Contents

List of Figures

List of Tables

Machine Learning for Information Retrieval: Neural Networks, Symbolic Learning, and Genetic Algorithms

Hsinchun Chen[1]

July 5, 1994

[1]MIS Department, College of Business and Public Administration, University of Arizona, Tucson, Arizona 85721, hchen@bpa.arizona.edu, (602) 621-4153.

Abstract

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 artificial-intelligence based inductive learning techniques including neural networks, symbolic learning, and genetic algorithms. These newer techniques, which are grounded on diverse paradigms, have provided great opportunities for researchers to enhance the information processing and retrieval capabilities of current information storage and retrieval systems.

In this article we first provide an overview of these newer techniques and their use in information science research. In order to familiarize readers with these techniques, we present three popular methods: the connectionist Hopfield network, the symbolic ID3/ID5R, and evolution-based genetic algorithms. We discuss their knowledge representations and algorithms in the context of information retrieval. Sample implementation and testing results from our own research are also provided for each technique. We believe these techniques are robust in their ability to analyze user queries, identify users' information needs, and suggest alternatives for search. With proper user-system interactions, these methods can greatly complement the prevailing full-text, keyword-based, probabilistic, and knowledge-based techniques.

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 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 [17] are able to maneuver and explore in these textual databases.

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 [3]. Probabilistic retrieval techniques have been used to improve the retrieval performance of information retrieval systems [75] [6]. The approach is based on two main parameters, the probability of relevance and the probability of irrelevance of a document. Despite various extensions, probabilistic methodology still requires the *independence assumption* for terms and it suffers from difficulty of estimating term-occurrence parameters correctly [106] [49].

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 [18]. Despite their usefulness, systems of this type are considered *performance systems* [108] – 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 [80] [13] [124]. 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 [90], multiple-layered, feed-forward neural networks such as Backpropagation networks [104], and evolution-based genetic algorithms [47]. Many information science researchers have started to experiment with these techniques as well [2] [66] [21] [22] [49].

In this paper, we aim to review the prevailing machine learning techniques and to present several sample implementations in information retrieval to illustrate the associated knowledge representations and algorithms. Our objectives are to bring these newer techniques to the attention of information science researchers by way of a comprehensive overview and discussion of algorithms. In Section 2, we review the probabilistic and knowledge-based techniques and the emerging machine learning methods developed in artificial intelligence (AI). We then summarize some recent work adopting AI techniques in information retrieval (IR). After the overview, we present in detail a neural network implementation (Hopfield network) in Section 3, a symbolic learning implementation (ID3 and ID5R) in Section 4, and a genetic algorithms implementation in Section 5. Detailed algorithms, selected IR examples, and preliminary testing results are also provided in these sections. We summarize the paper in Section 6.

2 Information Retrieval Using Probabilistic, Knowledge-Based, and Machine Learning Techniques

In classical information retrieval models, relevance feedback, document space modification, probabilistic techniques, and Bayesian inference networks are among the techniques most relevant to our research. In this section, we will first briefly summarize important findings in these areas and then present some results from knowledge-based systems research in information retrieval. 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 at the initial try, it is customary to conduct searches iteratively and reformulate query statements based on evaluation of the previously retrieved documents. One method for automatically generating improved query formulations is the well-known *relevance-feedback* process [106] [101] [59] [60]. A query can be improved iteratively by taking an available query vector (of terms) and adding terms from the relevant documents, while substracting terms from the irrelevant documents. A single iteration of relevance feedback usually produces improvements of from 40 to 60 percent in search precision [106]. A similar approach can also be used to alter the document representation. *Document-vector modification* changes and improves document indexes based on the user relevance feedback of relevant and irrelevant documents [9]. 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* of a given document to a user 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 [42] [44] [49]. In order to simplify computation, an assumption is usually made that terms are distributed independently [75]. In [42] [44], Fuhr and his coworkers discussed probabilistic models as an application of machine learning. They presented three different probabilistic learning strategies for information retrieval. First, the classical binary independence retrieval model [100] [130] implemented a *query-oriented* strategy. In the relevance feedback phase, given a query, relevance information was provided for a set of documents. In the application phase, this model can be applied to all documents in the collection, but only for the same initial query. The second *document-oriented* strategy collected relevance feedback data for a specific document from a set of queries [75]. The parameters derived from these data can be used only for the same document, but for all queries submitted to the system. Neither of these strategies can be generalized to all documents and for all queries. Fuhr et al. proposed a third, *feature-oriented*, strategy. In query-oriented and document-oriented strategies, the concept of abstraction was adopted implicitly by regarding terms associated with the query or the document, instead of the query or document. In this feature-oriented strategy, abstraction was accomplished by using features of terms (e.g., the number of query terms, length of the document text, the with-document frequency of a term, etc.) instead of terms themselves. The feature-oriented strategy provides a more general form of probabilistic learning and produces bigger learning samples for estimation; but the disadvantage is the heuristics required to define appropriate features for analysis. After transforming terms into features, Fuhr and his coworkers [43] adopted more sophisticated general-purpose statistical and machine learning algorithms such as regression methods and the decision-tree building ID3 algorithm [92] for indexing and retrieval. In summary, by using features of terms instead of terms, Fuhr et al. were able to derive larger learning samples during relevance feedback. The general-purpose analytical techniques of regression methods and ID3 they adopted are similar to the techniques to be discussed in this paper.

The use of Bayesian classification and inference networks for information retrieval and indexing represents an extension of the probabilistic models [75] [117]. The basic inference network consists of a document network and a query network [117] [118]

[119] that is intended to capture all of the significant probabilistic dependencies among the variables represented by nodes in the document and query networks. Given the prior probabilities associated with the documents and the conditional probabilities associated with the interior nodes, the posterior probability associated with each node in the network can be computed using Bayesian statistics. The feedback process in a Bayesian inference network is similar to conventional relevance feedback and the estimation problems are essentially equivalent to those observed in probabilistic models. In [119], Tzeras and Hartmann showed that the network can be applied for automatic indexing in large subject fields with encouraging results although it does not perform better than the probabilistic indexing technique described in [43]. In [118], Turtle and Croft showed that given equivalent document representations and query forms, the inference network model performed better than conventional probabilistic models.

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 very different from those developed in symbolic machine learning, neural networks, and genetic algorithms. In the following two subsections, we will first review knowledge-based information retrieval, and then provide an extensive discussion of the recent machine learning paradigms for information retrieval.

2.2 Knowledge-Based Systems in IR

Creating computer systems with knowledge or "intelligence" has long been the goal of researchers in artificial intelligence. Many interesting knowledge-based systems have been developed in the past few decades for such applications as medical diagnosis, engineering troubleshooting, and business decision making [55]. Most of these systems have been developed based on the manual knowledge acquisition process, a significant bottleneck for knowledge-based systems development. A recent approach to knowledge elicitation is referred to as "knowledge mining" or "knowledge discovery" [88] [39]. Grounded on various AI-based machine learning techniques, the approach is automatic and it acquires knowledge or identifies patterns directly from examples or databases. We review some important work in *knowledge-based systems in IR* and *learning systems in IR*, respectively, in the next two subsections.

There have been many attempts to capture information specialists' domain knowledge, search strategies, and query refinement heuristics in document retrieval systems design. Some of such systems are "computer-delegated," in that decision-making has been delegated to the system and some are "computer-assisted," wherein users and the computer form a partnership [11]. Because computer-assisted systems have been shown to be more adaptable and useful for search tasks than computer-delegated systems, many knowledge-based systems of this type have been developed for IR over the past decade.

CoalSORT [82], a knowledge-based system, facilitates the use of bibliographic databases in coal technology. A semantic network, representing an expert's domain knowledge, embodies the system's intelligence. PLEXUS [122], developed by Vickery and Brooks, is an expert system that helps users find information about gardening. Natural language queries are accepted. The system has a knowledge base of search strategies and term classifications similar to a thesaurus. EP-X [111] is a prototype knowledge-based system that assists in searching environmental pollution literature. This system makes extensive use of domain knowledge, represented as hierarchically defined semantic primitives and frames. The system interacts with users to suggest broadening or narrowing operations. GRANT [25], developed by Cohen and Kjeldsen, is an expert system for finding sources of funding for given research proposals. Its search method - constrained spreading activation in a semantic network - makes inferences about the goals of the user and thus finds information that the user has not explicitly requested but that is likely to be useful. Fox's CODER system [38] consists of a thesaurus that was generated from the Handbook of Artificial Intelligence and Collin's Dictionary. In CANSEARCH [89], a thesaurus is presented as a menu. Users browse and select terms for their queries from the menu. It was designed to enable doctors to search the MEDLINE medical database for cancer literature. The "Intelligent Intermediary for Information Retrieval" (I^3R), developed by Croft [28], consists of a group of "experts" that communicate via a common data structure, called the blackboard. The system consists of a user model builder, a query model builder, a thesaurus expert, a search expert (for suggesting statistics-based search strategies), a browser expert, and an explainer. The IOTA system, developed by Chiaramella and Defude, includes natural language processing of queries, deductive capabilities (related to user modeling, search strategies definition, use of expert and domain knowledge), management of full-text documents, and relevance evaluation of answers [24]. Chen and Dhar's METACAT [18] incorporates several human search strategies and a portion of the Library of Congress Subject Headings (LCSH) for bibliographic search. The system also includes a branch-and-bound algorithm for an automatic thesaurus (LCSH) consultation process.

The National Library of Medicine's thesaurus projects are probably the largest-scale effort that uses the knowledge in existing thesauri. In one of the projects, Rada and Martin [94] [76] conducted experiments for the automatic addition of concepts to MeSH (Medical Subject Headings) by including the CMIT (Current Medical Information and Terminology) and SNOMED (Systematized Nomenclature of Medicine) thesauri. Access to various sets of documents can be facilitated by using thesauri and the connections that are made among thesauri. The Unified Medical Language System (UMLS) project is a long-term effort to build an intelligent automated system that understands biomedical terms and their interrelationships and uses this understanding to help users retrieve and organize information from machine-readable sources [58] [78] [72]. The UMLS includes a Metathesaurus, a Semantic Network, and an Information Sources Map. The Metathesaurus contains information about biomedical concepts and their representation in more than 10 different vocabularies and thesauri. The Semantic Network contains information about the types of terms (e.g., “disease”, “virus,” etc.) in the Metathesaurus and the permissible relationships among these types. The Information Sources Map contains information about the scope, location, vocabulary, and access conditions of biomedical databases of all kinds.

Another important component of information retrieval is user modeling capability, which is a unique characteristic of reference librarians. During the user-librarian consultation process, the librarian develops an understanding of the type of user being dealt with on the basis of verbal and non-verbal clues. Usually, the educational level of the user, the type of question, the way the question is phrased, the purpose of the search, and the expected search results all play major roles in helping the librarian determine the needs of the user. The librarian, in essence, creates models of the user profile and the task requirements during the consultation process.

User modeling has played a crucial role in applications such as question-answering systems, intelligent tutoring systems, and consultation systems [1] [114] [110] [131] [16]. An intelligent interface for document retrieval systems must also exhibit the user modeling capability of experienced human intermediaries. Daniels proposed a frame-based representation for a user model and rules for interacting with the users. She has shown that user modeling is a necessary function in the pre-search information interaction [30]. Rich’s Grundy system builds models of its users, with the aid of stereotypes, and then uses those models to guide it in its task, suggesting novels that people may find interesting [97] [98] [99]. IR-NLI II [8] incorporates user modeling into a domain-independent bibliographic retrieval expert system. A user model is built based on the user’s amount of domain knowledge and search experience.

