some point in the document.

The use of term context in the formation of decision rules has also been investigated in an order-independent manner. The RIPPER rule-generation system (Cohen, 1995b), which was originally designed to deal effectively with situations in which there are a very large number of features, can also accommodate set-valued features (Cohen, 1996a). This allows a term’s

```
if article contains word “bomb”
    then if article contains words “injure” or “kill”
            then terrorism article
            else ~terrorism article
    else if article contains word “kidnapping”
```

Figure 2 Rules representing a decision tree generated by CART

context to be represented as a set of other terms that co-occur in the document, with no indication of term ordering or proximity (Cohen and Singer, 1996a).

When human comprehension of the concept description is not important, other methods are often applied. Versions of the nearest neighbor technique are used when clusters of documents or queries are generated and compared (Towell *et al.*, 1995). Distance metrics are usually very simple, such as the sum of the differences of term counts (Salton *et al.*, 1994). Bayesian probabilistic algorithms also have no readily-comprehensible concept description. Lewis and Ringuette (1994) used Bayes' rule to estimate $P(C_j=1|D)$, the probability that a category C_j should be assigned to a document D given the category's prior probability and the conditional probabilities of particular words occurring in a document given that it belongs in the category. Each category is assigned to its top scoring documents in proportion to the number of times it was assigned on the training data. Lewis and Ringuette compared the precision and recall of their Bayesian algorithm with a decision tree learning algorithm and found little difference between the two. The Bayesian classifier was able to handle ten times as many features as the decision tree learner, although both algorithms peaked in performance between ten and one hundred features.

Hearst (1993) proposed a *case-based reasoning* approach to represent the main topics and subtopics in full-text documents. A subtopic structure outline is an abstract representation of a document, and can be used as an example for case-based reasoning. Document cases are placed in a network, positioned according to which cases they resemble. To specify which features may differ without materially affecting the topic, a term vector is associated with each section of a document. Cases are organized first by their main topic, and then grouped according to which and how many of their section term vectors are similar.

Unsupervised learning schemes have seen relatively little application in document classification. While attractive in that they do not require training data to be pre-classified, the algorithms themselves are generally far more computation-intensive than supervised schemes. The added processing time is particularly significant in a domain such as text classification, in which instances may be described by hundreds or thousands of attributes. Trials of unsupervised schemes include Aone *et al.* (1996), who use the conceptual clustering scheme COBWEB (Fisher, 1987) to induce the natural groupings of close-captioned text associated with video newsfeeds; Liere and Tadepalli (1996), who explore the effectiveness of AutoClass

```
machine_learning_article(S) :– learning(S,P).
machine_learning_article(S) :– decision(S,P), tree(S,Q), successor(P,Q).
```

Figure 3 Relational rules generated by FOIL for document classification

(Cheeseman *et al.*, 1988) in producing a classification model for a portion of the Reuters corpus; and Green and Edwards (1996), who use AutoClass to cluster news items gathered from several sources into “stories,” which are groupings of documents covering similar topics.

Neural networks and genetic algorithms have also been used for classifying documents (Chen, 1995; Towell *et al.*, 1995; Dasigi and Mann, 1996; Wiener *et al*., 1995). As with the more complex inductive schemes, these experiments have usually been performed as controls or feasibility studies. All feasibility experiments in this area show that machine learning techniques can be applied productively to text classification problems when the domain is constrained to a small feature set and a small number of categories. Although these techniques perform a certain amount of feature selection when generating a concept description, it has been shown that it is necessary to prefilter the text to remove ineffectual features.

Specialized algorithms

A number of learning algorithms have been developed strictly for text classification. An early paper uses a probabilistic model to match words appearing in a document to a set of indexing terms (Maron, 1961). A training set of documents is classified manually, and a statistical model is constructed of the correlation between “clue” words (nouns, hyphenated words, and other terms felt to convey subject information) and the classifications associated with documents containing those terms. The statistical model can then be used to classify new documents probabilistically, and the model can be refined by basing it upon very large training sets. Maron, optimistically as it turned out, felt that the problem of automated indexing was essentially solved:

> No real intellectual breakthroughs are required before a machine will be able to index rather well. Just as in the case of machine translation of natural language, the road is gradual but, by and large, straightforward.
>
> (Maron, 1961)

Bhuyan and Raghavan (1991) developed a probabilistic scheme that builds document relationship clusters based on user feedback. To store information about pairs of documents, three graphs are generated from a series of queries. If both documents in a pair are relevant to a query, weight is added to the corresponding edge in the *POS_POS* graph to reflect the similarity of the documents. If the first document is relevant and the second is not, the edge of the *POS_NEG* graph is similarly updated; and conversely for the third graph, *NEG_POS*. A new graph *G* is generated by summing the weights of the *POS_NEG* and *NEG_POS* graphs and subtracting those of the *POS_POS* graph. The graph *G* indicates the similarity between documents, with lower weighted edges denoting greater similarity.

In an extension to the OKAPI system, Goker and McCluskey (1991) define concepts with a set of *concept term structures*. Initially the concept description is bound to the set of all terms that appear in more than one document. Then further highly-weighted terms are added until all relevant documents are covered. This set is called the *active set*; the remaining terms make up the *passive set*. After each user session new terms are merged into each set and existing terms are transferred between sets as their weights change. The active set is limited to 32 terms by moving the terms with lowest strength back into the passive set. After each query, the concept description can be used to refine the ranking between documents that were initially given the same weight. A document's new weight is the sum of the words that appear in both the concept description and the document.

Inference networks rank documents based on the probability that they satisfy the user's information need (Turtle and Croft, 1990, 1991, 1992; Tzeras and Hartmann, 1993; Haines and Croft, 1993; Callan *et al*., 1995). Figure 4 shows the structure of an inference network used for information retrieval. Although such networks were not specifically developed for

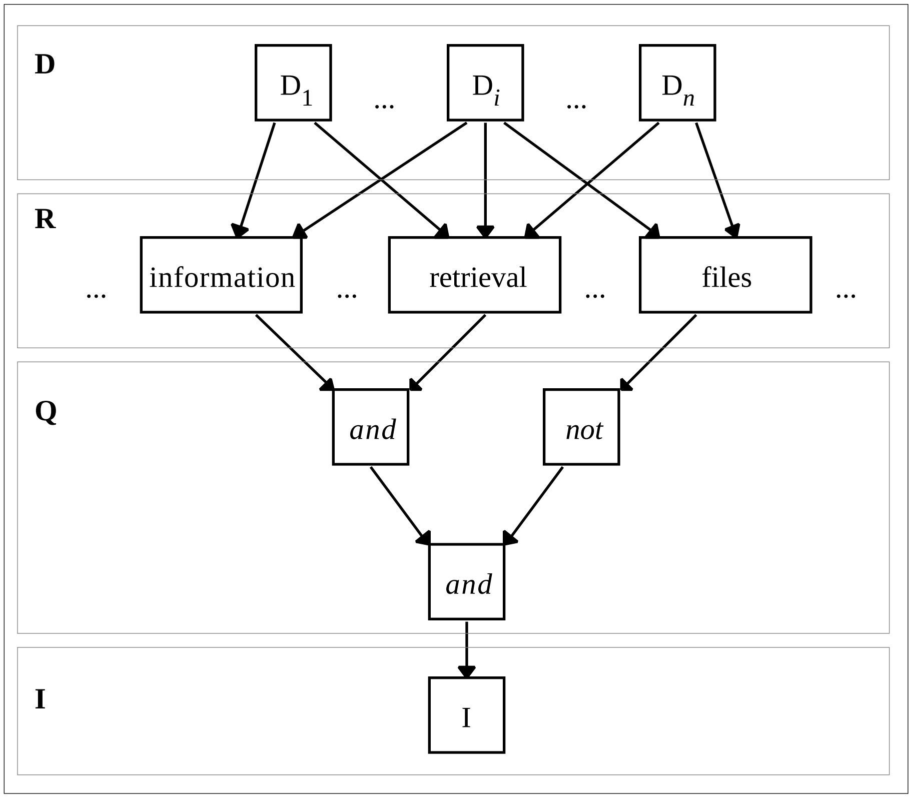

Figure 4 An example inference network created from the query *(information and retrieval) and (not files)*

information retrieval, this structure indicates the information necessary for applying the technique to this domain. The *D* nodes represent documents in the system's database, the *R* nodes describe the contents of the documents, the *Q* nodes represent queries, and the *I* node is the user's information need. The idea is that documents with similar content are linked to the same *R* nodes. The *D* and *R* nodes are constant for a set of documents: they reflect relationships between the documents. The *Q* and *I* nodes are created for each query—in this case *(information and retrieval) and (not files)*. Probabilities filter down the network from each of the document nodes at the top, and the value that arrives at the *I* node indicates the relevance of each document to the user's information need. The inference network architecture can also be used to address the collection fusion problem (Callan *et al*., 1995) by constructing a meta-network in which leaves represent entire collections, rather than individual documents. The user's query is processed against the network to retrieve a ranked list of collections, and the most highly ranked collections are then searched.

