Contents

List of Figures

List of Tables

Inductive Query by Examples (IQBE): A Machine Learning Approach

Hsinchun Chen[1] and Linlin She[2].

Abstract:

This paper presents an incremental, inductive learning approach to query-by-examples for information retrieval (IR) and database management systems (DBMS). After briefly reviewing conventional information retrieval techniques and the prevailing database query paradigms, we introduce the ID5R algorithm, previously developed by Utgoff, for ``intelligent'' and system-supported query processing.

We describe in detail how we adapted the ID5R algorithm for IR/DBMS applications and we present two examples, one for IR applications and the other for DBMS applications, to demonstrate the feasibility of the approach.

Using a larger test collection of about 1000 document records from the COMPEN CD-ROM computing literature database and using recall as a performance measure, our experiment showed that the incremental ID5R performed significantly better than a batch inductive learning algorithm (called ID3) which we developed earlier. Both algorithms, however, were robust and efficient in helping users develop abstract queries from examples. We believe this research has shed light on the feasibility and the novel characteristics of a new query paradigm, namely, inductive query-by-examples (IQBE). Directions of our current research are summarized at the end of the paper.

Introduction

It has been estimated that the amount of information in the world doubles every 20 months. The number of online databases, in particular, and the amounts of the information reside in them have increased even more dramatically in recent years. The process required to retrieve relevant information from large-scale databases has become problematic and cognitively-demanding [10] [8]. This situation is particularly evident in textual retrieval systems and commercial databases, which are widely used in libraries and bibliographic databases (e.g., books and articles), in the business world (e.g., personnel files, newsletters, and electronic data interchanges), and in scientific applications (e.g., electronic community systems and scientific databases). Only users with extensive subject knowledge, system knowledge, and classification scheme knowledge [9] are able to maneuver and explore in these databases.

Most current information retrieval systems still rely on inverted index and Boolean query techniques, which have obvious disadvantages. One main drawback is that these techniques depend on the users' ability to articulate their needs with precise query syntax. For example, many searchers have difficulties formulating complex Boolean queries. In commercial DBMSs, despite of the availability of high-level declarative relational query languages [16], queries still need to be formulated in precise commands. That is, users need to be able to know exactly what they want and be able to formulate their queries in terms of exact values for the tables, attributes, etc.

However, according to past research, users often do not have ``queries,'' but

what Belkin calls an ``anomalous state of knowledge" [2] [3]. Users often expect to refine this anomalous state into a query through an interactive process. They may not be able to articulate precisely what they want, but they often have little trouble in recognizing instances of the information they desire. Because it is easier for searchers to identify precise records or documents, instead of stating their needs in abstract and conceptual queries, the *relevance feedback* process of information search has been shown to be effective and less cognitively-demanding [31].

In this research we have proposed a novel approach to implementing relevance feedback in IR/DBMS, called Inductive Query by Example (IQBE). Ground on symbolic AI-based machine learning research, this new query paradigm allows users to identify a few records of interest to them and then to use the underlying common characteristics of these records (e.g., keywords, attribute values, etc.) to formulate high-level, abstract queries. This interactive process of relevance feedback from users and inductive learning by the system can help form an intelligent and fruitful human-system partnership in database queries.

Given that database query is by nature a classification problem, all records can be classified as belonging to either a positive class (desired records) or a negative class (undesired records). Quinlan's ID3 decision-tree building algorithm [27] [28] and Utgoff's ID5R algorithm (an incremental version of ID3) [33] are particularly suitable for solving such classification problems. Based on our past experience of adopting ID3 in IR [14], this research examined the feasibility and usefulness of adopting an incremental ID5R for IR/DBMS applications.

The structure of this paper is as follows. Section 2 presents an overview of the prevailing IR/DBMS query techniques and the inductive learning paradigm. Section 3 introduces the IQBE algorithm, mainly based on Quinlan's ID3 and Utgoff's ID5R, Section 4 provides two sample sessions of adopting the IQBE approach in IR and DBMS, respectively. A large-scale evaluation of the performance of this technique using a 1000-document database is shown in Section 5. In Section 6, we present conclusions and suggest future directions.