Despite successes in numerous domains, the development process for knowledge-based systems is often slow and painstaking. Knowledge engineers or system designers need to be able to identify subject and classification knowledge from some sources (usually some domain experts) and to represent the knowledge in computer systems. The inference engines of such systems, which mainly emulate human problem-solving strategies and cognitive processes [18], may not be applicable across different applications.

After examining the potential contribution of knowledge-based techniques (natural language processing and expert systems, in particular) to the information retrieval and management tasks, Sparck Jones [112] warned that it is important not to overestimate the potential of such techniques for IR. She argued that for really hard tasks we will not be able to replace humans by machines in the foreseeable future and many information operations are rather shallow, linguistic tasks, which do not involve elaborate reasoning or complex knowledge. However, she believed AI can contribute to specialized systems and in situations where users and systems complement each other (i.e., computer-assisted systems).

2.3 Learning Systems: Neural Networks, Symbolic Learning, and Genetic Algorithms

Unlike the manual knowledge acquisition process and the linguistics-based natural language processing technique used in knowledge-based systems design, learning systems rely on algorithms to extract knowledge or identify patterns in examples or data. Various statistics-based algorithms have been developed by management scientists and have been used extensively over the past few decades for quantitative data analysis. These algorithms examine quantitative data for the purposes of [86]: 1) clustering descriptors with common characteristics, e.g., nearest neighbor methods, factor analysis, and principal components analysis; 2) hypothesis testing for differences among different populations, e.g., t-test and analysis of variance (ANOVA); 3) trend analysis, e.g., time series analysis; and 4) correlation between variables, e.g., correlation coefficient, discriminant analysis, and linear/multiple regression analysis [40] [84]. These analysis techniques often rely on complex mathematical models, stringent assumptions, or special underlying distributions. The findings are then presented in mathematical formulas and parameters.

2.3.1 Learning Systems: An Overview

The symbolic machine learning technique, the resurgent neural networks approach, and evolution-based genetic algorithms provide drastically different methods of data analysis and knowledge discovery [124] [123] [85] [34] [62] [15]. These techniques, which are diverse in their origins and behaviors, have shown unique capabilities for analyzing both qualitative, symbolic data and quantitative, numeric data. We provide below a brief overview of these three classes of techniques, along with a representative technique for each class.

- *Symbolic Learning and ID3:*

  Symbolic machine learning techniques, which can be classified based on such underlying learning strategies as rote learning, learning by being told, learning by analogy, learning from examples, and learning from discovery [13], have been studied extensively by AI researchers over the past two decades. Among these techniques, learning from examples, a special case of inductive learning, appears to be the most promising symbolic machine learning technique for knowledge discovery or data analysis. It induces a general concept description that best describes the positive and negative examples. Examples of algorithms which require both positive and negative examples are Quinlan's ID3 [91] and Mitchell's Version Space [81]. Some algorithms are batch-oriented, such as Stepp and Michalski's CLUSTER/RD algorithm [113] and ID3; but some are incremental, such as Utgoff's ID5R [120]. Many algorithms create a hierarchical arrangement of concepts for describing classes of objects, including Lebowitz' UNIMEM [67], Fisher's COBWEB [34], and Brieman's CART [10]. Most of the symbolic learning algorithms produce production rules or concept hierarchies as outputs. These representations are easy to understand and their implementation is typically efficient (especially when compared with neural networks and genetic algorithms).

Among the numerous symbolic learning algorithms which have been developed over the past 15 years, Quinlan's ID3 decision-tree building algorithm and its descendants [91] [92] are popular and powerful algorithms for inductive learning. ID3 takes objects of a known class, described in terms of a fixed collection of properties or attributes, and produces a decision tree incorporating these attributes that correctly classifies all the given objects. It uses an information-economics approach aimed at minimizing the expected number of tests to classify an object. Its output can be summarized in terms of IF-THEN rules.

- *Neural Networks and Backpropagation:*

  The foundation of the neural networks paradigm was laid in the 1950s and this approach has attracted significant attention in the past decade due to the development of more powerful hardware and neural algorithms [105]. Nearly all connectionist algorithms have a strong learning component. In symbolic machine learning, knowledge is represented in the form of symbolic descriptions of the learned concepts, e.g., production rules or concept hierarchies. In connectionist learning, on the other hand, knowledge is learned and remembered by a network of interconnected neurons, weighted synapses, and threshold logic units [73] [103]. Learning algorithms can be applied to adjust connection weights so that the network can predict or classify unknown examples correctly. Neural networks have been adopted in various engineering, business, military, and biomedical domains [73] [19] [109] [125]. For example, Hopfield networks have been used extensively in the area of global optimization and search [115] [57]; Kohonen networks have been adopted in unsupervised learning and pattern recognition [64]. For a good overview of various artificial neural systems, readers are referred to [73].

  Among the numerous artificial neural networks which have been proposed recently, Backpropagation networks have been extremely popular for their unique learning capability [125]. Backpropagation networks [104] are fully connected, layered, feed-forward models. Activations flow from the input layer through the hidden layer, then to the output layer. A Backpropagation network typically starts out with a random set of weights. The network adjusts its weights each time it sees an input-output pair. Each pair is processed at two stages, a forward pass and a backward pass. The forward pass involves presenting a sample input to the network and letting activations flow until they reach the output layer. During the backward pass, the network's actual output is compared with the target output and error estimates are computed for the output units. The weights connected to the output units are adjusted in order to reduce the errors (a gradient descent method). The error estimates of the output units are then used to derive error estimates for the units in the hidden layer. Finally, errors are propagated back to the connections stemming from the input units. The Backpropagation network updates its weights incrementally until the network stabilizes.

- *Simulated Evolution and Genetic Algorithms:*

  During the past decade there has been a growing interest in algorithms which rely on analogies to natural processes and Darwinian survival of the fittest. The emergence of massively parallel computers made these algorithms of practical interest. There are currently three main avenues of research in simulated evolution: genetic algorithms, evolution strategies, and evolutionary programming [35]. Each method emphasizes a different facet of natural evolution. Genetic algorithms stress chromosomal operations such as crossover and mutation [56] [5]. Evolution strategies emphasize individual behavioral changes. Evolutionary programming stresses behavioral changes at the level of the species [36] [37]. Fogel provides an excellent review of the history and recent efforts in this area in [35]. Among these methods, genetic algorithms have been used successfully for various optimization problems in engineering and biomedical domains.

  Genetic algorithms were developed based on the principle of genetics [47] [65] [79]. In such algorithms a population of individuals (potential solutions) undergoes a sequence of unary (mutation) and higher order (crossover) transformations. These individuals strive for survival: a selection (reproduction) scheme, biased towards selecting fitter individuals, produces the individuals for the next generation. After some number of generations the program converges – the best individual represents the optimum solution.

Over the past years there have been several studies which compared the performance of these techniques for different applications as well as some systems which used hybrid representations and learning techniques. We summarize some of these studies below.

In [85], Mooney et al. found that ID3 was faster than a Backpropagation net, but the Backpropagation net was more adaptive to noisy data sets. The performances of these two techniques were comparable, however. Weiss and Kapouleas [123] [124] suggested using a re-sampling technique such as leave-one-out for evaluation instead of using a hold-out testing data set. Discriminant analysis methods, Backpropagation net, and decision tree-based inductive learning methods (ID3-like) were found to achieve comparable performance for several data sets. Fisher and McKusick [34] found that using batch learning, Backpropagation performed as well as ID3, but it was more noise-resistant. They also compared the effect of incremental learning versus batch learning. In [62], Kitano performed systematic, empirical studies on the speed of convergence of Backpropagation networks and genetic algorithms. The results indicated that genetic search is, at best, equally efficient as faster variants of a Backpropagation algorithm in very small scale networks, but far less efficient in larger networks. Earlier research by Montana and Davis [83], however, showed that using some domain-specific genetic operators to train the Backpropagation network, instead of using the conventional Backpropagation Delta learning rule, improved performance. Harp et al. [54] also achieved good results by using GAs for neural network design.

Systems developed by Kitano [62] and Harp et al. [54] are also considered hybrid systems (genetic algorithms and neural networks), as are systems like COGIN [51] which performed symbolic induction using genetic algorithms and SC-net [52] which is a fuzzy connectionist expert system. Other hybrid systems developed in recent years employ symbolic and neural net characteristics. For example, Touretzky and Hinton [116] and Gallant [46] proposed connectionist production systems, and Derthick [31] and Shastri [107] developed different connectionist semantic networks.

2.3.2 Learning Systems in IR

The adaptive learning techniques cited have also drawn attention from researchers in information science in recent years. In particular, Doszkocs et al. [32] has provided an excellent review of connectionist models for information retrieval and Lewis [68] has briefly surveyed previous research on machine learning in information retrieval and discussed promising areas for future research at the intersection of these two fields.

- *Neural Networks and IR:*

  Neural networks computing, in particular, seems to fit well with conventional retrieval models such as the vector space model [106] and the probabilistic model [75]. In [32], Doszkocs et al. provided an excellent overview of the use of connectionist models in information retrieval. These models include several related information processing approaches, such as artificial neural networks, spreading activation models, associative networks, and parallel distributed processing. In contrast to more conventional information processing models, connectionist models are "self-processing" in that no external program operates on the network: the network literally processes itself, with "intelligent behavior" emerging from the local interactions that occur concurrently between the numerous network nodes through their synaptic connections. By taking a broader definition of connectionist models, these authors were able to discuss the well-known vector space model, cosine measures of similarity, and automatic clustering and thesaurus in the context of *network representation.* Based on the network representation, spreading activation methods such as constrained spreading activation adopted in GRANT [25] and the branch-and-bound algorithm adopted in METACAT [18] can be considered as variants of connectionist activation. However, only a few systems are considered classical connectionist systems that typically consist of weighted, unlabelled links and exhibit some adaptive learning capabilities.

The work of Belew is probably the earliest connectionist model adopted in IR. In AIR [2], he developed a three-layer neural network of authors, index terms, and documents. The system used relevance feedback from its users to change its representation of authors, index terms, and documents over time. The result was a representation of the consensual meaning of keywords and documents shared by some group of users. One of his major contributions was the use of a modified correlational learning rule. The learning process created many new connections between documents and index terms. In [102], Rose and Belew extended AIR to a hybrid connectionist and symbolic system called SCALIR which used analogical reasoning to find relevant documents for legal research. Kwok [66] also developed a similar three-layer network of queries, index terms, and documents. A modified Hebbian learning rule was used to reformulate probabilistic information retrieval. Wilkinson and Hingston [126] [127] incorporated the vector space model in a neural network for document retrieval. Their network also consisted of three layers: queries, terms, and documents. They have shown that spreading activation through related terms can help improve retrieval performance.