The "Expert Network" (Yang, 1994, 1996) is trained using documents that have been assigned index terms by human experts according to a formal classification scheme. In the example case, sample documents are drawn from the MEDLINE medical bibliographic system and are classified according to the MeSH scheme. Input nodes in the network are terms occurring in the training documents. These terms are linked to the documents by a *tf*×*idf* weighting, and the documents are linked to subject categories by the human-assigned classifications, with weightings based on the conditional probabilities of categories given a document. To classify a new document, the similarity is measured between it and the training set to determine a number of matching training documents, along with associated similarity scores. Then, the new document is assigned the categories associated with the matched documents according to the categories' conditional probabilities and the documents' similarity scores. A version of the Expert Network sees daily use at the Mayo Clinic as a search engine for computer-aided coding of patient records (Yang, 1996).

The learning system in NEWSWEEDER (Lang, 1995) is based on the *minimum description length* principle. The minimum description length trades model complexity off against error rate, and is used in this domain to determine how to weight each term, and how to decide which terms have so little discriminatory power that they can be omitted. NEWSWEEDER's term distribution model assumes that the probabilities of occurrence of any two terms in a document are independent. However, the probabilities are allowed to depend on the length of the document. Distributions of each term are computed for each category, and across all categories. To create a model for each category a choice is made between the category-specific distribution and the overall distribution of each term. If the difference between the probabilities for a term is larger than a threshold, the category-specific distribution is used for that term, otherwise the overall distribution is used. The threshold value represents the cost of including parameters that specify the category. Finally, linear regression is used to produce a continuous prediction of the user's ratings for new documents.

The *multiple cause mixture model* (Sahami *et al*., 1996) is novel in that it supports unsupervised as well as supervised learning of classification categories. In unsupervised mode it does not require pre-classified training examples, and induces a set of "naturally occurring" categories for the document set. Unlike many earlier clustering schemes, this method can classify documents into multiple categories. Unfortunately, the unsupervised mode is extremely expensive computationally.

The *category discrimination method* algorithm for text categorization is based on the concept of *cue validity* in cognitive psychology (Goldberg, 1995). Features are selected for inclusion in the categorization model based on how well each feature distinguishes documents belonging to a given category from documents assigned to similar categories. The features whose cue validity exceeds a set threshold are then considered as candidates for a rule-based text categorizer; the categorizer is constructed by conducting a multi-stage search to determine the optimal parameter settings and to eliminate unnecessary features.

Existing document subject classifications can be adapted for better retrieval performance by *genetic algorithms* (Gordon, 1989). The genetic algorithm modifies document descriptions so as to improve their ability to match relevant queries and to fail to match non-relevant queries. As a result of this process, the relative density of document clusters increases so that descriptively similar documents are grouped together. These groups can then be identified by traditional clustering techniques.

3.2 Uncertainty feedback

Most retrieval systems store vast amounts of text and it is impractical to use it all to train a classification system. Any sample selected as a training set should include documents that are good discriminators for the categories represented in the collection. Random sampling is usually unsuitable, particularly when the distribution of categories is skewed and the sample size is relatively small. Relevance feedback (discussed in Section 4.3) performs a kind of sampling by selecting documents that are representative of a given category, but as the learning system improves many of these documents provide no new information for the system to learn from.

Lewis and Gale (1994) present an algorithm for sequentially selecting a subset of training documents based on *uncertainty feedback*, which identifies documents for which the system is most unsure of the appropriate category. By labeling these documents the supervisor creates a training set of example documents that are more disparate than those selected by random sampling. The system starts with a small set of examples, selected either randomly or as being particularly representative of some categories, and a classifier is created for these documents. It then proceeds through a *classification—feedback—create a new classifier* loop until a human supervisor judges the current concept description accurate enough. Experiments show that this point can be reached with as much as a 500-fold decrease in the number of examples required by random sampling.

4 Machine Learning for Query Formulation

The user's input to an information retrieval system is the query. Queries are most often Boolean expressions over document keywords, but they may also be whole or partial documents. Some systems accept natural language queries; they generally translate them into a Boolean expression for retrieval purposes. However, Boolean queries are far from ideal for representing users' information needs (Salton *et al.*, 1994; Hearst, 1994). Inexperienced users often find them difficult to structure effectively, and they provide no facility for indicating the relative importance of individual terms. To overcome these problems many systems allow the query process to be iterative. Using the original query as an initial guess of the user's information need, it is progressively refined until the user is satisfied with the set of documents retrieved.

This section shows how machine learning techniques can be applied to three important aspects of the query formulation process. The first is *user modeling*, an activity that occurs over a series of interactions with the system as it builds a profile of the user's interests. This is most useful when a user's area of interest remains relatively constant in a continually changing information environment, a scenario that we return to in Section 5. The second is *relevance feedback*, a standard technique for query refinement in which the system gains information from the user about the pertinence of selected matching documents. The third is *document filtering*, where a user specifies a query that is to be applied to an changing set of documents—this is appropriate when users' interests remain constant and the document database is continually expanding. First, however, we discuss in broad terms the problems that users experience when formulating queries for an information retrieval system.

4.1 Users find it hard to articulate what they want

People use information retrieval systems when they need information that they believe is present in documents in the system's collection. Because of the large volume of text and the wide range of topics covered, users cannot know exactly how to retrieve all, and only, the documents relevant to their information need—a problem that is exacerbated when they have only a vague idea of what they want. For example, a student researching a topic for a paper will probably begin with just a few keywords that characterize the information they want—they will certainly not have titles and authors of every paper in the collection related to their topic.

The user's unclear idea of what they want and how to get it means that the first query will be very general—most often just a concatenation of a few known keywords. Unfortunately, characteristics of natural language, such as homonymy, synonymy, and ambiguity (Lewis and Ringuette, 1994; Martin, 1995) make it impossible for any retrieval system to present the user with every relevant document given such an imprecise description. The system is unable to predict, from such a tiny sample of pertinent text, the documents important to the user's information needs. Returning all documents related in some way to the initial set of keywords is the simplest response, but in addition to relevant documents this would include many with very weak links to the area of interest, some mentioning it in passing and others using the same query terms for different purposes. The user would have to sift through a large number of documents for ones that appear interesting. Even when the system ranks the documents it retrieves, the query's lack of discriminatory power causes weight blocks (Goker and McCluskey, 1991), where many documents have the same rank.

An information retrieval system should help users by not overwhelming them with copious amounts of text. It should assist users to refine their initial query, so that they are eventually offered a smaller number of documents more illustrative of their information need. The system needs to extract enough information from the user to determine which of the documents related to the initial query are actually important. This is necessarily an iterative process, involving an attempt to converge on a far more explicit description of the user's information requirements.

4.2 Modeling user concepts

User modeling is an activity that occurs over a series of interactions with the system as it builds a profile of each user. It is most often used where a user's requirements remain constant, for in these situations it is the user's *interests* that the system tries to identify.

Consideration of how a retrieval system can model user interests by analyzing the queries they make is deferred to Section 5.1. Here we review how information about users can be solicited explicitly to form user models.

Bhatia *et al.* (1995) introduce a model that describes concepts in the user's vocabulary and can serve to interpret and refine queries in a manner appropriate for the individual user. First, the system "interviews" the user to determine objects, and relationships between them, that are important in the user's universe of discourse. The interviewing process is based on *personal construct theory* (Kelly, 1963). Objects named by the user (called *entities* in personal construct theory) are stored in a table, and presented back to the user in randomly selected triples. The user must come up with a perceived bipolar property of the entities (called a *construct*) that serves to differentiate one of the entities from the other two. The construct becomes a row of the table, and the user is asked to enter a value from a predetermined scale that indicates the relevance of the construct to each entity. The user may spontaneously add new entities and constructs throughout the interview.

