A Machine Learning Approach to Inductive Query by Examples: An Experiment Using Relevance Feedback, ID3, Genetic Algorithms, and Simulated Annealing

Hsinchun Chen, Ganesan Shankaranarayanan, and Linlin She
MIS Department, College of Business and Public Administration, University of Arizona, Tucson, AZ 85721. E-mail: hchen@bpa.arizona.edu; gshankar@bpa.arizona.edu; lshe@bpa.arizona.edu

Anand Iyer
I2 Technologies, Inc., 909 E. Las Colinas Blvd., Irving, TX 75039. E-mail: Iyer@I2.com

Information retrieval using probabilistic techniques has attracted significant attention on the part of researchers in information and computer science over the past few decades. In the 1980s, knowledge-based techniques also made an impressive contribution to "intelligent" information retrieval and indexing. More recently, information science researchers have turned to other newer inductive learning techniques including symbolic learning, genetic algorithms, and simulated annealing. These newer techniques, which are grounded in diverse paradigms, have provided great opportunities for researchers to enhance the information processing and retrieval capabilities of current information systems. In this article, we first provide an overview of these newer techniques and their use in information retrieval research. In order to familiarize readers with the techniques, we present three promising methods: The symbolic ID3 algorithm, evolution-based genetic algorithms, and simulated annealing. We discuss their knowledge representations and algorithms in the unique context of information retrieval. An experiment using a 8000-record COMPEN database was performed to examine the performances of these inductive query-by-example techniques in comparison with the performance of the conventional relevance feedback method. The machine learning techniques were shown to be able to help identify new documents which are similar to documents initially suggested by users, and documents which contain similar concepts to each other. Genetic algorithms, in particular, were found to out-perform relevance feedback in both document *recall* and *precision.* We believe these inductive machine learning techniques hold promise for the ability to analyze users' preferred documents (or records), identify users' underlying information needs, and also suggest alternatives for search for database management systems and Internet applications.

Received February 26, 1996; revised February 3, 1997 and July 15, 1997; accepted July 15, 1997.

© 1998 John Wiley & Sons, Inc.

1. Introduction

In the past few decades, the availability of cheap and effective storage devices and information systems has prompted the rapid growth and proliferation of relational, graphical, and textual databases. Information collection and storage efforts have become easier, but the amount of effort required to retrieve relevant information has become significantly greater, especially in large-scale databases. This situation is particularly evident for textual databases, which are widely used in traditional library science environments, in business applications (e.g., manuals, newsletters, and electronic data interchanges), and in scientific applications (e.g., electronic community systems and scientific databases). Information stored in these databases often has become voluminous, fragmented, and unstructured after years of intensive use. Only users with extensive subject area knowledge, system knowledge, and classification scheme knowledge are able to maneuver and explore in these textual databases (Chen & Dhar, 1990).

Most commercial information retrieval systems still rely on conventional inverted index and Boolean querying techniques. Even full-text retrieval has produced less than satisfactory results (Blair & Maron, 1985). Probabilistic retrieval techniques have been used to improve the retrieval performance of information retrieval systems (Bookstein & Swanson, 1975; Maron & Kuhns, 1960). Despite various extensions, probabilistic methodology still requires the *independence assumption* for terms and it suffers from difficulty of estimating term-occurrence parameters correctly (Gordon, 1988; Salton, 1989).

Since the late 1980s, knowledge-based techniques have been used extensively by information science researchers. These techniques have attempted to capture searchers' and information specialists' domain knowledge and classification scheme knowledge, effective search strategies, and query refinement heuristics in document retrieval systems design (Chen & Dhar, 1991). Despite their usefulness, systems of this type are considered *performance systems* (Simon, 1991)—they only *perform* what they were programmed to do (i.e., they are without *learning* ability). Significant efforts are often required to acquire knowledge from domain experts and to maintain and update the knowledge base.

A newer paradigm, generally considered to be the *machine learning* approach, has attracted attention of researchers in artificial intelligence, computer science, and other functional disciplines such as engineering, medicine, and business (Carbonell, Michalski, & Mitchell, 1983; Michalski, 1983; Weiss & Kulikowski, 1991). In contrast to *performance systems* which acquire knowledge from human experts, *machine learning systems* acquire knowledge automatically from examples, i.e., from source data. The most frequently used techniques include symbolic, inductive learning algorithms such as ID3 (Quinlan, 1979), multiple-layered, feed-forward neural networks such as Backpropagation networks (Rumelhart, Hinton, & Williams, 1986), evolution-based genetic algorithms (Goldberg, 1989), and the physics-based simulated annealing (van Laarhoven & Aarts, 1988). Many information science researchers have started to experiment with these techniques as well (Belew, 1989; Chen & Lynch, 1992; Chen, Lynch, Basu, & Ng, 1993; Gordon, 1988; Kwok, 1989).

In this article, we aim to examine the prevailing machine learning (and search) techniques and their implementations in information retrieval. In Section 2, we review the probabilistic techniques and the emerging machine learning methods. We then summarize some recent work adopting such techniques in information retrieval (IR). After the overview, in Section 3, we present the conventional relevance feedback method, ID3, genetic algorithms, and simulated annealing for performing ''Inductive Query by Examples'' (IQBE)—a process in which searchers provide sample documents (examples) and the algorithms ''induce'' (or ''learn'') the key concepts (represented as terms or keywords) in the documents in order to find other relevant documents. Preliminary testing results using an 8,000-record database based on a popular similarity function (Jaccard's score), and results of a user evaluation experiment are also provided in Section 4. We present conclusions and planned future research directions in Section 5.

2. Information Retrieval Using Probabilistic and Machine Learning Techniques

In classical information retrieval models, relevance feedback, document space modification, and probabilistic models are among the techniques most relevant to our research. In this section, we will first briefly summarize relevant work in these areas. However, our main purpose will be to present research in machine learning for information retrieval. Similarities and differences among techniques will be discussed.

2.1. Relevance Feedback and Probabilistic Models in IR

One of the most important and difficult operations in information retrieval is to generate queries that can succinctly identify relevant documents and reject irrelevant documents. Since it is often difficult to accomplish a successful search on the initial try, it is customary to conduct searches iteratively and to reformulate query statements based on evaluation of the previously retrieved documents. One method for automatically generating improved query formulations is the well-known and effective *relevance-feedback* process (Ide, 1971; Ide & Salton, 1971; Rocchio, 1971; Salton, 1989). A query can be improved iteratively by taking an available query vector (of terms) and adding terms from the relevant documents, while subtracting terms from the irrelevant documents. A single iteration of relevance feedback usually produces improvements of from 40 to 60% in search precision (Salton, 1989).