While the above systems represent information retrieval applications in terms of their main components of documents, queries, index terms, authors, etc. other researchers used different neural networks for more specific tasks. Lin [71] adopted a Kohonen network for information retrieval. Kohonen's feature map, which produced a two-dimensional grid representation for N-dimensional features, was applied to construct a self-organizing (unsupervised learning), visual representation of the semantic relationships between input documents. In [74], a neural algorithm developed by MacLeod was used for document clustering. The algorithm compared favorably with conventional hierarchical clustering algorithms. Chen et al. [21] [22] [14] reported a series of experiments and system developments which generated an automatically-created weighted network of keywords from large textual databases and integrated it with several existing man-made thesauri (e.g., LCSH). Instead of using a three-layer design, Chen's systems developed a single-layer, interconnected, weighted/labelled network of keywords (concepts) for "concept-based" information retrieval. A blackboard-based design which supported browsing and automatic concept exploration using the Hopfield neural network's parallel relaxation method was adopted to facilitate the usage of several thesauri [22]. In [14] the performance of a branch-and-bound serial search algorithm was compared with that of the parallel Hopfield network activation in a hybrid neural-semantic network (one neural network and two semantic networks). Both methods achieved similar performance, but the Hopfield activation method appeared to activate concepts from different networks more evenly.

- *Symbolic Learning and IR:*

  Despite the popularity of using neural networks for information retrieval, we see only limited use of symbolic learning techniques for IR. In [4], 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. Text classification involves the task of classifying documents with respect to a set of two or more predefined classes [69]. A number of systems were built based on human categorization rules (a knowledge-based system approach) [96]. However, a range of statistical techniques including probabilistic models, factor analysis, regression, and nearest neighbor methods have been adopted [69] [77] [4]. Fuhr et al. [43] adopted regressions methods and ID3 for their feature-based automatic indexing technique. Crawford, Fung, and their coworkers [45] [26] [27] have developed a probabilistic induction technique called CONSTRUCTOR and have compared it with the popular CART algorithm [10]. 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 [23], 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. For a test collection of about 1,000 documents, both symbolic learning algorithms did a good job in identifying the concepts (keywords) which best represent the set of documents identified by users as relevant (positive examples) and irrelevant (negative examples). More testing, however, is under way to determine the effectiveness of example-based document retrieval using ID3 and ID5R.

  Several recent works which involved using symbolic learning techniques in the related database areas were also identified, especially in relational database management systems (RDBMS). In [12] [53], Cai et al. developed an attribute-oriented, tree-ascending method for extracting characteristic and classification rules from relational databases. The technique relied on some existing conceptual tree for identifying higher-level, abstract concepts in the attributes. In [61], Ioannidis et al. examined the idea of incorporating machine learning algorithms (UNIMEM and COBWEB) into a database system for monitoring the stream of incoming queries and generating hierarchies with the most important concepts expressed in those queries. The goal is for these hierarchies to provide valuable input for dynamically modifying the physical and logical designs of a database. Also related to database design, Borgida and Williamson [7] proposed the use of machine learning to represent exceptions in databases that are based on semantic data models. Li and McLeod [70] used machine learning techniques to handle object flavor evolution in object-oriented databases.

- *Genetic Algorithms and IR:*

  Our literature search revealed several implementations of genetic algorithms in information retrieval. In [49], Gordon presented a genetic algorithms based approach for document indexing. Competing document descriptions (keywords) are associated with a document and altered over time by using genetic mutation and crossover operators. In his design, a keyword represents a gene (a bit pattern), a document's list of keywords represents individuals (a bit string), and a collection of documents initially judged relevant by a user represents the initial population. Based on a Jaccard's score matching function (fitness measure), the initial population evolved through generations and eventually converged to an optimal (improved) population – a set of keywords which best described the documents. In [50], Gordon adopted a similar approach to document clustering. His experiment showed that after genetically redescribing the subject description of documents, descriptions of documents found co-relevant to a set of queries will bunch together. Redescription improved the relative density of co-relevant documents by 39.74% after 20 generations and 56.61% after 40 generations. Raghavan and Agarwal [95] have also studied the genetic algorithms in connection with document clustering. In [87], Petry et al. applied genetic programming to a weighted information retrieval system. In their research, a weighted Boolean query was modified in order to improve recall and precision. They found that the form of the fitness function has a significant effect upon performance. Yang and his coworkers [128] [129] 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. Frieder and Siegelmann [41] also reported a data placement strategy for parallel information retrieval systems using a genetic algorithms approach. Their results compared favorably with pseudo-optimal document allocations. In

[20], a GA-NN hybrid system, called GANNET, was developed for IR. 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.

Following this overview, we present three sample implementations of neural networks, symbolic learning, and genetic algorithms, respectively, for illustration purposes. We hope that examining these implementations in the context of IR will encourage other researchers to appreciate these techniques and adopt them in their own research.

3 Neural Networks for IR

Neural networks provide a convenient knowledge representation for IR applications in which nodes typically represent IR objects such as keywords, authors, and citations and bidirectional links represent their weighted associations (of relevance). The learning property of Backpropagation networks and the parallel search property of the Hopfield network provide effective means for identifying relevant information items in databases. Variants of the Backpropagation learning in IR can be found in [2] [66]. In this section, we review a Hopfield network implementation and its associated parallel search property.

3.1 A Hopfield Network: Knowledge Representation and Procedure

The Hopfield net [57] [115], was introduced as a neural net that can be used as a content-addressable memory. Knowledge and information can be stored in single-layered interconnected neurons (nodes) and weighted synapses (links) and can be retrieved based on the network's parallel relaxation method – nodes are activated in parallel and are traversed until the network reaches a stable state (convergence). It had been used for various classification tasks and global optimization [73] [109].

A variant of the Hopfield network for creating a network of related keywords developed by Chen et al. [21] [22] used an asymmetric similarity function to produce thesauri (or knowledge bases) for different domain-specific databases. These automatic thesauri were then integrated with some existing manually-created thesauri for assisting concept exploration and query refinement. A variant of the Hopfield parallel relaxation procedure for network search [22] and concept clustering [19] had been reported earlier.

The implementation reported below incorporated the basic Hopfield net iteration and convergence ideas. However, significant modification was also made to accommodate unique characteristics of information retrieval, e.g., asymmetric link weights and the continuous SIGMOID transformation function. With the initial search terms provided by searchers and the association of keywords captured by the network, the Hopfield parallel relaxation algorithm activates neighboring terms, combines weighted links, performs a transformation function (a SIGMOID function, f_s), and determines the outputs of newly activated nodes. The process repeats until node outputs remain unchanged with further iterations. The node outputs then represent the concepts that are strongly related to the initial search terms. A sketch of the Hopfield net activation algorithm follows:

1. **Assigning synaptic weights:**

   For thesauri which were generated automatically using a similarity function (e.g., the COSINE function) [33], the resulting links represent probabilistic, synaptic weights between any two concepts. For other external thesauri which contain only symbolic links (e.g., narrower term, synonymous term, broader term, etc.), a user-guided procedure of assigning a probabilistic weight to each symbolic link can be adopted [22].

   The "training" phase of the Hopfield net is completed when the weights have been computed or assigned. t_{ij} represents the "synaptic" weight from node i to node j.

2. **Initialization with search terms:**

   An initial set of search terms is provided by searchers, which serves as the input pattern. Each node in the network which matches the search terms is initialized (at time 0) to have a weight of 1.

   $\mu_i(0) = x_i,\ 0 \leq i \leq n-1$

   $\mu_i(t)$ is the output of node i at time t and x_i which has a value between 0 and 1, indicates the input pattern for node i.

3. **Activation, weight computation, and iteration:**

   $$\mu_j(t+1) = f_s[\sum_{i=0}^{n-1} t_{ij}\mu_i(t)],\ 0 \leq j \leq n-1$$

   where f_s is the continuous SIGMOID transformation function as shown below [63] [29].

$$f_s(net_j) = \frac{1}{1 + \exp[\frac{-(net_j - \theta_j)}{\theta_0}]}$$

where $net_j = \sum_{i=0}^{n-1} t_{ij}\mu_i(t)$, θ_j serves as a threshold or bias and θ_0 is used to modify the shape of the SIGMOID function.

This formula shows the *parallel relaxation* property of the Hopfield net. At each iteration, all nodes are activated at the same time. The weight computation scheme, $net_j = \sum_{i=0}^{n-1} t_{ij}\mu_i(t)$, is a unique characteristic of the Hopfield net algorithm. Based on parallel activation, each newly activated node derives its new weight based on the summation of the products of the weights assigned to its neighbors and their synapses.

4. **Convergence:**

   The above process is repeated until there is no change in terms of output between two iterations, which is accomplished by checking:

$$\sum_{j=0}^{n-1} |\mu_j(t+1) - \mu_j(t)| \leq \epsilon$$

   where ϵ is the maximal allowable error (a small number). The final output represents the set of terms relevant to the starting keywords. Some default threshold values were selected for (θ_j, θ_0).

3.2 A Hopfield Network Example

A sample session of the Hopfield net spreading activation is presented below. Three thesauri were incorporated in the experiment: a Public thesaurus (generated automatically from 3000 articles extracted from DIALOG), the ACM Computing Review Classification System (ACM CRCS), and a portion of the Library of Congress Subject Headings (LCSH) in the computing area. The links in the ACM CRCS and in the LCSH were assigned weights between 0 and 1. Several user subjects (MIS graduate students) were also asked to reviewed selected articles and create their own folders for topics of special interest to them. Notice that some keywords were folder names assigned by the users (in the format of *.*), e.g., QUERY.OPT folder for query optimization topics, DBMS.AI folder for artificial intelligence and databases topics, KEVIN.HOT folder for "HOT" (current) topics selected by a user, Kevin. In the example shown below, the searcher was asked to identify descriptors which were relevant to "knowledge indexed deductive search." The initial search terms were: "information retrieval," "knowledge base," "thesaurus," and "automatic indexing" (as shown in the following interaction).

Iteration no.	Suggested Terms	Activations
0	INFORMATION RETRIEVAL	1.00
	KNOWLEDGE BASE	1.00
	THESAURUS	1.00
	AUTOMATIC INDEXING	1.00
1	INDEXING	0.65
	KEVIN.HOT	0.56
	CLASSIFICATION	0.50
	EXPERT SYSTEMS	0.50
	ROSS.HOT	0.44
2	RECALL	0.50
3	INFORMATION RETRIEVAL SYSTEM EVALUATION	0.26
4	SELLING - INFORMATION STORAGE AND RETRIEVAL SYSTEMS	0.15
...	...	...