Literature Review

Information Retrieval (IR) and Database Management Systems (DBMS) Query Techniques

Most commercial information retrieval systems still rely on conventional inverted index and Boolean querying techniques. Even full-text retrieval produced less than satisfactory results [4]. In the past few decades, probabilistic retrieval techniques have been used to improve the retrieval performance of information retrieval systems [26] [7]. 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 the difficulty of estimating term-occurrence parameters correctly [31] [22].

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, query refinement heuristics, and natural language processing capabilities in document retrieval systems design [10]. Despite their usefulness, systems of this type are considered *performance systems* [32] - they only *perform* what they were programmed to do (i.e., they are without *learning* ability).

Another major drawback of conventional IR query languages is that these languages require searchers to state precisely what they want. Searchers need to be able to express their needs in terms of precise queries (either in Boolean form or natural languages). As stated earlier, searchers' lack of knowledge in the search domain (anomalous state of knowledge) often precludes their using precise query syntax. To remedy this, many IR systems provide facilities for *relevance feedback*, with which searchers can identify documents of interest to them. IR systems can then use the keywords assigned to these desired documents to find other potentially relevant documents - a simple keyword matching process [31]. The IR systems made no attempt to distinguish among the attributes of the desired documents for their relative importance to the searchers' needs. We believe that the ability for a system to ``learn'' from its searchers' relevance feedback is the key to designing truly adaptive and

intelligent information retrieval systems.

In database management systems, user-friendly query languages were only available after the introduction of relational databases in the 1970s [16]. Currently, user-driven DBMS query languages can be classified as: (1) a command-driven language, (2) a menu (or form) based interface, and (3) a natural language interface.

Declarative command languages such as SQL and QUEL are similar to the Boolean queries in conventional IR in that precise syntax and well-articulated needs are required of the users. However, the syntax of such relational query languages are much like English and generally easy to learn. With some minimal training, most searchers can use these languages easily (if they know exactly what they want from the databases).

Many relational DBMSs provide menu or form based interfaces, which do not require users to issue explicit commands such as SELECT..FROM...WHERE..., The interface is operated by choosing items from a menu or filling in items on a form, e.g., ``query-by-form'' (QBF). By calling up a form on the screen and filling in the desired attribute values, a user can use that form to formulate simple queries concerning a particular table. Such menu-driven interface requires even less training of the users, but they still need to have a clear idea of what they want. Similar to the NLP effort in IR [18], researchers in DBMS have explored various designs for natural language queries in DBMS. For example, Codd's Rendezvous System [15] used a lexicon to convert natural language queries into relational calculus with a predefined set of phrase transformation rules. Harris's INTELLECT system [23], now available commercially, searches a database to discover the user's intended meaning by using grammar rules of English syntax. Requirements for developing NLP query front-end were proposed in [34] [24]. Natural language processing systems typically suffer from heavy processing overhead and require a database-specific lexicon and grammatical rules. As a result, users may not be aware of how the system actually interprets a query and the system has no way of predicting exactly what the user may want [20].

Despite the popularity of the inverted index based information retrieval systems and the relational DBMSs, these systems only perform a keyword-matching function and require users to articulate their needs clearly and

precisely. We believe a more ``intelligent'' and proactive database system should be able to ``learn'' from a user's query session, and especially to perform some inductive reasoning based the desired and undesired records (and the associated attributes of those records) identified by the users. Such a novel query paradigm, which we refer to as the Inductive Query by Examples (IQBE) approach, requires equipping the database systems with an inductive learning component. We present an overview of inductive learning techniques below.

Inductive Learning for IR/DBMS

The symbolic machine learning techniques, the resurgent neural networks approach, and evolution-based genetic algorithms provide drastically different methods for inductive learning. These techniques, which are diverse in their origins and behaviors, have shown unique capabilities for analyzing large amounts of data and identifying patterns and behaviors. In symbolic machine learning, knowledge is presented in the form of symbolic descriptions of the learned concepts. One of the most promising symbolic machine algorithms is Quinlan's ID3, which is a decision-tree building algorithm based on the entropy concept [27] [28]. ID5R, a variant of ID3, is an efficient algorithm for incremental learning [33]. In connectionist learning, knowledge is learned and remembered by a network of interconnected neurons, weighted synapses, and threshold logic units [25] [26]. One of the most popular algorithms in this class is the Backpropagation network developed by Rumelhart and McClelland [30]. 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. The best known algorithm in this class is probably the classifier system proposed by Holland [6], which represents knowledge in terms of parallel, interconnected production rules.