A similar approach can also be used to alter the document representation. *Document-vector modification* changes and improves document indexes, based on user relevance feedback of relevant and irrelevant documents (Brauen, 1971). Using such a technique, the vectors of documents previously retrieved in response to a given query are modified by moving relevant documents closer to the query, and at the same time moving irrelevant documents away from the query. While the relevance feedback procedure is efficient and intuitively appealing, it does not attempt to analyze characteristics associated with the relevant and irrelevant documents in order to ''infer'' what concepts (terms) are most appropriate for representing a given query (or queries).

In probabilistic information retrieval, the goal is to estimate the *probability of relevance* to a user of a given document with respect to a given query. Probabilistic assumptions about the distribution of elements in the representations within relevant and irrelevant documents are required. Using relevance feedback from a few documents, the model can be applied in order to estimate the probability of relevance for the remaining documents in a collection (Fuhr & Buckley, 1991; Fuhr & Pfeifer, 1994; Gordon, 1988). In order to simplify computation, an assumption is usually made that terms are distributed independently (Maron & Kuhns, 1960). Fuhr and his coworkers discussed probabilistic models as an application of machine learning (Fuhr & Buckley, 1991; Fuhr & Pfeifer, 1994).

Although relevance feedback and probabilistic models exhibit interesting query or document refinement capabilities, their abstraction processes are based on either simple addition/removal of terms, or probabilistic assumptions and principles. Their learning behaviors are significantly different from those developed in the machine learning areas, especially neural networks, symbolic learning, genetic algorithms, and simulated annealing.

2.2. Learning Systems for IR

Chen (1995) provides a good survey of neural networks, symbolic learning, and genetic algorithms adopted for information retrieval. In this research, symbolic learning (ID3), genetic algorithms, and simulated annealing were adopted.

In Blosseville, Hebrail, Monteil, and Penot (1992), the researchers used discriminant analysis and a simple symbolic learning technique for automatic text classification. Their symbolic learning process represented the numeric classification results in terms of IF-THEN rules. Fuhr et al. (1990) adopted regression methods and ID3 for their feature-based automatic indexing technique. Crawford, Fung, and their coworkers (Crawford, Fung, Appelbaum, & Tong, 1991; Crawford & Fung, 1992; Fung & Crawford, 1990) have developed a probabilistic induction technique called CONSTRUCTOR and have compared it with the popular CART algorithm (Breiman, Friedman, Olshen, & Stone, 1984). Their experiment showed that CONSTRUCTOR's output is more interpretable than that produced by CART, but CART can be applied to more situations (e.g., real-valued training sets). In 1994, Chen and She adopted ID3 and the incremental ID5R algorithm for information retrieval. Both algorithms were able to use user-supplied samples of desired documents to construct decision trees of important keywords which could represent the users' queries.

Our literature search revealed several recent implementations of genetic algorithms in information retrieval. In 1988, Gordon presented a genetic algorithms-based approach for document indexing. Competing document descriptions (keywords) were associated with a document, and altered over time by using genetic mutation and crossover operators. In 1991, Gordon adopted a similar approach to document clustering. Yang and his coworkers (Yang, Korfhage, & Rasmussen, 1993) have developed adaptive retrieval methods based on genetic algorithms and the vector space model using relevance feedback. They reported the effect of adopting genetic algorithms in large databases, the impact of genetic operators, and GA's parallel searching capability. In 1994–1995, Chen and Kim reported a GA-neural-network hybrid system for IR, called GANNET. The system performed *concept optimization* for user-selected documents using genetic algorithms. It then used the optimized concepts to perform *concept exploration* in a large network of related concepts through the Hopfield net parallel relaxation procedure. A Jaccard's score was also adopted to compute the ''fitness'' of subject descriptions for information retrieval.

Despite an extensive literature search, we found no application of simulated annealing for IR. However, as will be described below, the similarity between simulated annealing and genetic algorithms prompted us to incorporate this method in our experiment involving learning algorithms for IR. The diverse philosophy of learning (or search) demonstrated in simulated annealing made it an interesting candidate for our experiment.

3. Relevance Feedback, ID3, Genetic Algorithms, and Simulated Annealing for IR

In this section, we summarize the various algorithms and the knowledge representations and adaptations we developed in the context of IR.

3.1. Relevance Feedback for IR

One proven method for automatically generating improved query statements is the well-known relevance-feedback process (Ide, 1971; Rocchio, 1971). The main assumption behind relevance feedback (RF) is that documents relevant to a particular query resemble each other in the sense that they are represented by reasonably similar vectors of keywords or descriptors (Salton, 1989). This implies that if a retrieved document has been identified as relevant to a given query, the query formulation can be improved by increasing its similarity to such a previously retrieved relevant document. The reformulated query is expected to retrieve additional relevant documents that are similar to the originally identified relevant document. A sketch of the relevance feedback procedure adopted in our research follows:

(1) **Select initial relevant documents:** Using any traditional search options (e.g., Boolean, keyword), searchers present a query statement, Q^t, to search an initial set of documents.
(2) **Relevance feedback:** By browsing these documents, searchers can select documents deemed relevant to their queries. In this research, we only considered ''positive'' documents, i.e., documents selected as relevant; documents deemed irrelevant were not considered in the query refinement process. (See Salton, 1989, for other forms of relevance feedback.)
(3) **Query refinement and search:** Keywords derived from the relevant documents are then added to Q^t to form a refined query, Q^{t+1}. A new search is then performed to identify other new documents. Steps 2 and 3 are repeated until a searcher decides to stop.

Although the relevance feedback computation is simple (i.e., merely adding or deleting keywords to/from queries), it has been shown to significantly improve both search recall and precision (Salton, 1989). This intuitively appealing and simple method of query refinement was taken as the benchmark for comparison with other more sophisticated machine learning algorithms that aimed to ''learn'' from the common characteristics of the relevant documents. In addition to examining the computational characteristics and the performances of the selected algorithms, we also examined the impact of sample sizes, i.e., number of documents used for analysis. Although relevance feedback operated nicely for different sample sizes, many learning and search algorithms such as ID3 and genetic algorithms needed to have a large sample size in order to perform adequately. More details will be discussed below.

3.2. *Symbolic Learning (ID3) for IR*

Among the various symbolic learning algorithms developed over the past decade, ID3 and its variants have been tested extensively and shown to rival other machine learning techniques in predictive power (Mooney, Shavlik, Towell, & Gove, 1989; Weiss & Kulikowski, 1991). ID3 is a decision-tree building algorithm developed by Quinlan (1979, 1983). It adopts a divide-and-conquer strategy and the entropy measure for object classification. Its goal is to classify mixed objects into their associated classes based the objects' attribute values.

In IR, we can assume that there exists a database (universe) of documents (or records). Documents are described by attributes (keywords, primary keys, fields). Each document in the database then belongs to only one of two possible classes:

- The ''positive'' class (+): Consisting of documents that are desired; and
- the ''negative'' class (−): Consisting of documents that are undesired.