Table 1: Sample Hopfield net iterations

```
*--------------------*
    Initial terms:  {* Supplied by the subject. *}
    -------------
 1. (P L) INFORMATION RETRIEVAL {* P: Public, A: ACM, L: LCSH *}
 2. (P  ) KNOWLEDGE BASE
 3. (P  ) THESAURUS
 4. (P L) AUTOMATIC INDEXING
*--------------------*

Enter the number of system-suggested terms or '0' to quit >> 10
{* The users supplied a target number of relevant terms.  *}
```

Given these starting terms, the Hopfield net iterated and converged after 11 iterations. The activated terms after the first four iterations and their associated levels of activation are shown in Table 1. Due to the damping effect of the parallel search property (i.e., the farther away from the initial search terms, the weaker the activation), terms activated at later iterations had lower activation values and were less relevant to the initial search terms in general. Fourteen terms were suggested after the complete Hopfield net activation. Searchers could browse the system-suggested list, select terms of interest, and then activate the Hopfield net again. The user-system interaction continued until the user decided to stop.

```
{* The system reported 14 relevant terms as shown below. *}
```

```
  1. ( ) INDEXING
  2. ( ) SELLING - INFORMATION STORAGE AND RETRIEVAL SYSTEMS
  3. ( ) KEVIN.HOT
  4. ( ) INFORMATION RETRIEVAL SYSTEM EVALUATION
  5. ( ) RECALL
  6. ( ) EXPERT SYSTEMS
  7. ( ) CLASSIFICATION
  8. ( ) DBMS.AI
  9. ( ) ROSS.HOT
 10. ( ) INFORMATION STORAGE AND RETRIEVAL SYSTEMS
 11. ( ) INFORMATION RETRIEVAL
 12. ( ) KNOWLEDGE BASE
 13. ( ) THESAURUS
 14. ( ) AUTOMATIC INDEXING

Enter numbers [1 to 14] or '0' to quit:  1, 2, 4, 5, 7, 10-14
{* The user selected terms he deemed relevant.
   The system confirmed the selections made and display the
   source for each term. *}

 1. (P  ) INDEXING
 2. (  L) SELLING - INFORMATION STORAGE AND RETRIEVAL SYSTEMS
 3. (P  ) INFORMATION RETRIEVAL SYSTEM EVALUATION
 4. (P  ) RECALL
 5. (P  ) CLASSIFICATION
 6. (  L) INFORMATION STORAGE AND RETRIEVAL SYSTEMS
 7. (P L) INFORMATION RETRIEVAL
 8. (P  ) KNOWLEDGE BASE
 9. (P  ) THESAURUS
10. (P L) AUTOMATIC INDEXING

Enter the number of system-suggested terms or '0' to quit >> 30
{* The uses decide to broaden the search by requesting the
   Hopfield network to identify 30 new terms based on the
   terms he had selected. *}
 ........

Enter number [1 to 40] or '0' to quit:  3-7, 9, 33, 35, 36, 38
 ........

Enter numbers [1 to 67] or '0' to quit: 0
{* The system listed his final selections. *}

 1. (P  ) PRECISION
 2. (P L) INFORMATION RETRIEVAL
 3. (P  ) INDEXING
 4. (P L) AUTOMATIC INDEXING
 5. (P  ) RECALL
 6. (  L) AUTOMATIC ABSTRACTING
 7. (  L) AUTOMATIC CLASSIFICATION
 8. (  L) AUTOMATIC INFORMATION RETRIEVAL
 9. (P  ) INFORMATION RETRIEVAL SYSTEM EVALUATION
10. (P  ) THESAURUS
11. (  L) INFORMATION STORAGE AND RETRIEVAL SYSTEMS
12. (P  ) KNOWLEDGE BASE

{* A total of 12 terms were selected. Eight terms were suggested by the
   Hopfield net algorithm. *}
```

In a more structured benchmark experiment, we tested 30 sample queries using the

Hopfield algorithm in an attempt to understand the general behavior of the algorithm. We tested 5 cases each for queries with 1 term, 2 terms, 3 terms, 4 terms, 5 terms, and 10 terms, a total of 30 cases. A few examples of the queries used, all in the computing area, were: (1-term: Natural Language Processing), (2-terms: Group Decision Support Systems, Collaboration), (3-terms: Systems Analysis and Design, Simulation and Modeling, Optimization), etc.

For each query, we selected terms from different knowledge sources, "P" for the Public KB, "A" for the ACM CRCS, and "L" for the LCSH, as shown in Table 2. Some terms may have appeared in more than one knowledge source. The three knowledge sources contained about 14,000 terms and 80,000 weighted links. The results shown in Table 2 reveal the number of iterations, the computing times, and the sources of knowledge for the query terms and the system-suggested terms. The reason for investigating the source of knowledge for system-suggested terms was to show the extent to which the Hopfield algorithm branched out and utilized knowledge from various knowledge sources.

Despite the variation in the number of starting terms, the response times increased only slightly when the number of starting terms was increased. The average response time was 24.5 seconds after about an average of about 19 iterations by the Hopfield network. The reason for this was that the Hopfield net thresholds (θ_0 and θ_j) helped prune the search space. However, more stringent thresholds may need to be adopted to achieve reasonable real-time response for large databases.

Another important observation was that the Hopfield net appeared to invoke the different knowledge sources quite evenly. As shown in Table 2, for most queries the Hopfield net (NN) almost always produced terms from all three knowledge sources. Most terms suggested by the algorithm appeared relevant and many of them were multiple links away from the initial search terms (conventional Hypertext browsing does not traverse multiple links effectively). However, detailed user studies need to be performed in order to examine the usefulness of the algorithm in search, especially for large-scale applications.

4 Symbolic Learning for IR

Even though symbolic learning techniques have been adopted frequently in various database, engineering, and business domains, we see only limited use of such techniques in IR. For illustration purposes, we summarize below a symbolic learning for IR implementation based on the ID3 and ID5R algorithms [23].

Case	No. of terms	Query terms in (P,A,L)	Suggested terms in NN:(P,A,L)	No. of iterat. NN	Times (secs) NN
1	1	(1,1,1)	(12,7,7)	18	21
2	1	(1,0,1)	(5,0,16)	15	14
3	1	(1,1,1)	(11,5,11)	14	18
4	1	(0,0,1)	(0,0,20)	11	10
5	1	(1,0,1)	(4,4,19)	17	26
6	2	(2,1,0)	(19,2,3)	21	18
7	2	(2,0,2)	(16,0,8)	19	22
8	2	(2,0,0)	(20,3,4)	20	24
9	2	(2,1,1)	(11,5,11)	15	16
10	2	(2,1,2)	(11,0,12)	27	29
11	3	(3,0,1)	(20,0,18)	19	31
12	3	(1,2,1)	(4,11,8)	22	34
13	3	(2,1,3)	(22,1,8)	18	29
14	3	(1,3,1)	(20,2,2)	16	23
15	3	(1,2,2)	(13,9,3)	9	10
16	4	(2,2,4)	(17,4,4)	17	11
17	4	(3,2,2)	(11,2,13)	19	31
18	4	(2,3,2)	(18,5,6)	24	33
19	4	(1,3,4)	(18,2,5)	19	32
20	4	(1,2,1)	(15,8,3)	18	6
21	5	(1,4,1)	(19,4,6)	16	27
22	5	(4,2,2)	(10,1,12)	15	27
23	5	(3,2,4)	(2,0,18)	11	23
24	5	(5,0,1)	(19,0,3)	23	33
25	5	(5,0,1)	(20,0,1)	12	30
26	10	(8,0,3)	(11,0,13)	17	34
27	10	(10,1,3)	(13,2,10)	25	32
28	10	(8,0,4)	(16,0,8)	24	36
29	10	(9,1,5)	(19,1,6)	27	25
30	10	(8,2,3)	(20,2,3)	28	31
average	5	(3.1,1.2,1.9)	(14.5,2.5,8.5)	18.8	24.5

Table 2: Results of Hopfield network testing

4.1 ID3/ID5R: Knowledge Representation and Procedure

ID3 is a decision-tree building algorithm developed by Quinlan [90] [91]. It adopts a divide-and-conquer strategy for object classification. Its goal is to classify mixed objects into their associated classes based the objects' attribute values. In a decision tree, one can classify a node as:

- a leaf node that contains a class name, or
- a non-leaf node (or decision node) that contains an attribute test.

Each training instance or object is represented as a list of attribute-value pairs, which constitutes a conjunctive description of that instance. The instance is labeled with the name of the class to which it belongs. Using the divide-and-conquer strategy, ID3 picks an attribute and uses it to classify a list of objects based on their values associated with this attribute. The subclasses which are created by this division procedure are then further divided by picking other attributes. This process continues until each subclass produced contains only a single type of objects. In order to produce the simplest decision tree (a minimal tree) for classification purpose, ID3 adopts an information-theoretic approach which aims at minimizing the expected number of tests to classify an object. An *entropy* (a measure of uncertainty) concept is used to help decide which attribute should be selected next. In general, an attribute which can help put objects in their proper classes tends to reduce more *entropy* and thus should be selected as a test node.

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

- the "positive" class (+): consisting of records that are desired; and
- the "negative" class (-): consisting of records that are undesired.

Different database users may desire different sets of documents due to their unique information needs, and the set of documents desired by one user often constitutes only a small portion of the entire database. Enabling the system to identify this small set of positive documents is therefore a challenging task.

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 classification 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. This set of documents served as the training examples for the ID3 algorithm. Entropy was calculated by using the following function [91]:

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

   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. 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.

Considered as an incremental version of the ID3 algorithm, ID5R, developed by Utgoff [120], is guaranteed to build the same decision tree as ID3 for a given set of training instances [93]. In ID5R, a non-leaf node contains an attribute test (same as in ID3) and a set of other non-test attributes, each with object counts for the possible values of the attribute. This additional non-test attribute and object count information at each no-leaf node allows ID5R to update a decision tree without rebuilding the entire tree. During the tree re-building process, an old test node may be replaced by a new attribute or swapped to other positions in the tree. As in ID3, the tree building process requires much less computation and time than other inductive learning methods, including neural networks and genetic algorithms.

In order to create a robust and real-time inductive learning system, a *relevance feedback* scheme was introduced into our implementation. Although the proposed inductive learning algorithms require users to provide examples to confirm their interests, it is inconceivable that users will be able to browse the entire database to identify such instances. An incremental, interactive feedback process therefore was designed to allow users to examine a few documents at a time. In essence, our ID5R algorithm was implemented such that it provided a few suggested documents based on the documents initially provided by a user after examining a small portion of the database. When a predetermined number of desired documents had been found (say 3, in our implementation), the system presented these documents to the user immediately for evaluation (as desired or undesired). This iterative system-induction and user-feedback process continued until the user decided to stop or the complete database had been traversed.

During the relevance feedback process, the newly confirmed documents, either desired or undesired, could be used by ID5R to update the decision tree it previously had constructed. It was shown that when more examples are provided by the users and when the database is more exhaustively searched, ID5R can significantly improve its classification accuracy and search performance.

4.2 An ID3/ID5R Example

We created a small test database of 60 records. For evaluation purposes, we were able to manually select a small set of target desired documents (i.e., 8 documents in the areas of information retrieval and keywording). The goal of the experiment was to present a few documents at a time to our system and see whether the system would be able to identify them after the iterative relevance feedback process. The performance of our ID5R-based system was also compared with that of the more conventional ID3 algorithm, which used only an initial set of desired documents to generate a query tree. Sample entries in the literature database are shown below, where the first column represents the document number, and the remaining columns represent different numbers of keywords (2-5) associated with the document.