These very different techniques have been found to achieve similar performances for various engineering, business, and biomedical applications. However, ID3 and ID5R have been recognized to be simple, easy to understand, and fast. Neural networks and genetic algorithms, on the other hand, require significant amounts of computation but can be implemented elegantly in parallel machines. Neural networks also exhibit excellent noise-resistant and fault-tolerant capabilities [35] [21].

The learning algorithms cited above have drawn attention from researchers in information science, computer science, and MIS. Neural network computing, in particular, seems to fit well with conventional retrieval models such as the vector space model and the probabilistic models [26]. In [19], Doszkocs et al. provided an excellent overview of the use of connectionist models in information retrieval. 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 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.

The work of Belew is probably the earliest connectionist model adopted in IR. In AIR [1], 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. Based on the network representation, spreading activation methods such as constrained spreading activation adopted in GRANT [17] and the branch-and-bound algorithm adopted in METACAT [10] can be considered as variants of connectionist activation.

Chen et al. [12] [13] [] 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 [13]. In [] 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. Despite

the popularity of using neural networks for information retrieval, we see only limited use of other inductive learning techniques.

In [22], Gordon presented a genetic algorithm based approach for document indexing. Computing document descriptions (keywords) are associated with a document and altered over time by using genetic mutation and crossover operators. In [11], we developed a GA-NN hybrid system, called GANNET, for IR. The system performed *concept optimization* for user-selected documents using the 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.

In [5], the researchers used discriminant analysis and a simple symbolic learning technique for document classification. Their symbolic learning process simply represented the numeric classification results in terms of IF-THEN rules. In [14], we adopted an ID3 decision-tree building algorithm for information retrieval. The ID3 algorithm was able to help construct conceptual queries by analyzing sample documents identified by the searchers. In this project we adapted ID5R for inductive query by examples in IR/DBMS. ID5R was chosen because of its natural and comprehensible representation (i.e., decision trees or production rules), its efficient (fast) real-time performance, and its incremental learning capability. More details will be discussed in the next section.

ID5R for Inductive Query by Examples

In this section, we describe in detail the ID3 and ID5R algorithms and our modifications to them.

ID3 is a decision-tree building algorithm developed by Quinlan [27] [28]. 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 the 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 only contains 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 first. 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. Interested readers are referred to [29] for an up-to-date description of the basic ID3 algorithm and its variants.

Considered as the incremental version of the ID3 algorithm, ID5R, developed by Utgoff [33], is guaranteed to build the same decision tree as ID3 for a given set of training instances [29]. In ID5R, a non-leaf node contains an attribute test (same as in ID3) and a set of other non-test attributes, each with the 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. The algorithmic detail for the original ID5R can be found in [33].

Both ID3 and ID5R assume a universe of objects that need to be classified into a set of disjoint classes. Both also assume that the objects are described by a set of attributes, each of which may have a small range of values (discrete or continuous) that uniquely classify each object of the universe as belonging to one of the disjoint classes. Based on these assumptions, we adapted the algorithms for the unique characteristics of the IR/DBMS environment.

- **Positive and negative classes:** In IR/DBMS, 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 [14], we adopted an ID3 algorithm for adaptive IR. In the 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.

- **Relevance feedback:** In order to create a robust and real-time inductive learning system, a *relevance feedback* scheme was introduced into our system. Since, 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 the users after examining a small portion of the database. When a predetermined number of desired documents had been found (say 3, in our current 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 users decided to stop or the complete database had been traversed.

  During the relevance feedback process, the newly confirmed documents,

either desired or undesired, can be used by ID5R to update the decision tree it previously constructed. 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.

We developed the two algorithms (ID3 and ID5R) and a simple interface in C language. Our prototype system is called the IQBE for Interactive Query by Examples. The system can be run on 386/486 personal computers or UNIX work stations such as DECstation 5000/120.