In our implementation, we maintained a list of all the keywords that existed in the desired documents and used this list to decide what attributes were crucial to describing documents in the positive class. The test at each non-leaf node of the decision tree determined the presence or absence of a particular keyword: ''Yes'' meant that the test keyword existed in a document, and ''no'' meant that the keyword did not exist in a document. Thus, ID3 created a binary query (concept) tree. A sketch of the ID3 algorithm adopted follows:

(1) **Compute entropy for mixed classes:** Initially searchers were requested to provide a set of positive and negative documents (providing negative documents was optional). This set of documents served as the training examples for the ID3 algorithm. Entropy was calculated by using the *entropy* function (Quinlan, 1983):

$$entropy = -p_{pos}\log p_{pos} - p_{neg}\log p_{neg}$$

where p_{pos} and p_{neg} represented the proportions of the documents which were positive or negative, respectively.

(2) **Select the best attribute based on entropy reduction:** For each untested attribute (keyword), the algorithm computed an entropy value for its use when classifying mixed documents. A new set of positive and a new set of negative documents were generated based on each attribute (keyword) considered and the entropy value computed. Each branch of the decision tree represented the existence or non-existence of a particular keyword. The keyword which reduced the entropy most served as the next decision node in the tree. As a ''greedy'' algorithm, ID3 always aims at maximizing local entropy reduction and never backtracks.

(3) **Iterate until all documents are classified:** Repeating Steps 1 and 2, ID3 computed the entropy value of each mixed class and identified the best attribute for further classifying the class. The process was continued until each class contained either all positive or all negative documents.

Previous research has shown that the ID3 tree-building process requires much less computation than other inductive learning methods, including neural networks and genetic algorithms. However, as when many other learning algorithms are used, clean and large sample sizes are required for good classification results.

3.3. *Genetic Algorithms for IR*

Genetic algorithms (GAs) (Goldberg, 1989; Koza, 1992; Michalewicz, 1992) are problem solving systems based on principles of evolution and heredity. Their use is often compared with that of neural networks and the symbolic learning methods, and their self-adaptiveness property is extremely appealing for IR applications.

Genetic algorithms use a vocabulary borrowed from natural genetics in that they talk about *genes* (or bits), chromosomes (individuals or bit strings), and population (of individuals). Populations evolve through generations. Our genetic algorithm for IR was executed in the following steps:

(1) **Initialize population and evaluate fitness:** When adopting GAs in IR, each gene (bit) in the chromosome (bit string) represented a certain keyword or concept. The loci (locations of a certain gene) decided the existence (1, ON) or nonexistence (0, OFF) of a concept. A chromosome therefore represented a document which consisted of multiple concepts.

An evaluation function for the *fitness* of each chromosome was selected based on the Jaccard's score [a popular similarity function used in IR (Rasmussen, 1992; Salton, 1989; Van Rijsbergen, 1979)] as used by Gordon for document indexing (Gordon, 1988). The Jaccard's score between two sets, X and Y, was computed as:

$$\#(X \cap Y)/\#(X \cup Y)$$

where $\#(S)$ indicated the cardinality of set S. It was

```
ANALYSIS OF VARIANCE
SOURCE     DF        SS        MS         F        p
FACTOR      8    1.7455    0.2182    16.52    0.000
ERROR     558    7.3718    0.0132
TOTAL     566    9.1173
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
 LEVEL      N      MEAN     STDEV  ------+---------+---------+---------+
J0         63    0.0604    0.0616   (---*---)
J1-RF      63    0.1443    0.0987               (---*---)
J2-RF      63    0.2287    0.1569                           (---*---)
J1-ID3     63    0.1195    0.1007           (---*---)
J2-ID3     63    0.1577    0.0510                (----*---)
J1-GA      63    0.1454    0.0926               (---*---)
J2-GA      63    0.2361    0.1735                             (---*---)
J1-SA      63    0.1458    0.0878               (---*---)
J2-SA      63    0.2312    0.1471                           (---*---)
                                   ------+---------+---------+---------+
POOLED STDEV =   0.1149                0.070     0.140     0.210     0.280
```

FIG. 1. Results for all test cases.

used in our research to indicate the degree of similarity between the system-suggested set of documents and the initial searcher-identified set of relevant documents. A high Jaccard's score was desirable.

(2) **Reproduction (selection):** A roulette wheel with slots (F) sized according to the total fitness of the population was defined as follows:

$$F = \sum_{i=1}^{popsize} fitness(V_i)$$

where $fitness(V_i)$ indicated the fitness value of chromosome V_i according to the Jaccard's score.

Each chromosome had a certain number of slots proportional to its fitness value. The selection process was based on spinning the wheel *popsize* (population size) times; each time we selected a single chromosome for a new population.

(3) **Recombination (crossover and mutation):** We then adopted the crossover and mutation operators to generate new chromosomes for the new generation.

(4) **Convergence:** Following reproduction, crossover, and mutation, the new population was ready for its next generation. The rest of the evolutions were simply cyclic repetitions of the above steps until the system reached a predetermined number of generations.

In summary, the initial population contained a set of documents which were judged relevant by a searcher. The goal of a GA was to find an optimal set of documents which best matched the searcher's needs (expressed in terms of underlying keywords or concepts). This process of evolving toward the better chromosomes (documents) based on the Jaccard's score of fitness was clearly more time-consuming than relevance feedback and ID3. The effect of the initial population size, i.e., the number of documents provided by a searcher, had not been made clear from earlier research (Gordon, 1988).

3.4. Simulated Annealing for IR

In its original form, the simulated annealing (SA) algorithm is based on an analogy between the simulation of the annealing of solids and the problem of solving large combinatorial optimization problems (van Laarhoven & Aarts, 1988). Despite their different philosophical paradigms, SA implementation showed substantial resemblance to that of genetic algorithms (Michalewicz, 1992). In the context of search, SA has been shown to be an excellent candidate for solving large-scale combinatorial optimization problems. SA adopted in this research can be described as follows.

(1) **Initial configuration:** We first initialized a finite configuration space (or solution space) S and then assigned a cost function C, which was a real number, to each configuration. Similar to the initial population in genetic algorithms, the initial simulated annealing configuration space contained a set of documents that were judged relevant by a searcher and was represented as a vector space of index terms. The cost function C allowed us to compute the cost of the proposed solution. The algorithm was initialized with a high value for the control parameter **c** and a candidate solution (configuration) i. A generation mechanism allowed us to generate a configuration j given the configuration i.