...	...
010	generic, keyword, reference
013	modeling, thesaurus, terrorism
014	modeling, simulation, thesaurus, terrorism
018	keyword, thesaurus
021	ID3, AI, NN
022	file, keyword
023	hierarchy, interface, index
030	carat, AI, expert, keyword, thesaurus
031	AI, protocol, thesaurus
048	keyword, retrieval
049	cross-reference, remote use, redundancy
050	expectations, market, maintenance, quel, interface
...	...
107	IT, computerized, MIS
149	database, query, keyword

152	sort, indexing, merge, keyword
177	country, code, keyword, ISO

Initially the user was able to identify the following documents as desired (+) or undesired (−), respectively (documents which the user had seen before):

006	thesaurus, remote use, keyword (+)
008	retrieval, interface (+)
083	syntax checking, remote use, test, user (−)
084	interface, protocol, standardization (−)

Providing negative documents was optional. If a user could not think of an example of a document which was undesired, the system by default automatically generated one negative document which contained no keyword identical to any that was present in the desired set. The initial positive keyword list then consisted of all keywords from desired documents, i.e., thesaurus, remote use, keyword, retrieval, interface (in that order). Therefore the set of initial training instances can be represented as:

Initial Training Instances

y	y	y	n	n	(+)
n	n	n	y	y	(+)
n	y	n	n	n	(-)
n	n	n	n	y	(-)

If a document contained a particular keyword in the keyword list, its attribute value was labeled 'y' ('yes'), otherwise the value was 'n' ('no'). Based on the set of training instances, ID3 first computed the entropy value when adopting "thesaurus" (the first keyword obtained from the desired documents). It then computed the entropy values when adopting other positive keywords. The "thesaurus" keyword produced the most entropy reduction and was thus selected as the first decision node. Following the same computation, "retrieval" was selected as the next (and last) decision node. ID3 constructed the decision tree shown in Figure 1. In the figure, [x,y] means x instances were in the negative class and y instances were in the positive class. The decision tree in Figure 1 can be represented as production rules: (1) IF a document has "thesaurus" as a keyword THEN it is desired (one +, the rightmost branch); (2) IF a document does not have "thesaurus" as a keyword, but has "retrieval" THEN it is also a desired document (one +, the middle branch); (3) IF a document does not have "thesaurus" or "retrieval" as a keyword THEN it is an undesired document (two –, the leftmost branch).

Figure 1: Initial tree created for an IR example

Based on this decision tree, the system searched the database for similar documents and identified three more documents as presented below:

013	modeling, thesaurus, terrorism (+)
014	modeling, simulation, thesaurus, terrorism (+)
018	keyword, thesaurus (+)

These documents were then presented to the user, who provided feedback as to whether or not they were desired. If the user confirmed that document 018 was desired but rejected documents 013 and 014, ID5R used the new (contradictory) evidence to update its current tree. The new training instances for ID5R were:

New Training Instances

y	n	n	n	n	(-)
y	n	n	n	n	(-)
y	n	y	n	n	(+)

The system produced a new tree as shown in Figure 2. This new tree looked different from the original one and can be summarized by the following rules: (1) IF a document has "keyword" as a keyword THEN it is desired (two +, the rightmost branch); (2) IF a document does not have "keyword" as a keyword, but has "retrieval" THEN it is also a desired document (one +, the middle branch); (3) IF a document does not have "keyword" or "retrieval" as a keyword THEN it is an undesired document (four −, the leftmost branch). The whole process was repeated until the entire database was traversed. For this particular example, the final decision tree was the same as the one shown in Figure 2.

Figure 2: Updated tree after relevance feedback

In order to determine how ID5R performed during the user relevance feedback process we examined its *recall* at each point of relevance feedback and compared its performance with that of ID3. ID3 used only the initial document feedback from the users to construct a decision tree and used the tree to search the database. ID5R, on the other hand, collected new evidence during each iteration and updated its trees accordingly. The *recall* measure was defined as:

$$Recall = \frac{\text{Number of relevant records retrieved}}{\text{Total number of relevant records in database}}$$

We developed a test database of about 1000 documents from the 1992 COMPEN CD-ROM collection of computing literature. We then identified 10 research topics, each of which had between five and 20 relevant documents in the database (manually identified). The testing was conducted by comparing the recall of the ID3 algorithm and that of the ID5R incremental approach using the 10 research topics.

Detailed results of the experiment are presented in Table 3. ID5R and ID3 achieved the same levels of performance for five of the ten test cases (Cases 3 and 6-9). After we examined these cases carefully, we found that the initial documents presented for these cases had very precise keywords assigned to them. New instances provided during relevance feedback were consistent with the initial documents, thus ID5R did not revise its decision tree. (At each interaction, ID5R searched only a portion of the entire database. The trees constructed by ID3 remained constant because ID3 did not have any interaction with its users. However, in order to compare its results with those of the ID5R fairly, ID3's performance at each interaction was computed based on the same documents visited by ID5R. As more documents were examined, ID3's classification results may also have improved.)

Case	Int. 1 ID3/ID5R	Int. 2 ID3/ID5R	Int. 3 ID3/ID5R	Int. 4 ID3/ID5R	Int. 5 ID3/ID5R	Int. 6 ID3/ID5R	Int. 7 ID3/ID5R	Int. 8 ID3/ID5R	Target
1	1/1	1/2	2/3	5/6	6/9				10
2	0/0	0/1	0/2	1/4	1/5	2/7	2/8	3/10	11
3	1/1	2/2	3/3	4/4					4
4	1/1	1/1	1/2	1/3	2/4	5/7			10
5	0/0	0/1	0/2	3/5					6
6	1/1	2/2	5/5	6/6					6
7	1/1	2/2	3/3	5/5					5
8	2/2	3/3	3/3	6/6	7/7				8
9	5/5	7/7	8/8	9/9	10/10	11/11			12
10	1/1	2/2	3/3	4/4	7/7	7/10			10
Avg. hits	1.3/1.3	2/2.3	2.8/3.4	4.4/5.2	5.1/6.2	5.6/7.1	5.6/7.2	5.7/7.4	8.2
Avg. recall	16.0/16.0	16.5/31.2	35.0/40.1	55.5/64.1	66.3/79.3	74.0/90.4	74.0/91.3	74.9/93.1	

Table 3: Results of ID3 and ID5R testing

For the other five test cases, ID5R's performance increased gradually until it reached 93.1%. ID3 had been able to reach 74.9%. These research topics tended to have more diverse keywords in the initial documents provided. ID5R appeared to benefit from incremental query tree revision based on the relevance feedback information provided by users. In all 10 cases, ID5R was able to terminate in eight interactions. The response times were often less than a second for each decision-tree building process.

In conclusion, the symbolic ID3 algorithm and its ID5R variant both were shown to be promising techniques for inductive document retrieval. By using the entropy concept in selecting keywords, both algorithms were able to create minimal and understandable decision trees efficiently. However, ID5R's incremental learning and relevance feedback capabilities made it more robust and appealing for large-scale, real-time IR applications.

5 Genetic Algorithms for IR

Often compared with the neural networks and the symbolic learning methods, the self-adpativeness property of genetic algorithms is also extremely appealing for IR applications.

5.1 A Genetic Algorithm: Knowledge Representation and Procedure

Genetic algorithms (GAs) [47] [79] [65] are problem solving systems based on principles of evolution and heredity. A GA maintains a population of individuals, $P(t)$

$= x_1, \ldots, x_n$ at iteration t. Each individual represents a potential solution to the problem at hand and is implemented as some (possibly complex) data structure S. Each solution x_i is evaluated to give some measure of *fitness*. Then a new population at iteration $t + 1$ is formed by selecting the fitter individuals. Some members of the new population undergo transformation by means of genetic operators to form new solutions. There are unary transformations m_i (mutation type), which create new individuals by a small change in a single individual and higher order transformations c_j (crossover type), which create new individuals by combining parts from several (two or more) individuals. For example, if parents are represented by a five-dimensional vector $(a_1, a_2, a_3, a_4, a_5)$ and $(b_1, b_2, b_3, b_4, b_5)$, then a crossover of chromosomes after the second gene produces offspring $(a_1, a_2, b_3, b_4, b_5)$ and $(b_1, b_2, a_3, a_4, a_5)$. The control parameters for genetic operators (probability of crossover and mutation) need to be carefully selected to provide better performance. The intuition behind the crossover operation is information exchange between different potential solutions. After some number of generations the program converges - the best individual hopefully represents the optimum solution. Michalewicz provided an excellent algorithmic discussion of GAs in [79]. In [47] [48], Goldberg presented a good summary of many recent GA applications in biology, computer science, engineering, operations research, physical sciences, and social sciences.

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 was executed in the following steps:

1. **Initialize population and evaluate fitness:**

   To initialize a population, we needed first to decide the number of genes for each individual and the total number of chromosomes (*popsize*) in the initial population. When adopting GAs in IR, each gene (bit) in the chromosome (bit string) represents a certain keyword or concept. The loci (locations of a certain gene) decide the existence (1, ON) or nonexistence (0, OFF) of a concept. A chromosome therefore represents a document which consists of multiple concepts. The initial population contains a set of documents which were judged relevant by a searcher through relevance feedback. 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). An evaluation function for the *fitness* of each chromosome was selected based on Jaccard's score matching function as used by Gordon for document indexing [49]. The Jaccard's score between two sets, X and Y, was computed as:

   $$\#(X \bigcap Y)/\#(X \bigcup Y)$$

   where $\#(S)$ indicated the cardinality of set S. The Jaccard's score is a common measure of association in information retrieval [121].

2. **Reproduction (Selection):**

   Reproduction is the selection of a new population with respect to the probability distribution based on the the fitness values. Fitter individuals have better chances of being selected for reproduction [79]. 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* times; each time we selected a single chromosome for a new population. Obviously, some chromosomes were selected more than once. This is in accordance with the genetic inheritance: the best chromosomes get more copies, the average stay even, and the worst die off.

3. **Recombination (Crossover and Mutation):**

   Now we were ready to apply the first recombination operator, crossover, to the individuals in the new population. The probability of crossover, p_c, gave us the expected number $p_c \times popsize$ of chromosomes which should undergo the crossover operation. For each chromosome, we generated a random number r between 0 and 1; if $r < p_c$, then the chromosome was selected for crossover. We then mated selected pairs of chromosomes randomly: for each pair of coupled chromosomes we generated a random number *pos* from the range of $(1..m - 1)$, where m was the total number of genes in a chromosome. The number *pos* indicated the position of the crossing point. The coupled chromosomes exchanged genes at the crossing point as described earlier.

   The next recombination operator, mutation, was performed on a bit-by-bit basis. The probability of mutation, p_m, gave us the expected number of mutated bits $p_m \times m \times popsize$. Every bit in all chromosomes of the whole population had an equal chance to undergo mutation, i.e., change from 0 to 1 or vice versa. For each chromosome in the crossovered population and for each bit within the chromosome, we generated a random number r from the range of $(0..1)$; if $r < p_m$, we mutated the bit. Typical p_c selected ranged between 0.7 and 0.9 and p_m ranged between 0.01 and 0.03.

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 or converged (i.e., showed no improvement in the overall fitness of the population).

5.2 A GA Example

We present a sample session, implementation details, and some benchmark testing results below. In our system a keyword represented a gene (bit) in GAs; a user-selected document represented a chromosome (individual); and a set of user-selected documents represented the initial population.

The keywords used in the set of user-selected documents were first identified to represent the underlying bit strings for the initial population. Each bit represented the same unique keyword throughout the complete GA process. When a keyword was present in a document, the bit was set to 1, otherwise 0. Each document could then be represented in terms of a sequence of 0s and 1s. The keywords of five user-selected documents are presented below. The set of unique concepts present in these sample documents is also summarized – 33 keywords (genes) in total. As in the Hopfield network example, some concepts were folder names assigned by the users (in the format of *.*), e.g., QUERY.OPT folder for query optimization topics.