Two Examples

We present two examples to show how the system works. The first is a sample session for literature search (with document id and keywords). The second example is a DBMS application (i.e., tabular format, multiple fields).

An Information Retrieval Example

We developed a small text literature 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 IQBE system and see whether the system would be able to identify this target of documents 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. 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 (2-5) keywords 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 is 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:

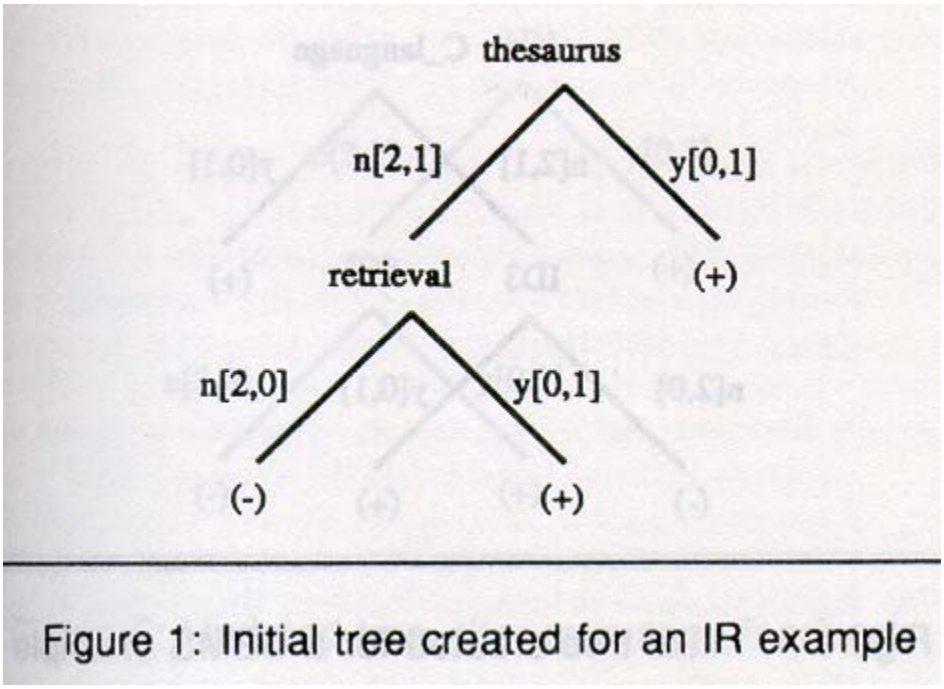
. *Initial Training Instances*

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 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 looks different from the original one and can be summarized as 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