The constructs supplied by the user represent their personal *point of view* of the objects around them based on their environment and background (Kelly, 1963). A different person may categorize similar objects with entirely different constructs. For example, one person may not find the construct *black-vs.-white* relevant to the entity *time of day*, whereas a person who regularly uses a train timetable that divides the day into strips of black-on-white and white-on-black text may regard this as a relevant discriminator. *Black-vs.-white* is not a natural property of *time of day*, it is one devised as an arbitrary distinguishing feature of the entity. Constructs provide the basis on which objects are understood, and may not necessarily reflect their actual differential groupings. Therefore the system will develop a different structure for each user's world view.

The user profile is employed by the system to map concepts important to the user to keywords occurring in documents in the database. The user selects a small set of documents with which they are quite familiar, with at least one document containing an example of each concept to be used during querying. The system creates a mapping for each concept to a set of index terms obtained from the appropriate example documents. Documents are represented in term-vector format, with terms weighted using the *tf*×*idf* method. Terms with weights above an empirically determined threshold are selected as representative of the document. The user enters queries using their own vocabulary and concepts, and the system maps these to production rules and keywords to search the remainder of the database.

4.3 Relevance feedback

Because of the general nature of initial queries, which often have only a few terms, and the system's imperfect representation of the content of documents, a user's initial query to an information retrieval system usually returns only a few interesting documents, diluted by many irrelevant ones that match only weakly (Lewis, 1991). The documents may be ranked, but the user is still required to scan through the list to determine where the cutoff point for irrelevant documents lies. Often users will accept a short, incomplete list of documents, or a single reference, and the result of their initial query will suffice. However, when a more complete list of relevant documents is required, the initial query must be modified. Users may do this themselves, but will often be unable to make effective refinements, having stated their requirements to the best of their ability in the first place.

The *relevance feedback* technique expands queries by extracting significant parts of documents retrieved by the user's query and then asking the user to indicate each part's relevance to their requirements (Robertson and Sparck Jones, 1994; Harman, 1992; Lewis, 1991). The user may be presented with whole documents, abstracts, selected passages (Allan, 1995), keywords, or other terms the system deems representative of the results of the initial query. These items are usually ranked, and their number limited to reduce the risk of including worthless terms. The selected terms are then added to the initial query, existing terms reweighted according to their performance in the previous search, and the query processed again. This procedure is repeated until the user is satisfied with the list returned by the system.

Many weighting schemes and similarity measures have been proposed (for a survey see Jones and Furnas, 1987). In practice, it appears that no single ranking function is superior for all users, over all queries. Bartell *et al*. (1994) address the problem of function selection by providing a general similarity measure whose parameters are optimized to a particular user over several queries. Initial tests appear promising. In an small-scale experiment using 51 training queries and 25 test queries, the learned similarity measure performed at least as well as several "classic" ranking functions, and within 1% of an estimated optimal similarity measure.

A learning system can use terms or documents indicated to be relevant as positive examples, and those deemed irrelevant as negative examples (Lewis, 1991). The user's first query becomes the initial concept description, and the system attempts to refine this as the relevance feedback process continues. The small size of the initial query limits the number of features to learn from, inadvertently combating the dimensionality problem. The feature set is expanded as new terms are found to be important to relevant documents, but it will never approach the number of terms present in the documents. The terms of the initial query can also be given more emphasis than positive examples introduced later, so that they always have more weight during the learning process.

The amount of training data available to a learning system via relevance feedback is relatively small, with a bias towards positive examples. Bias occurs because the system is trying to find positive examples of the user's desired concept, and presents the user with items that best match the concept so far developed. Lewis (1991) suggests that bias is probably appropriate in this situation.

Haines and Croft (1993) describe extensions to an inference network model of information retrieval to include relevance feedback techniques. They investigated several variables pertaining to relevance feedback, such as term reweighting, additional terms, and new term weighting methods. In the inference network model, queries are represented as links between the query nodes (*Q* nodes of Figure 4), and the information need node (*I* node). Query term weights are determined by the *weighted sum* form of the link matrix at the *I* node. To incorporate terms determined by relevance feedback, new links are added between the *I* node and the new *Q* nodes created for each new query concept. The link matrix weights are re-estimated using the sample of relevant documents. The weight associated with a query term is used to predict the probability that the information need is satisfied given that a document has that term. Relevance feedback involves re-estimation of this probability.

Bhuyan and Raghavan (1991) store relevance feedback information from several users in three graphs denoting different relationships between pairs of documents. When enough information has been stored, the graphs are combined and used to form clusters. The system tries to obtain a consistent classification for documents over a number of user interactions. Documents considered jointly relevant by a number of users are placed in the same group, and the clusters are used in later searches to retrieve related documents more efficiently.

The information retrieval systems of Goker and McCluskey (1991), Jennings and Higuchi (1992), Lang (1995), and Krulwich (1995a) all use relevance feedback as part of the learning cycle. The technique is used to inform the system which documents or terms are useful as discriminators of documents that interest the user. These terms are used as features for the learning algorithm. Savoy (1994) extends this technique to hypertext retrieval by using relevance feedback to update and construct links between documents according to their co-appearance in sets of documents relevant to queries. Gordon (1988) uses a genetic algorithm to combine information from several different descriptions of the same document with user relevance judgments. Each document description is assumed to be relevant to a different set of users, and the genetic algorithm attempts to differentially move descriptions closer to relevant queries. Thompson (1991) presents a method for using the statistical technique of combination of expert opinion (CEO) to merge indexers' classifications and relevance feedback from searchers.

Chen (1995) uses relevance feedback in an iterative text categorization scheme using the ID5R version of Quinlan's ID3 algorithm. ID5R is an iterative decision tree learner that rearranges the structure of the tree to accommodate new examples as they are presented. Chen initially presents the algorithm with a set of keyword groups representing positive examples of the desired concept—a similar set of negative examples is also supplied, as ID5R requires both positive and negative examples to learn a concept description. The resulting decision tree is used to search the remainder of the database, and the set of new documents retrieved is presented to the user for the relevance feedback step. The user classifies them as either positive or negative, and this information is used to update the decision tree. This process is repeated until the entire database is classified correctly.

Yang and Korfhage (1994) modify user queries rather than document descriptions. They note that previous research has shown that including term weightings in queries can improve retrieval performance in a system based on the vector-space model; however, it is also well known that users find it difficult to assign correct weights. A genetic algorithm is used to test the fitness of a variety of queries containing different weightings for query terms, fitness being measured by relevance feedback over the document sets retrieved by each weighted form of the query. Tests over the Cranfield Collection (a standard set of documents, queries, and known relevances between documents and queries) show that this technique achieves higher precision than both the original unmodified queries and previously reported results from other relevance feedback techniques on the Cranfield document set.

Belew (1989) and Crouch *et al.* (1994) incorporate user relevance ratings into a connectionist retrieval system. Nodes in the network corresponding to documents judged by the user to be relevant are given a powerful excitatory signal, while nodes corresponding to irrelevant documents are strongly inhibited. The network cycles again, producing a new document ranking—and perhaps adding additional documents to the list of potentially relevant articles. Kwok (1989, 1991) suggests a similar neural network architecture for using relevance feedback to improve rankings by modifying document connections.

Boyan *et al*. (1996) use a range of machine learning methods, including reinforcement learning techniques that propagate rewards through a graph, to improve the rankings returned by an Web search engine based on feedback collected unintrusively from users. The system appears like a regular search engine. Users do not give explicit feedback; instead, the system records which hits they follow. The system is uses the *tf*×*idf* retrieval metric but also takes account of such things as whether the word appears in a title, heading, is bold, italicized, or underlined, and so on. These features appear as weights, and are optimized using simulated annealing.

5 Machine Learning for Information Filtering

In most information retrieval domains a user's interactions with the system occur over an extended period of time. In some cases interaction is a discrete event unrelated to previous interactions, whereas other domains involve continuous interaction around the same topic. For example, a search for books about functional programming languages might be a one-off exercise with a library's computerized cataloguing system, whereas following a thread in a USENET newsgroup would be an ongoing procedure with interactions occurring daily or even hourly. In the former case the user is probably willing to endure an extended dialogue in order to locate the desired material. The outcome may be several books on the subject in question, and the user may not need to return for some time.