Input Documents and Keywords

DOC0	DATA RETRIEVAL, DATABASE, COMPUTER NETWORKS, IMPROVEMENTS, INFORMATION RETRIEVAL, METHOD, NETWORK, MULTIPLE, QUERY, RELATION, RELATIONAL, RETRIEVAL, QUERIES, RELATIONAL DATABASES, RELATIONAL DATABASE, US, CARAT.DAT, GQP.DAT,ORUS.DAT, QUERY.OPT
DOC1	INFORMATION, INFORMATION RETRIEVAL, INFORMATION STORAGE, INDEXING, RETRIEVAL, STORAGE, US, KEVIN.HOT
DOC2	ARTIFICIAL INTELLIGENCE, INFORMATION RETRIEVAL SYSTEMS, INFORMATION RETRIEVAL, INDEXING, NATURAL LANGUAGE PROCESSING, US, DBMS.AI, GQP.DAT
DOC3	FUZZY SET THEORY, INFORMATION RETRIEVAL SYSTEMS, INDEXING, PERFORMANCE, RETRIEVAL SYSTEMS, RETRIEVAL, QUERIES, US, KEVIN.HOT
DOC4	INFORMATION RETRIEVAL SYSTEMS, INDEXING, RETRIEVAL, STAIRS, US, KEVIN.HOT

Total Set of Concepts

DATA RETRIEVAL, DATABASE, COMPUTER NETWORKS, IMPROVEMENTS, INFORMATION RETRIEVAL, METHOD, NETWORK, MULTIPLE, QUERY, RELATION, RELATIONAL, RETRIEVAL, QUERIES, RELATIONAL DATABASES, RELATIONAL DATABASE, US, CARAT.DAT, GQP.DAT, ORUS.DAT, QUERY.OPT, INFORMATION, INFORMATION STORAGE, INDEXING, STORAGE, KEVIN.HOT, ARTIFICIAL INTELLIGENCE, INFORMATION RETRIEVAL SYSTEMS, NATURAL LANGUAGE PROCESSING, DBMS.AI, FUZZY SET THEORY, PERFORMANCE, RETRIEVAL SYSTEMS, STAIRS,

Initial Genetic Pattern of Chromosome in Population

chromosome	fitness
111111111111111111110000000000000	[0.287744]
000010000001000100001111100000000	[0.411692]

```
0000100000000000101000010011110000          [0.367556]
0000000000011001000000010101001110          [0.427473]
0000000000010001000000010101000001          [0.451212]

    Average Fitness = 0.3891
```

We computed the fitness of each document based on its relevance to the documents in the user-selected set. Higher Jaccard's score (a value between 0 and 1) indicated stronger relevance between two documents. For document 0, we computed five different Jaccard's scores between document 0 and documents 0, 1, 2, 3, and 4, respectively (shown below). An average fitness was then computed for document 0 (0.28774). The same procedure was applied to other documents to compute their fitness. A document which included more concepts shared by other documents had a higher Jaccard's score.

```
  Jaccard's Score of DOC0 and DOC0      = 1.000000
  Jaccard's Score of DOC0 and DOC1      = 0.120000
  Jaccard's Score of DOC0 and DOC2      = 0.120000
  Jaccard's Score of DOC0 and DOC3      = 0.115384
  Jaccard's Score of DOC0 and DOC4      = 0.083333

Average Fitness (Jaccard's Score) of Document0 : 0.28774
```

If a user provided documents which are closely related, the average fitness for the complete document set was high. If the user-selected documents were only loosely related, their overall fitness was low. Generally, GAs did a good job optimizing a document set which was initially low in fitness. Using the previous example, the overall Jaccard's score increased over generations. The optimized population contained only one single chromosome, with an average fitness value of 0.45121. The optimized chromosome contained six relevant keywords which best described the initial set of documents. Using these "optimized" keywords, an information retrieval system could proceed to suggest relevant documents to users. The user-GA interaction continued until a search was completed or the user decided to stop.

```
.___________________ Optimized Chromosomes in the Population ___________________
                    chromosome                         fitness
        0000000000010001000000010101000001           [0.45121]
        0000000000010001000000010101000001           [0.45121]
        0000000000010001000000010101000001           [0.45121]
        0000000000010001000000010101000001           [0.45121]
        0000000000010001000000010101000001           [0.45121]

            Average Fitness = 0.4512

.__________________ Derived Concepts from Optimized Population __________________

  RETRIEVAL, US, INDEXING, KEVIN.HOT, INFORMATION RETRIEVAL SYSTEMS, STAIRS,
```

Table 4 summarizes the results of a benchmark testing. In the testing we randomly retrieved five test cases of 1-document, 2-document, 3-document, 4-document, 5-document, and 10-document examples, respectively, from the 3000-document DIALOG-extracted database discussed earlier. There were 30 test cases in total. For each test case, an initial fitness based on the Jaccard's score was computed. For 1-document and 2-document test cases, their initial fitness tended to be higher due to the smaller sample size (see Column 2 of Table 4). In Table 4 we also report performance measures in terms of the Jaccard's scores for the GA processes, the CPU times, and the average improvements in fitness.

Using the GA optimization process, our system achieved an average fitness improvement from 5.38% to 17.7%. This improvement was slightly worse than the performance improvement for indexing reported by Gordon [49]. An interesting observation was that when more initial documents were present, the initial fitness tended to be lower, which allowed the system to do a better job in improving the preciseness of the initial keywords and in identifying other relevant documents. As shown in Table 4, fitness improvement increased as a function of the number of initial documents. This finding also suggested that when initial user-supplied documents are fuzzy and not well articulated, GAs may be able to make a more significant contribution in suggesting other relevant documents. This could be quite important for complex information retrieval sessions during which searchers need help in query articulation and search refinement.

The number of documents suggested by GANNET after the first GA process was between 9 and 13, with an average of about 11 documents. The CPU times required of the GA process also was quite reasonable, with an average of 0.168 seconds. The response times were significantly better than the Hopfield net activation. In conclusion, by using reproduction and the genetic operators, GAs provided an interesting system-aided way of analyzing users' intermediate search results and suggesting other potentially relevant documents.

6 Conclusion and Future Directions

Information retrieval research has been advancing very quickly over the past few decades. Researchers have experimented with techniques ranging from probabilistic models and the vector space model to the knowledge-based approach and the recent machine learning techniques. At each stage, significant insights regarding how to design more useful and "intelligent" information retrieval systems have been gained.

In this paper we present an extensive review of IR research that was mainly based on machine learning techniques. Connectionist modeling and learning, in particular, has attracted considerable attention due to its strong resemblance to some existing IR models and techniques. Symbolic machine learning and genetic algorithms, two

No.	*Init. Score*	*GA Score*	Impr. %	*CPU (sec.)*	*Doc. Selected*
1	1.0	1.0	0.0	0.067	7
2	1.0	1.0	0.0	0.05	25
3	1.0	1.0	0.0	0.067	7
4	1.0	1.0	0.0	0.05	9
5	1.0	1.0	0.0	0.067	5
1 doc.	*avg.*	1.0	0.0	0.06	10.6
1	0.5139	0.5139	0.0	0.083	10
2	0.5833	0.5833	0.0	0.1	8
3	0.6111	0.6111	0.0	0.083	5
4	0.6486	0.6486	0.0	0.067	10
5	0.7857	0.7857	0.0	0.083	16
2 doc.	*avg.*	0.6285	0.0	0.08	9.8
1	0.3841	0.3984	3.72	0.023	8
2	0.4157	0.4360	4.88	0.1	5
3	0.4286	0.4611	7.1	0.1	13
4	0.5032	0.5215	3.6	0.133	5
5	0.5899	0.6349	7.6	0.083	16
3 doc.	*avg.*	0.4904	5.38	0.088	9.4
1	0.2898	0.3010	3.8	0.117	22
2	0.3078	0.3142	2.1	0.1	15
3	0.3194	0.3495	9.4	0.283	5
4	0.3319	0.3442	3.7	0.25	11
5	0.4409	0.5060	14.7	0.25	10
4 doc.	*avg.*	0.3629	6.74	0.2	12.6
1	0.3048	0.3370	10.5	0.4	12
2	0.3068	0.3267	6.4	0.15	7
3	0.3194	0.3575	11.9	0.52	5
4	0.4655	0.5671	21.8	0.3	21
5	0.6181	0.7171	16.0	0.12	21
5 doc.	*avg.*	0.4610	13.32	0.298	13.2
1	0.2489	0.2824	13.5	0.32	18
2	0.2038	0.2282	12.9	0.35	8
3	0.2016	0.2343	16.2	0.47	6
4	0.4997	0.6201	24.1	0.13	5
5	0.3727	0.4540	21.8	0.13	11
10 doc.	*avg.*	0.3638	17.7	0.28	9.6
ALL	*AVG.*	0.5511	7.19	0.168	10.87

Table 4: Results of genetic algorithms testing

popular candidates for adaptive learning in other applications, on the other hand, have been used only rarely. However, these newer techniques have been found to exhibit promising inductive learning capabilities for selected IR applications.

For researchers who are interested in examining these techniques, this paper has discussed an algorithmic approach and knowledge representations appropriate for IR. We feel that the proper selection of knowledge representation and the adaptation of machine learning algorithms in the IR context are essential to the successful use of such techniques. For example, in IR a keyword could represent a node in the Hopfield net, a single bit in a genetic algorithm, or a decision node in ID3 and ID5R. Similarly, the *parallel relaxation* search of the Hopfield net, the *entropy reduction* scheme in ID3, and the *Darwinian selection* of genetic algorithms all need to be carefully studied and modified in the unique IR context.

Despite of some initial successful application 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. We briefly review several important research directions below:

- *Limitations of learning techniques for IR:* The performance of the inductive learning techniques relies strongly on the examples provided (as in any other statistical and classification techniques) [124]. In IR, these examples may include user-provided queries and documents collected during relevance feedback. The importance of sample size has been stressed heavily, even in the probabilistic models [42] [44]. In reality, user-provided relevance feedback information may be limited in quantity and noisy (i.e., contradictory or incorrect), which may have adverse effects for the IR or indexing tasks. Some learning techniques such as the neural networks approach have documented noise-resistent capability, but empirical evidence and research need to be performed to verify this characteristic in the context of IR and indexing. In our preliminary investigation, all three machine learning algorithms performed satisfactorily for small document samples, but the effect of the sample size needs to be examined more carefully.

  For large-scale real-life applications, neural networks and, to some extent, genetic algorithms may suffer from requiring extensive computation time and lack of interpretable results. Symbolic learning, on the other hand, efficiently produces simple production rules or decision tree representations. The effects of the representations on the cognition of searchers in the real-life retrieval environments (e.g., users' acceptance of the analytical results provided by an intelligent system) remain to be determined.

- *Applicability to the full-text retrieval environment:* In addition to extensive IR research conducted in probabilistic models, knowledge-based systems, and machine learning, significant efforts have also been made by many commercial companies in pursuit of more effective and "intelligent" information retrieval systems. In an attempt to understand the potential role of machine learning in commercial full-text retrieval systems, we examined several major full-text retrieval software packages on the market, including: BRS/SEARCH[1], BASIS/Plus[2], PixTex[3], and Topic[4].