(2) **Cooling and generating new configurations:** The value of the control parameter c was gradually decreased according to a *cooling schedule* as the algorithm was executed. At each value of the control parameter, a sequence of configurations was generated and for each such configuration, the cost of the

```
ANALYSIS OF VARIANCE
SOURCE     DF        SS        MS        F        p
FACTOR      8    1.6200    0.2025    15.50    0.000
ERROR     306    3.9975    0.0131
TOTAL     314    5.6175
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
 LEVEL      N      MEAN     STDEV  -----+---------+---------+---------+-
J0         35    0.0925    0.0645  (---*---)
J1-RF      35    0.2183    0.0647               (---*---)
J2-RF      35    0.3123    0.1602                        (---*---)
J1-ID3     35    0.1754    0.1024           (---*--)
J2-ID3     35    0.1649    0.0583          (--*---)
J1-GA      35    0.2084    0.0756              (---*---)
J2-GA      35    0.3254    0.1806                          (---*--)
J1-SA      35    0.2071    0.0596              (---*---)
J2-SA      35    0.2914    0.1667                      (---*---)
                                   -----+---------+---------+---------+-
POOLED STDEV =   0.1143                0.10      0.20      0.30      0.40
```

FIG. 2. Results for SMALL test cases.

solution was computed. For a sequence of configurations i and j, let δC_{ij} be defined as

$$\delta C_{ij} = C_j - C_i$$

Configuration j was accepted as the new configuration according to the following probability.

$$P(Acceptance) = Min\left(1, exp\left(\frac{-\delta C_{ij}}{c}\right)\right)$$

Thus, cost decreasing transitions were always accepted while cost increasing ones might or might not be, depending on the value of δC_{ij} and c. Formally, this statistical cooling process was modeled by a sequence of homogeneous Markov chains of finite length at decreasing values of the control parameter.

(3) **Termination:** The algorithm was terminated at some small value of c and the final configuration was taken as the solution to the problem at hand.

In our implementation, we adopted the GA representation for the configuration of SA. Thus, the input set of documents was translated into a bit string representation as described earlier. To generate a new candidate configuration, a random number generator was used to determine a ''mutation'' point in each of the bit strings representing a document. The Jaccard's score described earlier was used as the cost function. If the candidate configuration showed an increase in the Jaccard's score (when compared with the initial population), the configuration was accepted for the next transition. If the Jaccard's score decreased, a random number was generated and compared with the acceptance probability to determine whether the transition was to be accepted or not.

4. System Evaluation

The relevance feedback (RF), ID3, genetic algorithm (GA), and simulated annealing (SA) algorithms were developed in ANSI C and were run on a DEC Alpha 3000 (ULTRIX, 64-bit machine) workstation. In order to examine the computational characteristics of these algorithms and their abilities to ''learn'' from sample documents and help identify documents relevant to some initial queries, we created a database of about 8,000 records by extracting recent software and system engineering related documents from the publicly available document collection of the *EiCompendex*Plus*^{TM} (COMPEN) database.[1] We believe the size of our extracted database posed a challenge to our algorithms for analyzing and suggesting relevant documents.

Our system evaluation consisted of two phases: A benchmark testing experiment using 63 predetermined test cases of varying sizes, and a user evaluation experiment using human subjects and 36 actual queries. We were able to determine the computational characteristics and retrieval performances of the proposed algorithms.

4.1. A Benchmark Experiment

4.1.1. Experimental setting. In our benchmark experiment, the experimenters created two sets of test cases: (1) The SMALL set containing seven cases of 1-document, 2-document, 3-document, 4-document, 5-document, and 10-document examples (a total of 35 test cases); and (2) the LARGE set containing seven cases

[1] COMPEN is the machine-readable version of the Engineering Index, which provides abstracted information from the world's significant literature of engineering and technology (approximately 4,500 journals and selected government reports and books).

```
ANALYSIS OF VARIANCE
SOURCE      DF        SS         MS         F        p
FACTOR       8   0.53320    0.06665     30.02    0.000
ERROR      243   0.53942    0.00222
TOTAL      251   1.07262
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
 LEVEL       N      MEAN      STDEV  ---------+---------+---------+------
J0          28   0.02029    0.02158   (--*---)
J1-RF       28   0.05193    0.03397        (--*---)
J2-RF       28   0.12414    0.06183                     (---*--)
J1-ID3      28   0.04953    0.03050       (---*--)
J2-ID3      28   0.14856    0.03937                          (---*--)
J1-GA       28   0.06668    0.03002           (--*---)
J2-GA       28   0.12447    0.06917                     (---*--)
J1-SA       28   0.06921    0.04788           (---*--)
J2-SA       28   0.15586    0.06387                            (--*---)
                                   ---------+---------+---------+------
POOLED STDEV =   0.04712                 0.050     0.100     0.150
```

FIG. 3. Results for LARGE test cases.

of 20-document, 30-document, 40-document, and 50-document examples (a total of 28 test cases). Each test case included documents on varying topics and degrees of similarity. In addition to the general behaviors of the algorithms, we also examined the effects of sample size on performance. Each document contained different numbers of index terms previously assigned by the COMPEN database. In our experiments, these documents were represented as a vector space of their assigned index terms. Based on the vector space model, the four algorithms added and removed index terms during the inductive learning process. The systems searched only for COMPEN documents that contained index terms in the sample documents, a process which was efficient based on the inverted index we created for the COMPEN database.

The Jaccard's score was adopted as a measure of performance for the retrieved documents. Higher Jaccard's scores indicated stronger similarity and/or higher degrees of association (with some target documents) and were more desirable. For each test case of initial sample documents an average Jaccard's score (J0) was computed. Each document in a test case was first compared with all other documents in the same test case to obtain an (average) individual Jaccard's similarity score. J0 for the entire test case was computed by averaging the individual Jaccard's scores of all documents in the same test case. Thus, J0 can be considered a self-similarity score for the initial sample documents, i.e., indicated how similar to each other they are. After each algorithm had generated relevant keywords based on the example, those keywords were used to identify the 10 most relevant documents from the COMPEN database based on the index terms and weights. A new Jaccard's score (J1) between these 10 new documents and the initial set of sample documents was then computed to determine the similarity score between the algorithm's suggestions and the initial documents. Each new document was compared with all documents in the initial set. J1 could be considered the between-group similarity score. A third Jaccard's score (J2) was also computed to decide the self-similarity among the documents in the newly recommended set of 10 documents, a process similar to the J0 self-similarity computation.

A robust algorithm should suggest documents which are similar to the initial set of documents (i.e., high J1) and the set of recommended documents should also reveal a consistent theme or topic (i.e., high J2). In our evaluations, we computed J0, J1, and J2 for different methods, and compared the newer and more sophisticated learning algorithms against the conventional and proven method of relevance feedback.

4.1.2. Experimental results. We first discuss the general characteristic of each method, followed by a comparison and analysis of performance among the four algorithms.

Relevance feedback (RF) allocated weights based on the occurrence of keywords in the input. For example, in a five-document input set, if a keyword appeared in three documents, it was assigned a weight of 0.6 (3/5). Weights for other keywords were computed in a similar fashion. Unlike GA or SA, RF did not attempt any concept optimization. Weights for keywords were used only as a guide for retrieving documents. Each keyword had a positive weight, which contributed to and influenced the search. Typically, no one keyword (or set of keywords) dominated others in terms of weight, making it unlikely that any one keyword would dominate the output. The RF computation was simple and extremely efficient.