In situations where users' interests remain constant and the documents that satisfy those interests are continually changing, it is necessary to model the users' interests to *filter* the continuous stream of information for articles of interest. A user should not have to search the entire USENET feed to find a handful of articles that are probably only of passing interest. The effort required to find these articles is often more than users are willing to spare. In this situation it is necessary to keep track of interesting articles, and separate them from the morass of uninteresting text.

5.1 Modeling user interests

The most effective kind of user modeling occurs over a series of interactions with the system as it builds a profile of each user, a technique which is most often applicable when a user's requirements remain constant. In these situations it is the user's *interests* that the system tries to identify.

Most news reading systems allow users to keep a hotlist of relevant newsgroups, but force the user to rummage through these groups to determine which articles are of interest—usually on the strength of titles or a keyword search. As the number of groups and the range of topics increase, the user's task becomes overwhelming. A trade-off is necessary between looking through fewer articles over the range of interests, and finding postings that are truly interesting (Lang, 1995).

To lessen the strain on the user, some systems build a profile of the user's interests during interactive sessions and present articles that match that profile. This *user model* may be generated collaboratively with the user (Lang, 1995; Krulwich, 1995a) or from logs of user sessions (Goker and McCluskey, 1991), and is continually updated to reflect changes in the user's interests and the content of the text database.

Hull *et al.,* (1995) and Schutze *et al.* (1995) analyze the performance of three learning schemes—linear discriminant analysis, logistic regression, and neural networks—in improving the accuracy of relevant/non-relevant ratings. Their work is unusual in that the document base for their experiments, the Tipster corpus, is of a significant size: 3.2 Gigabytes of text in over one million documents, against which they processed 100 topics corresponding to the routing tasks of TREC-2 and TREC-3. Their results indicate that the classifiers achieve 10-15% higher accuracy (combined recall and precision) than Rocchi-expansion relevance feedback (Buckley *et al.,* 1994), which works from a linear combination of the query vector, the centroid of relevant documents, and sometimes the centroid of irrelevant documents.

Bloedorn *et al.,* (1996) attempt to improve the comprehensibility of user profiles by using a generalization hierarchy. They note that a user interested in documents on scuba, whitewater rafting, and kayaking is more generally described as having an interest in water sports—and that this broader term can then be used to suggest additional topics of interest (such as snorkeling), as well as to more naturally communicate the profile to the user. In practical use of new filtering agents, user comprehension of profiles can be important since the user may need to edit or validate the model learned by the system (Mitchell *et al,* 1994). In the Bloedorn *et al*. (1996) scheme, the term vector for a document is augmented by subject categories (generated by an automated text classifier) and tags describing organization, person, and place names (also automatically extracted). The user marks a sample set of documents as relevant (positive examples) or irrelevant (negative examples), and the profile is induced from this training set by a decision tree learner such as C4.5 (Quinlan, 1992).

The ideas discussed above can also be applied to *user group models*. Instead of building a profile of a single person, the system models a number of users and employs the information to select documents relevant to individuals, and to the group as a whole.

5.2 Information filtering applications

Machine learning has been applied to information filtering in several different domains: news reading, Web browsing, E-mail filtering, and database access.

News reading

Krulwich's system (Krulwich, 1995a; Krulwich, 1995b; Krulwich and Burkey, 1996) filters information from a number of sources in a Lotus Notes environment, using a term vector approach. Documents constitute four groups, ranging from general discussion to memos and bulletins. The number of documents relevant to the average user is estimated at less than a dozen of the 3000 that appear daily. By presenting documents that are predicted to be relevant to the user, and obtaining information on which are relevant and why, the system is able to fine tune each user's filter. During each nightly search of the database, several articles are collected that match the current profile. These documents are presented to the user who scans them for items of interest. As articles are selected the user is asked to indicate why they are interesting, be it the subject area, author, or whatever. These "reasons" become categories for the system to learn.

Lang (1995) describes a similar system, NEWSWEEDER, that learns user models for identifying interesting USENET news articles. In addition to providing the services of a traditional newsreader, NEWSWEEDER generates virtual newsgroups tailored to individual users' preferences. Each user's personalized list is ranked according to a predicted interest rating for each article. The user selects any articles from the list that appear interesting, reads them, and rates them from 1 (essential) to 5 (never want to see anything like it again). All other articles are marked "skip," indicating that the user does not even want to read them. Ratings are collected, and overnight the system learns a new model, based on linear regression, that is used to predict the ratings the user will give to the next set of articles appearing in their personalized newsgroup.

IAN (Green and Edwards, 1996) also filters USENET articles, using either C4.5 (Quinlan, 1993) or the instance-based IBPL algorithm (Payne and Edwards, 1995) to refine its model of user interests. Tests of both algorithms indicate that they are better suited to predicting coarse-grained interest classifications (such as interesting/not interesting) than fine-grained ones (such as level of interest on a graduated scale).

The user model in Jennings and Higuchi's (1992) USENET news filtering system is based on a neural network. An initial set of news articles are retrieved by the user and marked as relevant or irrelevant to their interests. These positive and negative examples are then used to train a neural network. The training features extracted from the documents are based on term vectors, and terms are assigned weightings based on their position in the article—for example, terms appearing in the “Subject” line are weighted more heavily than ones in the main text. Once trained, the network screens incoming articles and ranks them by predicted relevance. The network tracks changing user interests by noting the articles read or rejected during each user session, and later feeding these articles through the network as additional positive and negative examples.

The GroupLens architecture provides collaborative filtering for USENET newsgroups (Resnick *et al*., 1994). The algorithm that predicts a user’s interest in an unseen newsgroup posting is based on the common-sense heuristic that people who have agreed on postings in the past are likely to continue to agree in the future, at least for articles in the same newsgroup. Several techniques have been tested for correlating user ratings, including reinforcement learning and multivariate regression.

Web browsing

This type of profile development is also readily applicable to WWW browsing. The LIRA system (Balabanovic and Yun, 1995; Balabanovic and Shoham, 1995) develops a profile of user preferences in WWW page contents. The user model is based on *tf*×*idf* (term-frequency times inverse document frequency) weightings of terms found on previously examined WWW pages. Each morning the user is presented with a set of WWW pages to evaluate on an eleven-point scale, and these ratings are then used to adjust the weights in the user profile vector. LAW (Edwards *et al*., 1995) can interactively identify links of potential interest on a page that the user is browsing, and also autonomously searches for WWW pages matching the user profile. The profile updating algorithm is based on the user’s response to suggestions: if the user saves as a bookmark, prints, or frequently visits a link or WWW page, then it is classified as a positive example, otherwise it is a negative example. Experiments with profile updating schemes indicate that rule induction techniques such as C4.5 (Quinlan, 1992) and instance-based learning algorithms such as IBPL1 (Payne and Edwards, 1995) can achieve better predictive accuracy than simple *tf*×*idf* profile updates. The AARON system (Green and Edwards, 1996) uses a similar profile construction approach to construct a personalized “newspaper” from news items previously gathered from a variety of sites.

The WWW page recommendation system of Pazzani *et al*. (1995) also learns from both positive and negative examples, taken from a user-constructed hotlist and “coldlist” (set of URLs for Web pages that the user visited, but did not like). It learns separate profiles for each user topic, under the assumption that multiple profiles will be more accurate for each interest than a single conglomerate user model. Pannu and Sycara (1996) describe agent software that scans the WWW for conference announcements and requests for proposals that may be of interest to the user. The user's preferences are learned from a training set whose positive examples are papers and proposals written by the user, and whose negative examples are documents written by faculty working in other fields. *Tf*×*idf* and two neural networks were tested for updating the user's profile, with the former producing the best accuracy in terms of classification of new documents.

WebWatcher (Joachims *et al*., 1995 ; Armstrong *et al*., 1995) learns user preferences for link traversal across the WWW, and employs this model to suggest hyperlinks for the user to follow. The user first briefly describes the goals of an information search to WebWatcher: the types of information needed (for example, a paper or a description of software), a subject area, the name of a relevant author, etc. As the user explores a set of connected pages, WebWatcher highlights links that appear to be relevant to the search, where relevance is measured by similarity between keywords in the user's description of the search goals and keywords that relate to the available hyperlinks. The latter are terms in the link itself, in the sentence containing the link, and in the heading of the document associated with the link. WebWatcher's recommendations are refined offline by induction over a set of user interactions. Limited testing with three techniques—a Boolean concept learner, the compression algorithm Wordstat, and *tf*×*idf*—indicates that all three can potentially improve WebWatcher's accuracy in predicting the user's link traversal preferences.