Most full-text retrieval software has been designed to handle large volumes of text by indexing every word (and its position). This allows users to perform proximity search, morphological search (using prefix, suffix, or wildcards), and thesaurus search. BRS/SEARCH and BASIS/plus are typical of this type of software. PixTex and Topic, on the other hand, are among the most advanced full-text retrieval systems and feature "content-based IR" and "learning" capabilities. PixTex calls its indexing process "learning." The system automatically extracts patterns from binary data (texts or images) and associates (or "learns") the storage location of the data based on neural network technology (the exact form and algorithm are not clear due to the lack of publications and the proprietary nature of the product). By automatically storing visual scene or textual contents in terms of Huffman codes, the system can then retrieve other similar scene objects or texts during IR. Verity's Topic claims to use fuzzy logic in its design of "conceptual searching" for "intelligent" document retrieval systems. It allows users to create and re-use hierarchical, weighted query trees (thus becoming part of the *corporate memory*), which produce rank-ordered documents. It also appears to have some "similarity search" capability (e.g., 'find me all documents like this one.'). However, like PixTex, no algorithmic detail can be obtained. Despite the lack of implementation detail, we believe that with the extensive indexing capabilities provided by such full-text software, a simple user relevance feedback component and inductive machine learning algorithms, similar to the ones discussed in this research, could be incorporated to help identify what users want, based on the concepts (keywords) learned from the sample documents. As more researchers and practitioners recognize the need for concept-based and "intelligent" IR, application of machine learning algorithms presents unique challenges and opportunities.

We believe this research has shed light on the feasibility and usefulness of the newer, AI-based machine learning algorithms for IR. However, more extensive and systematic studies of various system parameters and for large-scale, real-life applications are needed. We hope by incorporating into IR inductive learning capabilities, which are

[1]Vended by BRS Software Products, McLean, Virginia, USA.
[2]Vended by Information Dimensions Inc., Dublin, Ohio, USA.
[3]Vended by Excalibur Technologies Corp., McLean, Virginia, USA.
[4]Vended by Verity, Inc., Mountain View, California, USA.

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 retrieval systems.

7 Acknowledgments

This project was supported mainly by NSF grant # IRI-9211418, 1992-1994 (NSF/CISE, Division of Information, Robotics, and Intelligent Systems).

References

[1] D. Appelt. The role of user modelling in language generation and communication planning. In *User Modelling Panel, Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 1298–1302, Los Angeles, CA, August 1985.

[2] R. K. Belew. Adaptive information retrieval. In *Proceedings of the Twelth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 11–20, Cambridge, MA, June 25-28, 1989.

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

[4] M. J. Blosseville, G. Hebrail, M. G. Monteil, and N. Penot. Automatic document classification: natural language processing, statistical analysis, and expert system techniques used together. In *Proceedings of the Fifteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 51–57, Copenhagen, Denmark, June 21-24 1992.

[5] L. B. Booker, D. E. Goldberg, and J. H. Holland. Classifier systems and genetic algorithms. In *Machine Learning, Paradigms and Methods*, Pages 235-282, Carbonell, J. G., Editor, The MIT Press, Cambridge, MA, 1990.

[6] A. Bookstein and D. R. Swanson. Probabilistic models for automatic indexing. *Journal of the American Society for Information Science*, 26(1):45–50, January-February 1975.

[7] A. Borgida and K. E. Williamson. Accommodating exceptions in a database, and refining the schema by learning from them. In *Proceedings of the 11th International VLDB Conference*, pages 72–81, Stockholm, August 1985.

[8] G. Brajnik, G. Guida, and C. Tasso. IR-NLI II: Applying man-machine interaction and artificial intelligence concepts to information retrieval. In *Proceedings of the Eleventh Annual International ACMSIGIR Conference on Research and Development in Information Retrieval*, pages 387–399, Grenoble, France, 1988.

[9] T. L. Brauen. Document vector modification. In *The Smart Retrieval System – Experiments in Automatic Document Processing*, pages 456–484, G. Salton, Editor, Prentice-Hall Inc., Englewood Cliffs, NJ, 1971.

[10] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Tree.* Wadsworth, Monterey, CA, 1984.

[11] M. K. Buckland and D. Florian. Expertise, task complexity, and artificial intelligence: A conceptual framework. *Journal of the American Society for Information Science*, 42(9):635–643, October 1991.

[12] Y. Cai, N. Cercone, and J. Han. Attribute-oriented induction in relational databases. In *Knowledge Discovery in Databases*, pages 213–228, G. Piatetsky-Shapiro and W. J. Frawley, Editors, The MIT Press, Cambridge, MA, 1991.

[13] J. G. Carbonell, R. S. Michalski, and T. M. Mitchell. An overview of machine learning. In *Machine Learning, An Artificial Intelligence Approach*, Pages 3-23, Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., Editors, Tioga Publishing Company, Palo Alto, CA, 1983.

[14] H. Chen, K. Basu, and T. Ng. An algorithmic approach to concept exploration in a large knowledge network (automatic thesaurus consultation): symbolic branch-and-bound vs. connectionist Hopfield net activation. In *Journal of the American Society for Information Science*, 1994, forthcoming.

[15] H. Chen, P. Buntin, L. She, S. Sutjahjo, C. Sommer, and D. Neely. Expert prediction, symbolic learning, and neural networks: An experiment on greyhound racing. *IEEE EXPERT*, forthcoming, 1994.

[16] H. Chen and V. Dhar. Reducing indeterminism in consultation: a cognitive model of user/librarian interaction. In *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-87)*, pages 285–289, Seattle, WA, July 13-17, 1987.

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

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

[19] H. Chen, P. Hsu, R. Orwig, L. Hoopes, and J. F. Nunamaker. Automatic concept classification of text from electronic meetings. *Communications of the ACM*, 1994, in press.

[20] H. Chen and J. Kim. GANNET: information retrieval using genetics algorithms and neural networks. In *Center for Management of Information, College of Business and Public Administration, University of Arizona, Working Paper, CMI-WPS*, 1993.

[21] H. Chen and K. J. Lynch. Automatic construction of networks of concepts characterizing document databases. *IEEE Transactions on Systems, Man and Cybernetics*, 22(5):885–902, September/October 1992.

[22] H. Chen, K. J. Lynch, K. Basu, and T. Ng. 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, April 1993.

[23] H. Chen and L. She. Inductive query by examples (IQBE): A machine learning approach. In *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.

[24] Y. Chiaramella and B. Defude. A prototype of an intelligent system for information retrieval: IOTA. *Information Processing and Management*, 23(4):285–303, 1987.

[25] P. R. Cohen and R. Kjeldsen. Information retrieval by constrained spreading activation in semantic networks. *Information Processing and Management*, 23(4):255–268, 1987.

[26] S. L. Crawford, R. Fung, L. A. Appelbaum, and R. M. Tong. Classification trees for information retrieval. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 245–249, Morgan Kaufmann, 1991.

[27] S. L. Crawford and R. M. Fung. An analysis of two probablistic model induction techniques. *Statistics and Computing*, 2(2):83–90, June 1992.

[28] W. B. Croft and R. H. Thompson. I^3R: A new approach to the design of document retrieval systems. *Journal of the American Society for Information Science*, 38(6):389–404, 1987.

[29] J. Dalton and A. Deshmane. Artificial neural networks. *IEEE Potentials*, 10(2):33–36, April 1991.

[30] P. J. Daniels. *The User Modelling Function of an Intelligent Interface for Document Retrieval Systems.* In B.C. Brookes (Ed.), Intelligent Information Systems for the Information Society, Elsevier Science Publishers B.V., North-Holland, Amsterdam, 1986.

[31] M. Derthick. *Mundane Reasoning by Parallel Constraint Satisfaction.* Carnegie Mellon University, Pittsburgh, PA, Ph.D. thesis, 1988.

[32] T. E. Doszkocs, J. Reggia, and X. Lin. Connectionist models and information retrieval. *Annual Review of Information Science and Technology (ARIST)*, 25:209–260, 1990.

[33] B. Everitt. *Cluster Analysis.* Second Edition, Heinemann Educational Books, London, England, 1980.

[34] D. H. Fisher and K. B. McKusick. An empirical comparison of ID3 and back-propagation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 788–793, Detroit, MI, August 20-25 1989.

[35] D. B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, January 1994.

[36] L. J. Fogel. Autonomous automata. *Industrial Research*, 4, 1962.

[37] L. J. Fogel. *On the Organization of Intellect.* Doctoral Dissertation, UCLA, Los Angeles, CA, 1964.

[38] E. A. Fox. Development of the CODER system: A testbed for artificial intelligence methods in information retrieval. *Information Processing and Management*, 23(4):341–366, 1987.

[39] W. J. Frawley, G. Pietetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases: an overview. In *Knowledge Discovery in Databases*, pages 1–30, G. Piatetsky-Shapiro and W. J. Frawley, Editors, The MIT Press, Cambridge, MA, 1991.

[40] J. E. Freund. *Mathematical Statistics.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1971.

[41] O. Frieder and H. T. Siegelmann. On the allocation of documents in multiprocessor information retrieval systems. In *Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 230–239, Chicago, IL, October 13-16 1991.

[42] N. Fuhr and C. Buckley. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems*, 9(3):223–248, July 1991.

[43] N. Fuhr, S. Hartmann, G. Knorz, G. Lustig, M. Schwantner, and K. Tzeras. AIR/X – a rule-based multistage indexing system for large subject fields. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 789–795, Boston, MA, July 29-August 3 1990.

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

[45] R. Fung and S. L. Crawford. Constructor: a system for the induction of probablistic models. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-90)*, pages 762–769, Boston, MA, July 29-August 3, 1990.

[46] S. I. Gallant. Connectionist expert system. *Communications of the ACM*, 31(2):152–169, 1988.

[47] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[48] D. E. Goldberg. Genetic and evolutionary algorithms come of age. *Communications of the ACM*, 37(3):113–119, March 1994.

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

[50] M. D. Gordon. User-based document clustering by redescribing subject descriptions with a genetic algorithm. *Journal of the American Society for Information Science*, 42(5):311–322, June 1991.

[51] D. P. Greene and S. F. Smith. COGIN: symbolic induction with genetic algorithms. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 111–116, San Jose, CA, July 12-16 1992.

[52] L. O. Hall and S. G. Romaniuk. A hybrid connectionist, symbolic learning system. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 783–788, Boston, MA, July 29-August 3 1990.

[53] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(1):29–40, February 1993.

[54] S. Harp, T. Samad, and A. Guha. Towards the genetic synthesis of neural networks. In *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1989.

[55] F. Hayes-Roth and N. Jacobstein. The state of knowledge-based systems. *Communications of the ACM*, 37(3):27–39, March 1994.

[56] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[57] J. J. Hopfield. Neural network and physical systems with collective computational abilities. *Proceedings of the National Academy of Science, USA*, 78(8), 1982.

[58] B. L. Humphreys and D. A. Lindberg. Building the unified medical language system. In *Proceedings of the Thirteenth Annual Symposium on Computer Applications in Medical Care*, Washington, DC: IEEE Computer Society Press, November, 5-8 1989.

[59] E. Ide. New experiments in relevance feedback. In *The Smart Retrieval System – Experiments in Automatic Document Processing*, pages 337–354, G. Salton, Editor, Prentice-Hall Inc., Englewood Cliffs, NJ, 1971.