Using ID3, a query tree was generated for each test case, with each branch representing a keyword. For the SMALL set of documents, the tree sizes were usually between 1 and 2 levels, i.e., 1–3 keywords were identified

TABLE 1. CPU times (in seconds) for different algorithms.

No. of documents	2	3	4	5	10	20	30	40	50
RF	0.0000	0.0000	0.0000	0.0000	0.0000	0.0060	0.0072	0.0078	0.0119
ID3	0.1429	0.1429	0.1429	0.1429	0.2857	0.2857	0.5714	0.5714	1.2857
GA	0.0143	0.0000	0.0143	0.1000	0.2714	2.6714	7.8286	18.9571	35.2429
SA	0.0790	0.4012	0.8826	2.5034	44.6424	288.0240	1009.8058	1307.8086	1508.1014

as crucial. As document size increased, the tree levels increased. For 20-document cases, the resulting trees had between 4 and 7 keywords and for 50-document cases, the resulting trees had between 20 and 35 keywords. The computation requirement increased significantly when the sample sizes were increased.

In our implementation of the genetic algorithm, we chose 40 as the number of iterations, based on our experimentation. For both the SMALL and LARGE test sets, the algorithm showed no significant improvement after 40 iterations. P_c and P_m were set at 0.8 and 0.02, respectively. After evolution, a few of the keywords could be determined as most crucial. (Not all common keywords were retained after the evolution.) In selecting a set of 10 documents from the database for the initial set of documents, the algorithm attempted to match as many optimized keywords as possible. As a result, the algorithm retrieved documents that, as a set, had a large number of keywords in common. The 40-generation evolution process was computationally expensive when compared with RF and ID3.

SA output had some distinct characteristics. As the algorithm progressed, the probability of accepting bad transitions (i.e., decreasing Jaccard's scores) became smaller and smaller. In the initial stages, some bits were mutated even though the corresponding Jaccard's score (J1) had decreased. Later, however, only the bits which increased the Jaccard's score were mutated. As a result, bits corresponding to keywords which appeared frequently in the input had high weights in the final population. The SA-suggested search gave higher priority to those documents in the database which contained more of these ''high-weight'' keywords. Other keywords which had moderate weights were not able to influence the search results. Thus, SA had the effect of zeroing in on keywords which appeared to have more importance as judged by the input documents. The process of ''exploring'' different configurations based on the acceptance probability was also extremely time-consuming.

- *Results for all test cases:* As shown in Figure 1 [one-way analysis of variance, using MINITAB (Ryan, Joiner, & Ryan, 1985)], for all 63 test cases, J1 and J2 improved over the initial J0 for all algorithms, at the 5% significance level. By analyzing the keywords associated with the selected documents, all algorithms appeared to have positively improved the similarity scores. However, except for ID3, which performed poorly for both J1 and J2, performances of the other three algorithms were not significantly different from RF at J1 or J2. That is, the more sophisticated ID3, GA, and SA did not out-perform the simpler RF algorithm.
- *Results for SMALL test cases:* A further analysis of the 35 cases of smaller sizes (2, 3, 4, 5, 10 documents) led to a similar conclusion (see Fig. 2). ID3 performed very poorly when the document sample sizes were small, especially for J2. All other algorithms performed better than ID3 in J2 (significance level, $p = 0\%$). When sample sizes were small, ID3 may have created query trees which were overly simplistic—the higher-level nodes in the trees often dominated the search results.
- *Results for LARGE test cases:* A detailed comparison of J1 and J2 for the remaining 28 test cases of larger document sizes revealed interesting results (see Fig. 3). J1 comparisons showed that GA and SA out-performed the benchmark RF method at $p = 9.1\%$ and $p = 13\%$ (slightly above the 10% statistical significance level), respectively (using two sample *t*-test on MINITAB). In J2 comparisons, SA out-performed the benchmark RF method at $p = 6.5\%$ and ID3 out-performed RF at $p = 8.5\%$. Overall, ID3 and SA performed well in J2 (as compared to RF) when the sample sizes were large. GA and SA, on the other hand, performed well for J1 (when compared with RF) when the sample sizes were large. It appeared that only when the sample sizes were large (i.e., when the queries were complex, fuzzy, and unfocused) did the more sophisticated algorithms demonstrate ability to converge on more focused and similar keywords, and thus find more similar document sets. These results may have significant implications for designing adaptive IR systems to assist in complex, large-scale IR sessions.

Table 1 summarizes the average CPU times (in seconds) for all (SMALL and LARGE) test cases of varying document sizes on a DEC Alpha 3000 workstation. Clearly, initial sample sizes played an important role in deciding the CPU times required of each algorithm. SA demanded significantly longer CPU times than other methods.

4.2. A User Evaluation Experiment

In an attempt to examine the performances of the proposed machine learning techniques for actual retrieval sessions, we designed a user evaluation experiment involving human subjects who retrieved documents of interest to them from the 8,000-record COMPEN database.

```
A. RECALL COMPARISON (ONE-WAY ANALYSIS OF VARIANCE):

ANALYSIS OF VARIANCE
SOURCE      DF        SS        MS         F        p
FACTOR       3    1.4364    0.4788     19.70    0.000
ERROR      140    3.4033    0.0243
TOTAL      143    4.8397
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
 LEVEL       N      MEAN     STDEV  -+---------+---------+---------+-----
RF          36    0.4817    0.1818               (----*----)
ID3         36    0.3416    0.1678  (----*----)
GA          36    0.5993    0.1094                           (----*----)
SA          36    0.5681    0.1551                        (----*----)
                                    -+---------+---------+---------+-----
POOLED STDEV =    0.1559           0.30      0.40      0.50      0.60

B. RECALL COMPARISON (TWO-SAMPLE T-TEST VS. RF):

TTEST MU ID3 = MU RF (VS NE): T= -3.40  P=0.0011  DF=  69
TTEST MU GA = MU RF (VS NE): T= 3.32  P=0.0016  DF=  57
TTEST MU SA = MU RF (VS NE): T= 2.17  P=0.034  DF=  68
```

FIG. 4. Recall comparison for actual queries.

Standard IR performance measures of document *recall* and *precision* were used in this experiment.

4.2.1. Experimental setting. A simple character-based, menu-driven interface operated by the researchers was developed in ANSI C, in addition to UNIX scripts for invoking the different algorithms. All experiments were performed on a DEC Alpha 3000 workstation. Twenty-one graduate students in the Information Systems, and Systems and Industrial Engineering Departments, were solicited as subjects. These subjects were asked first to present a few queries in the areas of software and system engineering. Many queries were intended to find citations and abstracts from the 8,000-record COMPEN database for end-of-semester class projects. Thirty-six queries were performed using the interface. (Several subjects performed more than one query session.)