The Letizia web-browsing agent browses the WWW in parallel with a user (Lieberman, 1995; Lieberman and Maulsby, 1996). As the user visits Web pages, Letizia incrementally builds a user model and uses the model to autonomously locate "interesting" pages. These pages are presented to the user only at the user's request. The user model is based on the content of the documents that the user manually selects (using a similarity measure based on *tf*×*idf*), and on heuristics related to the user's actions. For example, a page from which the user has selected several links will be treated as a positive example of an interesting document, while a page that the user scans quickly without selecting one of its links is treated as a negative example.

The FAB Web page recommendation system (Balabanovic, 1997; Balabanovic and Shoham, 1997), a follow-on to the LIRA system described above, is based on a collaborative filtering architecture that also includes adaptive "collection agents" that locate and retrieve documents in the system's recommendations database. Like users, these agents have profiles. *Search agents* use a best-first or beam search strategy to scan the Web directly for documents that match the agent's profile, while *index agents* automatically construct queries to Web indexers to locate new pages. A user's relevance rating is used to update the profile of the agent that originally located that page. It is hypothesized that this type of feedback will eventually result in agent specialization, each agent's profile evolving to represent a particular concept or document type. The process is accelerated by periodically eliminating collection agents whose retrieved documents prove unpopular with FAB's users.

In an interesting twist on Web monitors, the Do-I-Care agent (Starr *et al.*, 1996) monitors a set of WWW pages for interesting *changes*. Where a URL-minder notifies users of any modification to pre-selected pages, Do-I-Care uses machine learning techniques (in this version, Bayesian classification) to learn regularities in the types of change that the user finds interesting (for example, a significant increase in the document size or the addition of heavily weighted terms). Another imaginative approach is the application of Artificial Life to the WWW (Menczer *et al*., 1995). A population of agents, evolved using density-dependent selection, locate information for the user. Agents compete for relevant documents and gain the energy necessary for survival as a reward for presenting the user with appropriate information.

Finally, Cohen and Singer (1996b) address the problem of maintaining a WWW resource directory page—a list of pointers to documents on a given topic. Keeping these directories current is difficult, given the exponential growth of the Web. Their system begins with an existing directory, which is treated as a list of positive examples for the directory's topic. The documents linked to this directory are retrieved and fed into a rule learning system (in this case, RIPPER, Cohen, 1995b, or the *sleeping experts* algorithm, Cohen and Singer, 1996a). The rules are then translated into search engine queries that are periodically run to search for new documents to include in the directory.

E-mail filtering

Several systems have been developed to filter personal e-mail. Cohen (1996b) uses the RIPPER algorithm (Cohen 1995b) to induce mailbox classification rules from the contents of a user's personal mail files. Rules are based on the presence of a term in a given field of a message—for example, "classify an incoming message in the *call-for-papers* mailbox if it contains *cfp* and *95* in the subject field". The Maxims e-mail agent learns more complex tasks: it can prioritize, delete, forward, and archive messages as well as categorize them (Lashkari *et al.*, 1994). Maxims is based on the programming-by-demonstration paradigm; it observes the user dealing with email, stores situation/action pairs describing these observations, and predicts future actions based on matches to stored situations. As well as predicting the next action, Maxims also generates an estimate of its confidence in that prediction. This confidence level is based on intuitively important factors: the number of examples that the agent has seen, the user's consistency in handling the most closely matching prior situations, and the degree of similarity between the new situation and previous ones. Obviously, the longer that Maxims observes a user and the more consistent that user is in dealing with email, the better the predictions.

Database access

The OKAPI system (Goker and McCluskey, 1991) provides retrieval services for three databases at City University in London. OKAPI ranks retrieved documents by the probability that they are relevant a the user's query. This probability is calculated using a modified version of the term weighting formula in Robertson and Sparck Jones (1994). The system also uses standard relevance feedback techniques to expand the user's initial query. Goker and McCluskey (1991) describe an incremental learning algorithm that, when used with OKAPI, forms a model of users' areas of interest. The algorithm learns from logs of user sessions that contain details of search terms, search and response timings, number of references examined, and other information. The concept descriptions formed can be used next time a user carries out a search on a similar subject, with emphasis on alleviating weight block problems.

5.3 Augmenting models from other information sources

A significant problem for information filtering is the time that it takes to learn a model of each user's interests by monitoring their interactions. Kroon *et al.* (1996) reduce the model construction period by priming the model with information garnered from the user's home page, bookmarks, and responses to a brief questionnaire. AutoClass (Cheeseman *et al.*, 1988), a Bayesian clustering algorithm, was used to produce the initial user interest model. This algorithm appears particularly appropriate for the task, as the number of sub-classes—the different, potentially diverse and numerous, areas of interest that comprise a given user's model—does not have to be specified in advance.

For the systems described above, each user's filtering profile is developed (learned) in isolation. The Webhound WWW document recommendation system (Lashkari, 1995) uses an alternative retrieval technique: collaborative filtering. Rather than starting by correlating an individual user with a document, collaborative filtering bases its recommendations on correlations between users. Webhunter presents a user with a set of WWW pages to evaluate, and each person has a personal agent that records which Web pages that the user liked or disliked. The agent then compares itself with other agents, looking for ones with similar values for similar items. Highly correlated agents will accept recommendations from one another.

As a further refinement, Webhunter includes limited content analysis of documents to partition the document space. The idea is that while two users may both find one type of resource interesting, they may vehemently disagree on the usefulness of a different one. Reliance on a simple user correlation may mask these differences and lead to less-reliable Web page suggestions. Further, the set of WWW pages known to Webhunter is large in comparison with the number of pages that any individual user will have the patience to evaluate. This makes it less likely that any two users will have examined a large proportion of the same pages, which affects the ability of the agents to notice similarities between users. By using content analysis to group the WWW pages into clusters, it is easier to find correlations between user tastes.

6 Conclusions

Machine learning is becoming an important source of tools for automating information retrieval tasks. All phases of information retrieval can be (and are) performed manually, but automation has many benefits—larger document collections can be processed more quickly and consistently, and new techniques can be easily implemented and tested. Human ability at these tasks provides a benchmark performance threshold that many systems are approaching. However, perfect performance is probably unattainable in the foreseeable future because the models of natural languages used are necessarily very restricted.

The most important phase of the information retrieval process is *feature selection*, which we discussed at length in Sections 2.1 and 2.2. This is particularly significant when learning algorithms are employed at some point in the process. Although machine learning schemes perform feature selection themselves, they work best when given a small number of distinctive features to learn from. Limiting the feature set reduces the space and time consumed by the algorithm, and lessens the risk that poor features appear in concept descriptions by chance.

Simple techniques for eliminating potentially weak features often work well once the nature of the features to be used is determined. Most systems use the word as the basic unit of content, and stoplists, stemmers, and thesauri help to reduce the number of possible words a system has to cope with. When objects larger than words, such as phrases and headings, are used as features, the need for selection is even more critical. It is possible to determine the extent of such features using syntactic analysis, and even formatting information. Coarsergrained features do not suffer from problems such as synonymy, so the co-occurrence of a phrase in two documents is a better indication of similarity than the appearance of individual words.

The *term-frequency-times-inverse-document-frequency* measure of document similarity, coupled with the cosine rule to determine similarity, is one of the simplest and most effective automatic methods of classifying text. The fact that it uses the term vector representation for documents suggests that information about the distribution of important words is often all that is necessary for effective retrieval. The measure is resistant to the effects of stopwords and document length. Standard machine learning schemes using the same form of text representation are less resilient, requiring the data to be prefiltered to reduce the number of features. To keep space and time consumption to practical levels the feature space must be limited to several hundred words. This is generally a minute fraction of the total vocabulary of a text collection, so a good feature selection mechanism is necessary. When the number of classes and the number of features are suitably limited, machine learning algorithms generate concise and intuitive concept descriptions.

Four main phases of the information retrieval process are identified by Lewis (1991), and learning techniques have been applied to all of them. The first, *indexing*, incorporates feature extraction, document clustering, and text classification. Feature selection has already been noted as the most vital part of the process, and most researchers expend considerable effort in this area. Document clustering assists in retrieval by creating links between similar documents. This allows related documents to be retrieved once one of the documents has been deemed relevant to a query. Links may be made at any level, from common keywords to similar paragraph subtopics. By following links to an arbitrary level the system can retrieve documents to a given degree of similarity, and rank them accordingly.