[60] E. Ide and G. Salton. Inteactive search strategies and dynamic file organization in information retrieval. In *The Smart Retrieval System – Experiments in Automatic Document Processing*, pages 373–393, G. Salton, Editor, Prentice-Hall Inc., Englewood Cliffs, NJ, 1971.

[61] Y. E. Ioannidis, T. Saulys, and A. J. Whitsitt. Conceptual learning in database design. *ACM Transactions on Information Systems*, 10(3):265–293, July 1992.

[62] H. Kitano. Empirical studies on the speed of convergence of neural network training using genetic algorithms. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 789–795, Boston, MA, July 29-August 3 1990.

[63] K. Knight. Connectionist ideas and algorithms. *Communications of the ACM*, 33(11):59–74, November 1990.

[64] T. Kohonen. *Self-Organization and Associative Memory.* Third Edition, Springer-Verlag, Berlin Heidelberg, 1989.

[65] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* The MIT Press, Cambridge, MA, 1992.

[66] K. L. Kwok. A neural network for probablistic information retrieval. In *Proceedings of the Twelfth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 21–30, Cambridge, MA, June 25-28, 1989.

[67] M. Lebowitz. Concept learning in a rich input domain: Generalization-based memory. In *Machine Learning, An Artificial Intelligence Approach, Vol. II*, pages 193–214, Pages 463-482, Carbonell, J. G., Michalski, R. S., and Mitchell, T. M., Editors, Morgan Faufmann, Los Altos, CA, 1987.

[68] D. D. Lewis. Learning in intelligent information retrieval. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 235–239, Morgan Kaufmann, 1991.

[69] D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the Fifteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 37–50, Copenhagen, Denmark, June 21-24 1992.

[70] Q. Li and D. McLeod. Object flavor evolution through learning in an object-oriented database system. In *Expert Database Systems, Proceedings from the Second International Conference*, pages 469–495, Benjamin/Cummings, Menlo Park, CA, L. Kerschberg, Ed. 1989.

[71] X. Lin, D. Soergel, and G. Marchionini. A self-organizing semantic map for information retrieval. In *Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 262–269, Chicago, IL, October 13-16 1991.

[72] D. A. Lindberg and B. L. Humphreys. The UMLS knowledge sources: Tools for building better user interface. In *Proceedings of the Fourteenth Annual Symposium on Computer Applications in Medical Care*, Los Alamitos, CA: Institute of Electrical and Electronics Engineers, November, 4-7 1990.

[73] R. P. Lippmann. An introduction to computing with neural networks. *IEEE Acoustics Speech and Signal Processing Magazine*, 4(2):4–22, April 1987.

[74] K. J. MacLeod and W. Robertson. A neural algorithm for document clustering. *Information Processing & Management*, 27(4):337–346, 1991.

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

[76] B. K. Martin and R. Rada. Building a relational data base for a physician document index. *Med. Inf.*, 12(3):187–201, July-September 1987.

[77] B. Masand, L. Gordon, and D. Waltz. Classifying news stories using memory-based reasoning. In *Proceedings of the Fifteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 59–65, Copenhagen, Denmark, June 21-24 1992.

[78] A. T. McCray and W. T. Hole. The scope and structure of the first version of the UMLS semantic network. In *Proceedings of the Fourteenth Annual Symposium on Computer Applications in Medical Care*, Los Alamitos, CA: Institute of Electrical and Electronics Engineers, November, 4-7 1990.

[79] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, Berlin Heidelberg, 1992.

[80] R. S. Michalski. A theory and methodology of inductive learning. In *Machine Learning, An Artificial Intelligence Approach*, Pages 83-134, Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., Editors, Tioga Publishing Company, Palo Alto, CA, 1983.

[81] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982.

[82] I. Monarch and J. G. Carbonell. CoalSORT: A knowledge-based interface. *IEEE EXPERT*, pages 39–53, Spring 1987.

[83] D. J. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 762–767, Detroit, MI, August 20-25 1989.

[84] D. D. Montgomery. *Design and Analysis of Experiments.* John Wiley & Sons, New York, NY, 1976.

[85] R. Mooney, J. Shavlik, G. Towell, and A. Gove. An experimental comparison of symbolic and connectionist learning algorithms. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 775–780, Detroit, MI, August 20-25 1989.

[86] K. Parsaye, M. Chignell, S. Khoshafian, and H. Wong. *Intelligent Databases.* John Wiley & Sons, Inc., New York, NY, 1989.

[87] F. Petry, B. Buckles, D. Prabhu, and D. Kraft. Fuzzy information retrieval using genetic algorithms and relevance feedback. In *Proceedings of the ASIS Annual Meeting*, pages 122–125, 1993.

[88] G. Piatetsky-Shapiro. Workshop on knowledge discovery in real databases. In *International Joint Conference of Artificial Intelligence*, 1989.

[89] S. Pollitt. Cansearch: An expert systems approach to document retrieval. *Information Processing and Management*, 23(2):119–138, 1987.

[90] J. R. Quinlan. Discovering rules by induction from large collections of examples. In *Expert Systems in the Micro-electronic Age*, Pages 168-201, Michie, D., Editor, Edinburgh University Press, Edinburgh, Scotland, 1979.

[91] J. R. Quinlan. Learning efficient classification procedures and their application to chess end games. In *Machine Learning, An Artificial Intelligence Approach*, Pages 463-482, Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., Editors, Tioga Publishing Company, Palo Alto, CA, 1983.

[92] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[93] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kauffmann, Los Altos, CA, 1993.

[94] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1):17–30, January/February 1989.

[95] V. V. Raghavan and B. Agarwal. Optimal determination of user-oriented clusters: An application for the reproductive plan. In *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, pages 241–246, Cambridge, MA, July 1987.

[96] L. F. Rau and P. S. Jacobs. Creating segmented databases from free text for text retrieval. In *Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 337–346, Chicago, IL, October 13-16 1991.

[97] E. Rich. Building and exploiting user models. In *International Joint Conference of Artificial Intelligence, Tokyo, Japan, August*, pages 720–722, 1979.

[98] E. Rich. User modeling via stereotypes. *Cognitive Science*, 3:329–354, 1979.

[99] E. Rich. Users are individuals: Individualizing user models. *International Journal of Man-Machine Studies*, 18(3):199–214, March 1983.

[100] S. E. Roberston and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

[101] J. J. Rocchio. Relevance feedback in information retrieval. In *The Smart Retrieval System – Experiments in Automatic Document Processing*, pages 313–323, G. Salton, Editor, Prentice-Hall Inc., Englewood Cliffs, NJ, 1971.

[102] D. E. Rose and R. K. Belew. A connectionist and symbolic hybrid for improving legal research. *Int. Journal of Man-Machine Studies*, 35(1):1–33, 1991.

[103] D. E. Rumelhart, G. E. Hinton, and J. L. McClelland. A general framework for parallel distributed processing. In *Parallel Distributed Processing*, pages 45–76, D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, Editors, The MIT Press, Cambridge, MA, 1986.

[104] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing*, pages 318–362, D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, Editors, The MIT Press, Cambridge, MA, 1986.

[105] D. E. Rumelhart, B. Widrow, and M. A. Lehr. The basic ideas in neural networks. *Communications of the ACM*, 37(3):87–92, March 1994.

[106] G. Salton. *Automatic Text Processing.* Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.

[107] L. Shastri. Why semantic networks? In *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 109–136, J. F. Sowa, Editor, Morgan Kauffmann Publishers, Inc., San Mateo, CA, 1991.

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

[109] P. K. Simpson. *Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations.* McGraw-Hill Book Company, New York, NY, 1990.

[110] D. Sleeman. UMFE: a user modeling front-end subsystem. *International Journal of Man-Machine Studies*, 23:71–88, 1985.

[111] P. J. Smith, S. J. Shute, D. Galdes, and M. H. Chignell. Knowledge-based search tactics for an intelligent intermediary system. *ACM Transactions on Information Systems*, 7:246–270, 1989.

[112] K. Sparck Jones. The role of artificial intelligence in information retrieval. *Journal of the American Society for Information Science*, 42(8):558–565, September 1991.

[113] R. E. Stepp and R. S. Michalski. Conceptual clustering: Inventing goal-oriented classifications of structured objects. In *Machine Learning, An Artificial Intelligence Approach, Vol. II*, pages 472–498, Pages 463-482, Carbonell, J. G., Michalski, R. S., and Mitchell, T. M., Editors, Morgan Faufmann, Los Altos, CA, 1987.

[114] W. Swartout. Explanation and the role of the user model: how much will it help? In *User Modelling Panel, Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 1298–1302, Los Angeles, CA, August 1985.

[115] D. W. Tank and J. J. Hopfield. Collective computation in neuronlike circuits. *Scientific American*, 257(6):104–114, December 1987.

[116] D. Touretzky and G. E. Hinton. A distributed connectionist production system. *Cognitive Science*, 12(3):423–466, 1988.

[117] H. Turtle and W. B. Croft. Inference networks for document retrieval. In *Proceedings of the 13th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 1–24, Brussels, Belgium, September 5-7, 1990.

[118] H. Turtle and W. B. Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222, July 1991.

[119] K. Tzeras and S. Hartmann. Automatic indexing based on Bayesian inference networks. In *Proceedings of the 16th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 22–34, Pittsburgh, PA, June 27-July 1, 1993.

[120] P. E. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4:161–186, 1989.

[121] C. J. Van Rijsbergen. *Information Retrieval.* 2nd edition, Butterworths, London, 1979.

[122] A. Vickery and H. M. Brooks. PLEXUS - the expert system for referral. *Information Processing and Management*, 23(2):99–117, 1987.

[123] S. M. Weiss and I. Kapouleas. An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 781–787, Detroit, MI, August 20-25 1989.

[124] S. M. Weiss and C. A. Kulikowski. *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Networks, Machine Learning, and Expert Systems.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[125] B. Widrow, D. E. Rumelhart, and M. A. Lehr. Neural networks: applications in industry, business, and science. *Communications of the ACM*, 37(3):93–105, March 1994.

[126] R. Wilkinson and P. Hingston. Using the Cosine measure in neural network for document retrieval. In *Proceedings of the Fourteenth Annual International*

ACM/SIGIR Conference on Research and Development in Information Retrieval, pages 202–210, Chicago, IL, October 13-16 1991.

[127] R. Wilkinson, P. Hingston, and T. Osborn. Incorporating the vector space model in a neural network used for document retrieval. *Library Hi Tech*, 10(12):69–75, 1992.

[128] J. Yang and R. R. Korfhage. Effects of query term weights modification in document retrieval: a study based on a genetic algorithm. In *Proceedings of the Second Annual Symposium on Document Analysis and Information Retrieval*, pages 271–285, Las Vegas, NV, April 26-28 1993.

[129] J. Yang, R. R. Korfhage, and E. Rasmussen. Query improvement in information retrieval using genetic algorithms: a report on the experiments of the TREC project. In *Text Retrieval Conference (TREC-1)*, pages 31–58, Gaithersburg, MD, November 4-6 1993.

[130] C. T. Yu and G. Salton. Precision weighting: an effective automatic indexing method. *Journal of the ACM*, 23:76–88, 1976.

[131] A. Y. Zissos and I. H. Witten. User modeling for a computer coach: a case study. *International Journal of Man-Machine Studies*, 23:729–750, 1985.