Due to difficulty in operationalizing the recall and precision measures (especially recall) (Salton, Allan, & Buckley, 1994), we adopted a document evaluation design similar to the one reported in (Ekmekcioglu, Robertson, and Willet (1992). Subjects were asked to examine different sets of ranked documents for their relevance to the corresponding query.

Each subject's query was represented as a vector space of search terms. The researchers then used these terms to identify an initial set of documents. These documents were examined by the subjects, who make selections from them. The resulting set of user-selected documents was represented as a vector space of index terms and was used as the ''examples'' for the four algorithms. During the inductive query learning process, each algorithm identified a set of ranked terms (concepts) to represent the examples, and then used these terms to retrieve the 10 highest-ranked documents from the database. In addition, we also included a set of 10 ''noise'' documents that did not contain any keywords in the initial set of user-selected documents. After the inductive query learning process, the suggested documents (50 at most, but often less, after removing duplicates) were mixed and presented in random order to the subjects for further selection. Documents which were judged as relevant by the subjects were then used as examples in the next round of inductive query learning and relevance evaluation. The process was terminated when the subjects decided to stop, often after two to three rounds of iteration (when the system ran out of new suggestions). Each query session lasted between 15 and 30 minutes. Most subjects were able to find relevant documents during the IQBE process.

4.2.2. Experimental results. Among the 36 queries conducted by the subjects, none of the documents from the ''noise'' sets were selected as relevant. This somewhat confirmed the ''quality'' of their evaluations.

In this subsection, we report the recall and precision comparison of the four algorithms. Our main objective was to compare the performances of the newer inductive learning algorithms against that of the conventional relevance feedback method.

- *Results of recall comparison: Document recall* indicated the portion of the target documents (selected by the subjects after the complete search process) which was found in each of the four sets of documents suggested by the four algorithms, respectively. As shown in Figure 4, RF obtained a 48.17% recall level, which was significantly (at 10% confidence level) superior to the 34.16% recall level achieved by ID3, but signifi-

```
A. PRECISION COMPARISON (ONE-WAY ANALYSIS OF VARIANCE):

ANALYSIS OF VARIANCE
SOURCE      DF        SS         MS         F        p
FACTOR       3    0.1549     0.0516      2.11    0.102
ERROR      140    3.4322     0.0245
TOTAL      143    3.5871
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
 LEVEL       N      MEAN      STDEV  ---------+---------+---------+-------
RF          36    0.2418     0.1281   (-------*--------)
ID3         36    0.3060     0.1984            (--------*--------)
GA          36    0.3318     0.1204                 (-------*--------)
SA          36    0.2959     0.1667           (-------*--------)
                                     ---------+---------+---------+-------
POOLED STDEV =    0.1566                    0.240     0.300     0.360

B. PRECISION COMPARISON (TWO-SAMPLE T-TEST VS. RF):

TTEST MU ID3 = MU RF (VS NE): T= 1.63  P=0.11  DF=  59
TTEST MU GA = MU RF (VS NE): T= 3.07  P=0.0030  DF=  69
TTEST MU SA = MU RF (VS NE): T= 1.54  P=0.13  DF=  65
```

FIG. 5. Precision comparison for actual queries.

cantly inferior to 59.93 and 56.81% achieved by GA and SA, respectively. GA, in particular, was most impressive in its ability to find relevant documents for the subjects given the amount of computation required (SA was significantly slower than all other methods). ID3, on the other hand, seemed to over-generalize the query concepts and obtained the worst results. (More discussion will be provided below.)

- *Results of precision comparison: Document precision* indicated the portion of each system-suggested document list that appeared in the target document list. As shown in Figure 5, all three inductive learning methods out-performed RF in precision. Again, GA performed most impressively of all four methods in precision.

A close examination of the characteristics of the terms induced by the four algorithms helped depict the behaviors of the algorithms and explain the user evaluation experiment results. For selected query sessions, we plotted the terms suggested by each algorithm and their associated weights. A sample session is shown in Figure 6, where the x-axis indicates the 21 terms which appeared in the five example documents and the y-axis indicates their weights on a scale between 0 and 1. GA, SA, and RF generated a probabilistic weight for each term after their algorithmic computations (a natural by-product of these algorithms). For ID3, terms higher on the query tree were assigned greater probabilities than terms lower on the tree using a monotonically decreasing scale (1.0, 0.9, 0.8, and so on, for our implementation).

Several important observations could be made based on this graphical analysis. In all cases, RF produced all the terms which appeared in the initial documents and simply accumulated weights for terms which appeared in more than one document (for example, the three spikes shown in Fig. 6 represented terms 7, 16, and 17). The algorithm made no attempt to induce a smaller set of crucial terms (concepts) to represent the document set. The other three inductive learning algorithms, on the other hand, filtered out insignificant terms during the learning process (as is evident from the missing terms in Fig. 6, e.g., GA did not suggest terms 16–21).

Overall, we found that GA and SA removed about one-third to one-half of the terms that were deemed insignificant, and weighted selected terms more heavily than others based on its fitness function computation (for example, term 7 was weighted much higher than other terms by both GA and SA). However, ID3 appeared to over-generalize (or over-simplify) the query concepts by creating a query tree which contained only limited levels (and terms). This inductive learning process caused the adverse effect of zooming in on only a few selected terms and ignoring other potentially useful concepts (for example, in Fig. 6, only three terms were determined relevant by ID3).

In summary, we believe that the RF method suffered in recall and precision because it did not identify the most crucial concepts (keywords/terms) to represent the example documents. ID3, on the other hand, performed poorly in recall due to over-generalization. GA and SA appeared to strike a good balance between induction and generalization. As a result, crucial concepts were identified from the example documents and used to find more relevant documents, as was made evident by our user-evaluation experiment.

5. Conclusions and Future Directions

As the amount of data and information continues to grow at an exponential pace due to the rapid proliferation