The remaining three phases, *querying*, *comparison*, and *feedback*, usually form a loop which is repeated until the user is satisfied with the retrieved documents. This iterative process reduces the effort required of the user. Most users find it difficult to formulate their information needs in terms of keywords, and selection of “correct” words is a stressful task. Systems that use feedback about the relevance of retrieved documents help the user by strengthening the weight of discriminatory terms present in the query, and introducing new terms garnered from relevant documents.

The ease with which an effective query can be entered is enhanced by modeling the user. Creating a representation of the user’s information needs reduces the effort required to select relevant documents in a changing information environment, and continual updating of the model, by means of user feedback, lets the system track shifts in interest and information need. By creating a model of each user's background and experience the system can interpret individual users' queries appropriately.

Machine learning technology has been successfully applied in many information retrieval systems, both experimental and fielded. The learning techniques currently employed are simple, and some remaining problems may be overcome using more advanced technology. Machine learning is a rapidly growing field, and new algorithms and techniques are continually pushing the limits of performance higher. But what is really driving the applications of machine learning to information retrieval is not so much developments in machine learning technology as changes in our working environment that demand new ways of operating. The instant availability of enormous amounts of textual information on the Internet and in digital libraries has provoked a new interest in software agents that act on behalf of users, sifting through what is there to identify documents that may be relevant to users' individual needs. The application of machine learning to information retrieval is only just beginning.

References

Aha, D. (1992) "Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms." *International Journal of Man-Machine Studies* 36, pp. 267–287.

Allan, J. (1995) "Relevance feedback with too much data." *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval,* (Seattle, Washington, USA), pp. 337–343.

Aone, C., Bennett, S.W., and Gorlinsky, J. (1996) "Multi-media fusion through application of machine learning and NLP." *Proceedings of the AAAI Symposium on Machine Learning in Information Access*, (Stanford, CA, USA).

Apte, C., Dameru, F. and Weiss, S.M. (1994a) "Automated learning of decision rules for text categorization." *ACM Transactions on Information Systems* 12(3), pp. 233–250.

Apte, C., Dameru, F. and Weiss, S.M. (1994b) "Towards language independent automated learning of text categorization models." *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, (Dublin, Ireland), pp. 23–30.

Armstrong, R. Freitag, D., Joachims, T. and Mitchell, T. (1995) "WebWatcher: A learning apprentice for the World Wide Web." *Proceedings of the AAAI Symposium on Information Gathering from Heterogeneous, Distributed Environments,* (Stanford, CA, USA), pp. 6–12.

Baeza-Yates, R.S. (1992) "Introduction to data structures and algorithms related to information retrieval." In *Information retrieval: Data structures and algorithms*, William B. Frakes and Ricardo Baeza-Yates eds., Prentice Hall, Englewood Cliffs, New Jersey.

Balabanovic, M., Shoham, Y., and Yun, Y. (1995) "An adaptive agent for automated web browsing." Technical Report No. SIDL-WP-1995-0023. University of Stanford, California.

Balabanovic, M., and Shoham, Y. (1995) "Learning information retrieval agents: Experiments with automated Web browsing." *Proceedings of the AAAI Symposium on*

Information Gathering from Heterogeneous, Distributed Environments (Stanford, CA, USA), pp. 13–18.

Balabanovic, M. (1997) "An adaptive Web page recommendation service." To appear in *Proceedings of the International Conference on Autonomous Agents*, March.

Balabanovic, M., and Shoham, Y. (1997) "Combining content-based and collaborative recommendation." To appear in *Communications of the ACM*, (Marina Del Ray, CA, USA).

Bartell, B.T., Cottrell, G.W., and Belew, R.K. (1994) "Optimizing parameters in a ranked retrieval system using multi-query relevance feedback." *Proceedings of the Annual Symposium on Document Analysis and Information Retrieval,* (Las Vegas, NV, USA).

Belew, R.K.(1989) "Adaptive information retrieval: Using a connectionist representation to retrieve and learn about documents." *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval,* (Cambridge, MA, USA), pp. 3–10.

Bell, T.C., Cleary, J.G. and Witten, I.H. (1990) *Text Compression*. Prentice Hall, Englewood Cliffs, New Jersey.

Bhatia, S.J., Deogun, J.S. and Raghavan, V.V. (1995) "Conceptual query formulation and retrieval." *Journal of Intelligent Information Systems* 5(3), pp. 183–209.

Bloedorn, E., Mani, I., and MacMillan, T.R. (1996) "Representational issues in machine learning of user profiles." *Proceedings of the AAAI Symposium on Machine Learning in Information Access*, (Stanford, CA, USA).

Bhuyan, J.N. and Raghavan, V.V. (1991) "A probabilistic retrieval scheme for cluster-based adaptive information retrieval." *Proceedings of the International Workshop on Machine Learning,* (Evanston, Illinois), pp. 240–244.

Boyan, J., Freitag, D., and Joachims, T. (1996) "A machine learning architecture for optimizing web search engines." *Proceedings of the AAAI Workshop on Internet-based Information Systems,* (Portland, OR, USA), pp. 1–8.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. (1984) *Classification and Regression Trees.* Wadsworth Inc., Belmont, California.

Buckley, C., Salton, G., and Allan, J. (1994) "The effect of adding relevance information in a relevance feedback environment." *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval,* (Dublin, Ireland), pp. 92–300.

Buntine, W. (1990) *A theory of learning classification rules*. PhD thesis, School of Computing Science, University of Technology, Sydney (Australia).

Callan, J.P., Lu, Z., and Croft, W.B. (1995) "Searching distributed collections with inference networks." *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval,* (Seattle, WA, USA), pp. 21–28.

Carbonell, J. (Editor) (1990) *Machine learning: Paradigms and methods*. Bradford Books, MIT Press, Cambridge, Massachusetts.

Cheeseman, P., Kelly, J., Self, M., Stutz., J., Taylor, W., and Freeman, D. (1988) "AUTOCLASS: A Bayesian classification system." *Proceedings of the International Conference on Machine Learning,* pp. 54–64.

Chen, H. (1995) "Machine learning for information retrieval: Neural networks, symbolic learning, and genetic algorithms." *Journal of the American Society for Information Science* 46(3), pp. 194–216.

Cleary, J.G. and Trigg, L.E. (1995) "K*: An instance-based learner using an entropic distance measure." *Proceedings of the International Conference on Machine Learning,* (Tahoe City, CA, USA), pp. 108–114.

Cohen, W.W. (1995a) "Text categorization and relational learning." *Proceedings of the International Conference on Machine Learning*, (Tahoe City, CA, USA), pp. 124–132.

Cohen, W.W. (1995b) “Fast effective rule induction.” *Proceedings of the International Conference on Machine Learning*, (Tahoe City, CA, USA), pp. 115–123.

Cohen, W.W. (1996a) “Learning with set-valued features.” *Proceedings of the National Conference on Artificial Intelligence*, (Portland, OR, USA).

Cohen, W.W. (1996b) “Learning rules that classify e-mail.” *Proceedings of the AAAI Symposium on Machine Learning in Information Access,* (Stanford, CA, USA).

Cohen, W.W. and Singer, Y. (1996a) “Context-sensitive methods for text categorization.” *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval,* (Zurich, Switzerland).

Cohen, W.W. and Singer, Y. (1996b) “Learning to query the Web.” *Proceedings of the AAAI Workshop on Internet-based Information Systems,* (Portland, OR, USA), pp. 16–25.

Conrad, J.G. and Utt, M.H. (1994) “A system for discovering relationships by feature extraction from text databases.” *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, (Dublin, Ireland). pp. 260–270.

Crawford, S.L., Fung, R.M., Appelbaum, L.A. and Tong, R.M. (1991) “Classification trees for information retrieval.” *Proceedings of the International Workshop on Machine Learning,* (Evanston, Illinois), pp. 245–249.

Crouch, C.C., Crouch, D.B., and Nareddy, K. (1994) “Associative and adaptive retrieval in a connectionist system.” *International Journal of Expert Systems* 7(2), pp. 193–202.

Cunningham, S. and Summers, B. (1995) “Applying machine learning to subject classification and subject description for information retrieval.” *Proceedings of Artificial Neural Networks and Expert Systems,* (Dunedin, New Zealand), pp. 243–246.

Dasigi, V., and Mann, R.C. (1996) “Neural net learning issues in classification of free text documents.” *Proceedings of the AAAI Symposium on Machine Learning in Information Access*, (Stanford, CA, USA).

Deerwester, S., Dumais, S.T., Landauer, T.K., Furnas, G.W. and Harshman, R.A. (1990) “Indexing by latent semantic analysis.” *Journal of the Society for Information Science* 41(6), pp. 391–407.

Edwards, P., Bayer, D., Green, C.L., and Payne, T.R. (1995) “Experience with learning agents which manage Internet-based information.” *Proceedings of the AAAI Symposium on Machine Learning in Information Access*, (Stanford, CA, USA).

Fisher, D. (1987) “Knowledge acquisition via incremental conceptual clustering.” *Machine Learning 2,* pp. 139–172.

Frakes, W.B. (1992a) “Introduction to information storage and retrieval systems.” In *Information retrieval: Data structures and algorithms*, William B. Frakes and Ricardo Baeza-Yates eds., Prentice Hall, Englewood Cliffs, New Jersey.

Frakes, W.B. (1992b) “Stemming algorithms.” In *Information retrieval: Data structures and algorithms*, William B. Frakes and Ricardo Baeza-Yates eds., Prentice Hall, Englewood Cliffs, New Jersey.

Frakes, W.B. and Baeza-Yates, R. (Editors) (1992) *Information retrieval: Data structures and algorithms*, Prentice Hall, Englewood Cliffs, New Jersey.

Fuhr, N., and Buckley, C. (1991) “A probabilistic learning approach for document indexing.” *ACM Transactions on Information Systems* 9(3), pp. 223–248.

Furnas, G.W. (1985) “Experience with an adaptive indexing scheme.” *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, (San Francisco, CA, USA), pp. 131–135.

Gaines, B.R. (1991) “The tradeoff between knowledge and data in knowledge acquisition.” In *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W.J. Frawley, eds., AAAI Press, Menlo, California.

Ganascia, J.-G. (1991) “Deriving the learning bias from rule properties.” In Hayes, J.E., Mitchie, D., and Tyngu, E. (eds.), *Machine Intelligence* 12. Oxford: Clarendon Press, pp. 151–167.

Goker, A. and McCluskey, T.L. (1991) “Incremental learning in a probabilistic information retrieval system.” *Proceedings of the International Workshop on Machine Learning*, (Evanston, Illinois), pp. 255–259.

Goldberg, J.L. (1995) “CDM: An approach to learning in text categorization.” *Proceedings of the International Conference on Tools with Artificial Intelligence,* (Washington, DC, USA), pp. 258–265.

Gordon, M. (1988) “Probabilistic and genetic algorithms for document retrieval.” *Communications of the ACM* 31:10. pp. 1208–1218

Gordon, M.D. (1989) “User-based document clustering by redescribing subject descriptions with a genetic algorithm.” *Journal of the American Society for Information Science* 42(5), pp. 311–322.

Green, C.L, and Edwards, P. (1996) “Using machine learning to enhance software tools for Internet information management.” *Proceedings of the AAAI Workshop on Internet-based Information Systems,* (Portland, OR, USA), pp. 48–56.

Grefenstette, G. (1994) *Explorations in automatic thesaurus discovery*. Kluwer, Boston, Massachusetts.

Guntzer, U., Juttner, G., Seegmuller, G., and Sarre, F. (1989) “Automatic thesaurus construction by machine learning from retrieval sessions.” *Information Processing and Management* 25(3), pp. 265–273.

Haines, D. and Croft, W.B. (1993) “Relevance feedback and inference networks.” *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, (Pittsburgh, Pennsylvania), pp. 2–11.

Harman, D. (1992) “Relevance feedback and other query modification techniques.” In *Information retrieval: Data structures and algorithms*, William B. Frakes and Ricardo Baeza-Yates eds., Prentice Hall, Englewood Cliffs, New Jersey.

Hearst, M.A. (1993) “Cases as structured indexes for full-length documents.” *Proceedings of the Symposium on Case-based Reasoning and Information Retrieval*, pp. 140–145.

Hearst, M.A. (1994) “Context and structure in automated full-text information access.” Technical Report No. UCB/CSD-94/836, Computer Science Division (EECS), University of California, Berkeley, California.

Hendler, J.A. (1997) “Intelligent agents: Where AI meets information technology.” *IEEE Expert* 11(6), pp. 20–23.

Holte, R. (1993) “Very simple classification rules perform well on most commonly used datasets.” *Machine Learning* 11(1), pp. 63–90.

Hull, D., Pedersen, J., and Schutze, H. (1995) “Document routing as statistical classification.” *Proceedings of the AAAI Symposium on Machine Learning in Information Access*, (Stanford, CA, USA).

Jennings, A. and Higuchi, H. (1992) “A browser with a neural network user model.” *Library Hi Tech* 10(1–2). pp. 77–93.

Joachims, T., Mitchell, T., Freitag, D. and Armstrong, R. (1995) “WebWatcher: Machine learning and hypertext.” *Fachgruppentreffen Maschinelles Lernen*, Dortmund, Germany; August.

Jones, W.P., and Furnas, G.W. (1987) “Pictures of relevance: A geometric analysis of similarity measures.” *Journal of the American Society for Information Science* 38(6), pp. 420–442.

Kelly, G. (1963) *A theory of personality: The psychology of personal constructs*. W.W. Norton and Co., New York, New York.

Kroon, H.C.M. de, Mitchell, T.M., and Kerckhoffs, E.J.H. (1996) "Improving learning accuracy in information filtering." *Proceedings of the ICML Workshop on Machine Learning Meets Human Computer Interaction*, pp. 41–50.

Krulwich, B. (1995a) "Learning document category descriptions through the extraction of semantically significant phrases." *Proceedings of the IJCAI Workshop on Data Engineering for Inductive Learning*, (Montreal, Canada).

Krulwich, B. (1995b) "Learning user interests across heterogeneous document databases." *Proceedings of the AAAI Symposium on Information Gathering from Heterogeneous, Distributed Environments.* (Stanford, CA, USA), pp. 106–110.

Krulwich, B., and Burkey, C. (1996) "Learning user information interests through the extraction of semantically significant phrases." *Proceedings of the AAAI Symposium on Machine Learning in Information Access*, (Stanford, CA, USA).

Kwok, K.L. (1989) "A neural network for probabilistic information retrieval." *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval,* (Cambridge, MA, USA), pp. 21–31.

Kwok, K.L. (1991) "Query modification and expansion in a network with adaptive architecture." *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, (Chicago, Illinois, USA), pp. 192–201.

Lang, K. (1994) "NewsWeeder: An adaptive multi-user text filter." Internal Research Report, Carnegie Mellon University; August.

Lang, K. (1995) "NewsWeeder: Learning to filter Netnews." *Proceedings of the International Conference on Machine Learning,* (Tahoe City, California), pp. 331–339

Langley, P. (1996) *Elements of machine learning*. Morgan Kaufmann, San Francisco, CA.

Langley, P. and Simon, H.A. (1995) "Applications of machine learning and rule induction." *Communications of the ACM* 38(11), pp. 55–64.

Lashkari, Y., Metral, M., and Maes, P. (1994) "Collaborative interface agents." *Proceedings of the National Conference on Artificial Intelligence*, (Cambridge, Mass, USA), Vol. 1, pp. 444–449.

Lashkari, Y. (1995) "The Webhunter personalized document filtering system." <http://webhound.www.media.mit.edu/projects/webhound/>

Lehnert, W., Soderland, S., Aronow, D., Feng, F. and Shmueli, A. (1995) "Inductive text classification for medical applications." *Journal of Experimental and Theoretical Artificial Intelligence* 7, pp. 49–80.

Lewis, D.D. (1991) "Learning in intelligent information retrieval." *Proceedings of the International Workshop on Machine Learning*, (Evanston, Illinois), pp. 235–239.

Lewis, D.D. and Croft, W.B. (1990) "Term clustering of syntactic phrases." *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval,* (Brussels, Belgium), pp. 385–404.

Lewis, D.D. and Gale, W.A. (1994) "A sequential algorithm for training text classifiers." *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, (Dublin, Ireland), pp. 3–12.