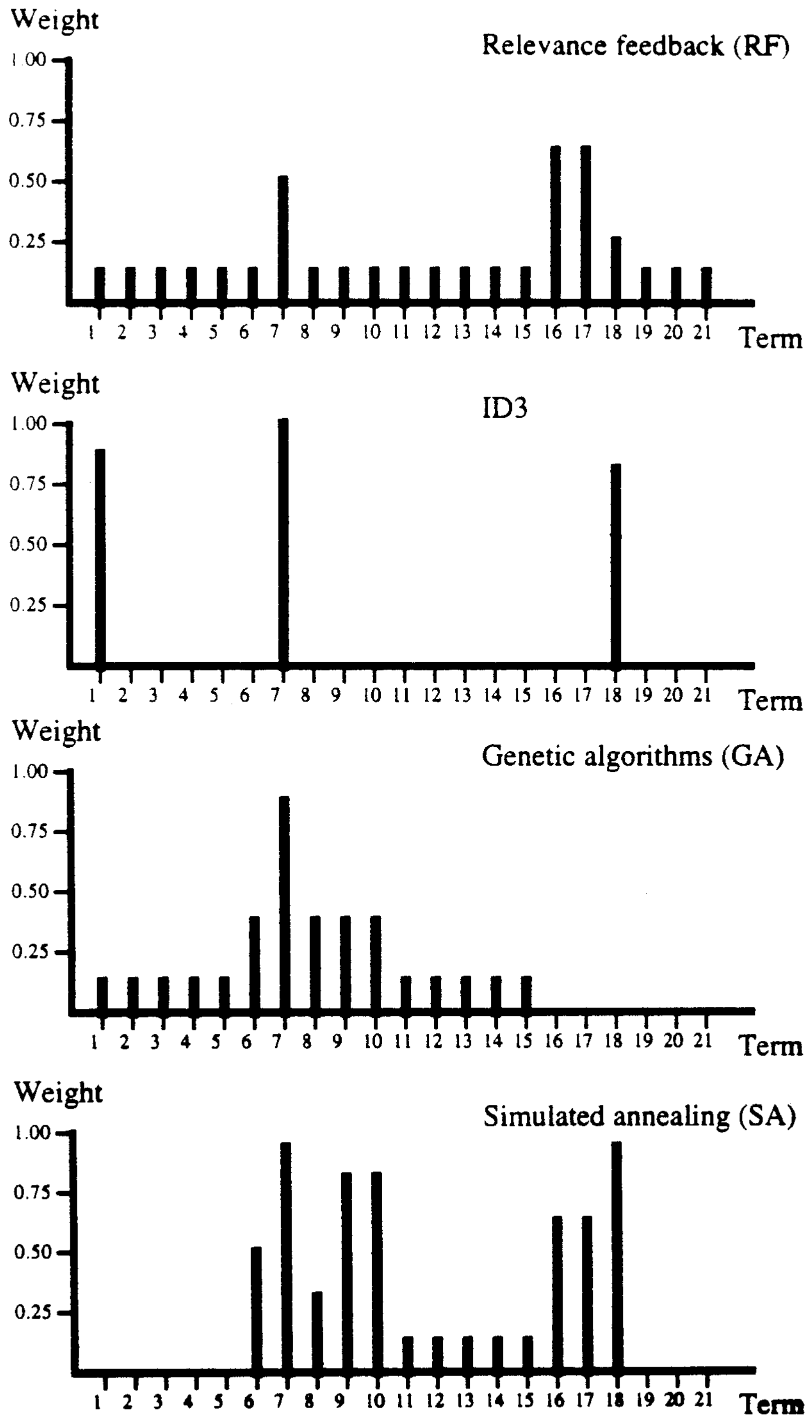

FIG. 6. Sample terms and associated weights suggested by RF, ID3, GA, and SA.

of large-scale digital libraries, database management systems, and Internet servers, the problem of finding relevant documents effectively and efficiently becomes even more pressing. Researchers have experimented with techniques ranging from probabilistic models and the vector space model to the recent knowledge-based and machine learning techniques.

In this article, we present results of the first phase of an ''intelligent'' IR research project that was mainly based on machine learning techniques, including the symbolic ID3, genetic algorithms, and simulated annealing. These algorithms were compared with the conventional and popular relevance feedback method for query refinement. Our benchmark experiment revealed that ID3 and SA performed very well (when compared with RF) in generating a set of similar documents (i.e., high J2) when the sample sizes were large. GA and SA, on the other hand, performed well for J1 (i.e., in identifying a set of documents similar to the initial documents) when compared with RF. In our user-evaluation experiment, we found that GA and SA out-performed RF in both recall and precision. ID3, on the other hand, performed poorly in recall. Considering the efficiency and simplicity of GA, and its performances in both experiments, we favor GA as the most promising machine learning algorithm for inductive query by examples.

Given the highly subjective and stringent nature of the evaluations done by our experimental users, based on their own relevance criteria, the values of precision and recall computed here can be considered acceptable, especially compared with the benchmark, traditional relevance feedback approach. It is also worthwhile to note that precision and recall are conflicting measures. Pursuing one would be at the cost of the other. We have not performed any explicit trade-off between the two.

Despite some initial successful applications of selected machine learning techniques for IR, there are numerous research directions that need to be pursued before we can develop a robust solution to ''intelligent'' information retrieval. The next phase of our project involves testing and expanding these methods for (relational) database management systems (DBMS) applications and for Internet search and user customization (as a key part of our NSF Digital Library project). We plan to examine the effects of including multiple and numeric attributes for IQBE in the DBMS environment, and the prospects of creating a GA-based intelligent agent (spider) for Internet applications.

We believe this research has shed light on the feasibility and usefulness of the newer machine learning algorithms for IR. However, more extensive and systematic studies of various system parameters and for other large-scale, real-life applications in DBMS and Internet are needed. We hope by incorporating into IR, inductive learning capabilities that are complementary to the prevailing full-text, keyword-based, probabilistic, or knowledge-based techniques, we will be able to advance the design of adaptive and ''intelligent'' information systems.

6. Acknowledgments

This project was supported mainly by: (1) NSF grant IRI-9211418, 1992–1994, ''Building a Concept Space for an Electronic Community System'' (NSF CISE Research Initiation Award); (2) AT&T Foundation's 1994 Special Purpose Grant in Science and Engineering Program, 1994–1995; and (3) NSF grant IRI 9411318, 1994–1998, ''Building the Interspace: Digital Library Infrastructure for a University Engineering Community'' (NSF/ARPA/NASA Digital Library Initiative).

References

Belew, R. K. (1989). Adaptive information retrieval. In N. J. Belkin & C. J. van Rijsbergen (Eds.), *Proceedings of the Twelfth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval,* Cambridge, MA, June 25–28, 1989 (pp. 11–20). New York: ACM Press.

Blair, D. C., & Maron, M. E. (1985). An evaluation of retrieval effectiveness for a full-text document-retrieval system. *Communications of the ACM, 28*(3), 289–299.

Blosseville, M. J., Hebrail, G., Monteil, M. G., & Penot, N. (1992). Automatic document classification: Natural language processing, statistical analysis, and expert system techniques used together. In N. J. Belkin, P. Ingnersen, & A. M. Pejtersen (Eds.), *Proceedings of the Fifteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval,* Copenhagen, Denmark, June 21–24 1992 (pp. 51–57). New York: ACM Press.

Bookstein, A., & Swanson, D. R. (1975). Probabilistic models for automatic indexing. *Journal of the American Society for Information Science, 26,* 45–50.

Brauen, T. L. (1971). Document vector modification. In G. Salton (Ed.), *The smart retrieval system—Experiments in automatic document processing* (pp. 456–484). Englewood Cliffs, NJ: Prentice-Hall Inc.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression tree.* Monterey, CA: Wadsworth.

Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (1983). An overview of machine learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning, an artificial intelligence approach* (pp. 3–23). Palo Alto, CA: Tioga Publishing Company.

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

Chen, H., & Dhar, V. (1990). User misconceptions of online information retrieval systems. *International Journal of Man-Machine Studies, 32*(6), 673–692.

Chen, H., & Dhar, V. (1991). Cognitive process as a basis for intelligent retrieval systems design. *Information Processing and Management, 27*(5), 405–432.

Chen, H., & Kim, J. (1994–1995). GANNET: A machine learning approach to document retrieval. *Journal of Management Information Systems, 11*(3), 7–41.

Chen, H., & Lynch, K. J. (1992). Automatic construction of networks of concepts characterizing document databases. *IEEE Transactions on Systems, Man and Cybernetics, 22*(5), 885–902.

Chen, H., Lynch, K. J., Basu, K., & Ng, D. T. (1993). Generating, integrating, and activating thesauri for concept-based document retrieval. *IEEE EXPERT, Special Series on Artificial Intelligence in Text-based Information Systems, 8*(2), 25–34.

Chen, H., & She, L. (1994). Inductive query by examples (IQBE): A machine learning approach. In J. F. Nunamaker, Jr. & R. H. Sprague (Eds.), *Proceedings of the 27th Annual Hawaii International Conference on System Sciences (HICSS-27), Information Sharing and Knowledge Discovery Track,* Maui, HI, January 4–7, 1994. (pp. 428–437). Los Alamitos, CA: IEEE Computer Science Press.

Crawford, S. L., & Fung, R. M. (1992). An analysis of two probabilistic model induction techniques. *Statistics and Computing, 2*(2), 83–90.

Crawford, S. L., Fung, R., Appelbaum, L. A., & Tong, R. M. (1991). Classification trees for information retrieval. In *Proceedings of the 8th International Workshop on Machine Learning* (pp. 245–249). San Mateo, CA: Morgan Kaufmann.

Ekmekcioglu, F. C., Robertson, A. M., & Willett, P. (1992). Effectiveness of query expansion in ranked-output document retrieval systems. *Journal of Information Science, 18,* 139–147.

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

Fuhr, N., Hartmann, S., Knorz, G., Lustig, G., Schwantner, M., & Tzeras, K. (1990). AIR/X—A rule-based multistage indexing system for large subject fields. In K. Forbus (Ed.), *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90),* Boston, MA, July 29–August 3, 1990 (pp. 789–795). Menlo Park, CA: AAAI Press.

Fuhr, N., & Pfeifer, U. (1994). Probabilistic information retrieval as a combination of abstraction, inductive learning, and probabilistic assumptions. *ACM Transactions on Information Systems, 12*(1), 92–115.

Fung, R., & Crawford, S. L. (1990). Constructor: A system for the induction of probabilistic models. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, Boston, MA, July 29–August 3, 1990 (pp. 762–769).

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.

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

Gordon, M. D. (1991). User-based document clustering by redescribing subject descriptions with a genetic algorithm. *Journal of the American Society for Information Science, 42,* 311–322.

Ide, E. (1971). New experiments in relevance feedback. In G. Salton (Ed.), *The smart retrieval system—Experiments in automatic document processing* (pp. 337–354). Englewood Cliffs, NJ: Prentice-Hall Inc.

Ide, E., & Salton, G. (1971). Interactive search strategies and dynamic file organization in information retrieval. In G. Salton (Ed.), *The smart retrieval system—Experiments in automatic document processing* (pp. 373–393). Englewood Cliffs, NJ: Prentice-Hall.

Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection.* Cambridge, MA: The MIT Press.

Kwok, K. L. (1989). A neural network for probabilistic information retrieval. In N. J. Belkin & C. J. van Rijsbergen (Eds.), *Proceedings of the Twelfth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval,* Cambridge, MA, June 25–28, 1989 (pp. 21–30). New York: ACM Press.

van Laarhoven, P. J. M., & Aarts, E. H. L. (1988). *Simulated annealing: Theory and applications.* Dordrecht: D. Reidel Publishing Company.

Maron, M. E., & Kuhns, J. L. (1960). On relevance, probabilistic indexing and information retrieval. *Journal of the ACM, 7*(3), 216–243.

Michalewicz, Z. (1992). *Genetic algorithms + data structures = evolution programs.* Berlin Heidelberg: Springer-Verlag.

Michalski, R. S. (1983). A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning, an artificial intelligence approach* (pp. 83–134). Palo Alto, CA: Tioga Publishing Company.

Mooney, R., Shavlik, J., Towell, G., & Gove, A. (1989). An experimental comparison of symbolic and connectionist learning algorithms. In N. S. Sridharan (Ed.), *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89),* Detroit, MI, August 20–25, 1989 (pp. 775–780). San Mateo, CA: Morgan Kaufmann.

Quinlan, J. R. (1979). Discovering rules by induction from large collections of examples. In D. Michie (Ed.), *Expert systems in the microelectronic age* (pp. 168–201). Edinburgh, Scotland: Edinburgh University Press.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning, an artificial intelligence approach* (pp. 463–482). Palo Alto, CA: Tioga Publishing Company.

Rasmussen, E. (1992). Clustering algorithms. In W. B. Frakes & R. Baeza-Yates (Eds.), *Information retrieval: Data structures and algorithms* (pp. 419–442). Englewood Cliffs, NJ: Prentice Hall.

Rocchio, J. J. (1971). Relevance feedback in information retrieval. In G. Salton (Ed.), *The smart retrieval system—Experiments in automatic document processing* (pp. 313–323). Englewood Cliffs, NJ: Prentice-Hall Inc.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group (Eds.), *Parallel distributed processing* (pp. 318–362). Cambridge, MA: The MIT Press.

Ryan, B. F., Joiner, B. L., & Ryan, T. A. (1985). *MINITAB handbook (2nd ed.).* Boston, MA: PWS-KENT Publishing Company.

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

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

Simon, H. (1991). Artificial intelligence: Where has it been, and where is it going? *IEEE Transactions on Knowledge and Data Engineering, 3*(2), 128–136.

Van Rijsbergen, C. J. (1979). *Information retrieval* (2nd ed.). London: Butterworths.

Weiss, S. M., & Kulikowski, C. A. (1991). *Computer systems that learn: Classification and prediction methods from statistics, neural networks, machine learning, and expert systems.* San Mateo, CA: Morgan Kaufmann Publishers, Inc.

Yang, J., Korfhage, R. R., & Rasmussen, E. (1993). Query improvement in information retrieval using genetic algorithms: A report on the experiments of the TREC project. In D. K. Harman (Ed.), *Text Retrieval Conference (TREC-1),* Gaithersburg, MD, November 4–6, 1993 (pp. 31–58). Washington, D.C.: NIST.