Lewis, D.D. and Ringuette, M. (1994) "A comparison of two learning algorithms for text categorization." *Proceedings of the Annual Symposium on Document Analysis and Information Retrieval,* (Las Vegas, NV, USA), pp. 81–93.

Lieberman, H. (1995) "Letizia: An agent that assists Web browsing." *Proceedings of the International Joint Conference on Artificial Intelligence,* (Montreal, Canada).

Lieberman, H., Maulsby, D. (1996) "Instructible agents: Software that just keeps getting better." *IBM Systems Journal* 35(3/4), pp. 539–556.

Liere, R., and Tadepalli, P. (1996) "The use of active learning in text categorization." *Proceedings of the AAAI Symposium on Machine Learning in Information Access,* (Stanford, CA, USA).

Martin, J.D. (1995) "Clustering full text documents." *Proceedings of the IJCAI Workshop on Data Engineering for Inductive Learning at IJCAI-95,* (Montreal, Canada).

Maron, M.E. (1961) "Automatic indexing: An experimental inquiry." *Journal of the ACM* 8, pp. 404–417.

Menczer, F., Belew, R.K., and Willuhn, W. (1995) "Artificial life applied to adaptive information agents." *Proceedings of the AAAI Symposium on Information Gathering from Heterogeneous, Distributed Environments* (Stanford, CA, USA).

Mitchell, T., Caruana, R., Freitag, D., McDermott, J., and Zabowski, D. (1994) "Experience with a learning personal assistant." *Communications of the ACM* 37(7), pp. 81–91.

Moffat, A. (1989) "Word based text compression." *Software—Practice and Experience* 19(2), pp. 185–198.

Moulinier, I. (1996) "A framework for comparing text categorization approaches." *Proceedings of the AAAI Symposium on Machine Learning in Information Access,* (Stanford, CA, USA).

Pannu, A.S., and Sycara, K. (1996) "Learning text filtering preferences." *Proceedings of the AAAI Symposium on Machine Learning And Information Access* (Stanford, CA, USA).

Pazzani, M., Muramatsu, J., and Bilsus, D. (1996) "Syskill & Webert: Identifying interesting web sites." *Proceedings of the AAAI Symposium on Machine Learning in Information Access*, (Stanford, CA, USA).

Quinlan, J.R. (1986) "Induction of decision trees." *Machine Learning* 1, pp. 81–106.

Quinlan, J.R. (1990) "Learning logical definitions from relations." *Machine Learning* 5, pp. 239–266.

Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Payne, T., and Edwards, P. (1995) "Interface agents that learn: An investigation of learning issues in a mail agent interface." *Technical Report AUCS/TR9508*, Department of Computing Science, University of Aberdeen, Scotland. (To appear in *Applied Artificial Intelligence*, 1997, Vol. 11(1).)

Pazzani, M., Nguyen, L, and Mantik, S. (1995) "Learning from hotlists and coldlists: Towards a WWW information filtering and seeking agent." *Proceedings of the Annual Conference on Tools with Artificial Intelligence* (Washington, DC, USA).

Piatetsky-Shapiro, G. and Frawley, W.J. (Editors) (1991) *Knowledge discovery in databases*. AAAI Press/MIT Press, Menlo Park, CA.

Resnick, P., Iacouvou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994) "GroupLens: An open architecture for collaborative filtering of Netnews." *Proceedings of the ACM Conference on Computer Supported Cooperative Work,* (Chapel Hill, NC, USA), pp. 175–186.

Rissanen, J. (1985) "Minimum description length principle," *Encyclopedia of the Statistical Sciences, Vol. 5,* edited by S. Kotz and N.L. Johnson. Wiley, NY, pp. 523–527.

Robertson, S.E. and Sparck Jones, K. (1994) "Simple, proven approaches to text retrieval." Technical Note, Department of Information Science, City University/Computer Laboratory, University of Cambridge; November.

Sahami, M., Hearst, M., and Saund, E. (1996) "Applying the multiple cause mixture model to text categorization." *Proceedings of the AAAI Symposium on Machine Learning in Information Access,* (Stanford, CA, USA).

Salton, G. (1988) *Automatic text processing: The transformation, analysis, and retrieval of information by computer*. Reading, MA: Addison-Wesley.

Salton, G., Allan, J. and Buckley, C. (1994) "Automatic structuring and retrieval of large text files." *Communications of the ACM* 37(2), pp. 97–108.

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

Salzberg, S.L. (1990) *Learning with nested generalized exemplars*. Kluwer Academic Publishers, Norwell, Massachusetts.

Savoy, J. (1994) "A learning scheme for information retrieval in hypertext." *Information Processing and Management* 30(4), pp. 515–533.

Schutze, H., Hull, D.A., and Pedersen, J.O. (1995) "A comparison of classifiers and document representations for the routing problem." *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval,* (Seattle, WA, USA), pp. 229–237.

Seshardi, V., Weiss, S.M. and Sasisekharan, R. (1995) "Feature extraction for massive data mining." *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, (Montreal, Canada), pp. 258–262.

Soderland, S. and Lehnert, W. (1995) "Learning domain-specific discourse rules for information extraction." *Proceedings of the AAAI Symposium on Empirical Methods in Discourse Interpretation and Generation*.

Starr, B., Ackerman, M.S., and Pazzani, M. (1996) "Do I Care?—Tell me what's changed on the Web." *Proceedings of the AAAI Symposium on Machine Learning in Information Access*, (Stanford, CA, USA).

Thompson, P. (1991) "Machine learning in the combination of expert opinion approach to IR." *Proceedings of the International Workshop on Machine Learning*, (Evanston, Illinois, USA), pp. 270–274.

Towell, G., Voorhees, E.M., Gupta, N.K. and Johnson-Laird, B. (1995) "Learning collection fusion strategies for information retrieval." *Proceedings of the International Conference on Machine Learning,* (Tahoe City, CA, USA), pp. 540–548.

Turtle, H.R., and Croft, W.B. (1990) "Inference networks for document retrieval." *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, (Brussels, Belgium), pp. 1–24.

Turtle, H.R., and Croft, W.B. (1991) "Evaluation of an inference network-based retrieval model." *ACM Transactions on Information Systems* 9(3), pp. 187–222.

Turtle, H.R., and Croft, W.B. (1992) "A comparison of text retrieval models." *Computer Journal* 35(3), pp. 279–290.

Tzeras, K., and Hartmann, S. (1993) "Automatic indexing based on Bayesian inference networks." *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, (Pittsburgh, PA, USA), pp. 22–34.

Utgoff, P.E. (1989) "Incremental induction of decision trees." *Machine Learning* 4, pp. 161–186.

van Rijsbergen, C.J. (1979) *Information retrieval*. London, England: Butterworths.

Voorhees, E.M., Gupta, N.K., and Johnson-Laird, B. (1995) "Learning collection fusion strategies." *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval,* (Seattle, WA, USA), pp. 172–179.

Wiener, E., Pedersen, J., and Weigend, A.S. (1995) “A neural network approach to topic spotting.” *Proceedings of the Symposium on Document Analysis and Information Retrieval,* (Las Vegas, NV, USA), pp. 317–332.

Wilkinson, R., and Hingston, P. (1991) “Using the cosine measure in a neural network for document retrieval.” *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, (Chicago, Illinois, USA), pp. 202–210.

Wilkinson, R. and Hingston, P. (1992) “Incorporating the vector space model in a neural network used for information retrieval.” *Library Hi Tech* 10(1/2). pp. 69–76.

Willett, P. (1988) “Recent trends in hierarchical document clustering: A critical review.” *Information Processing and Management* 24(5), pp. 577–597.

Witten, I.H., Moffat, A. and Bell, T.C. (1994) *Managing Gigabytes*. Van Nostrand Reinhold, New York, New York.

Wong, S.K.M, Cai, Y.J., and Yao, Y.Y. (1993) “Computation of term associations by a neural network.” *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, (Pittsburgh, PA, USA), pp. 107–115.

Yang, J.-J., and Korfhage, R.R. (1994) “Query modification using genetic algorithms in vector space models.” *International Journal of Expert Systems* 7(2), pp. 165–191.

Yang, Y. (1994) “Expert Network: Effective and efficient learning from human decisions in text categorization and retrieval.” *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, (Dublin, Ireland), pp. 13–22.

Yang, Y. (1996) “Sampling strategies and learning efficiency in text categorization.” *Proceedings of the AAAI Symposium on Machine Learning in Information Access*, (Stanford, CA, USA).