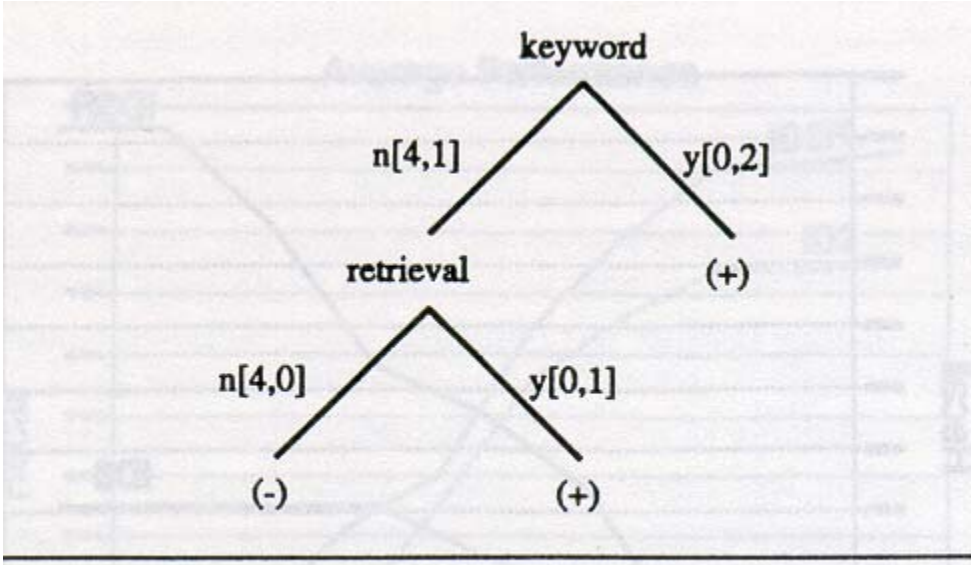


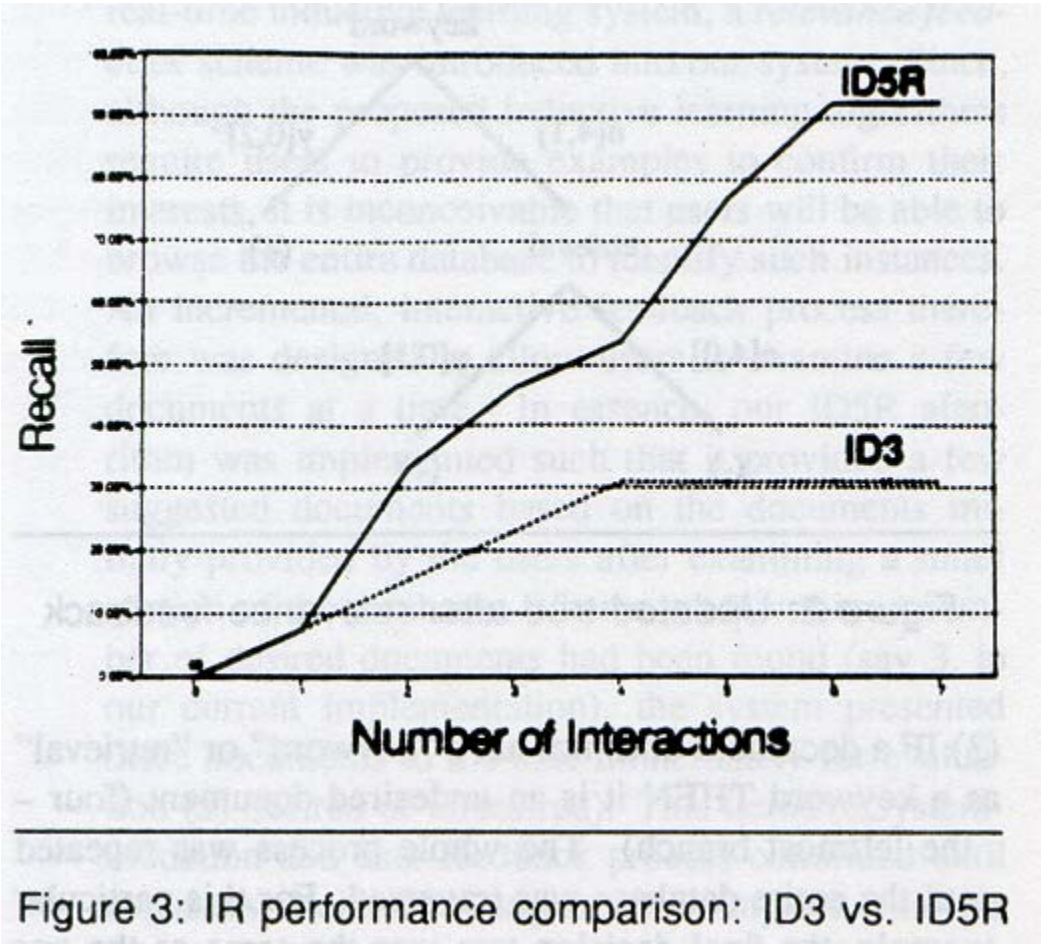
Figure 2: Updated tree after relevance feedback

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 only used the initial document feedback from the users to construct a decision tree and used the tree to search the entire database. ID5R, on the other hand, collected new evidence during each iteration and updated its trees accordingly. The *recall* measure is defined as:

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

As shown in Figure 3, ID5R took advantage of the new training instances and significantly improved it recall ratio to about 92% after 7 iterations. ID3, on the other, was not able to improve its recall performance even after the algorithm searched the entire database. Its eventual recall level was at 31%. (At each interaction point, ID5R searched only a portion of the entire database. ID3 did not have any interaction with its users and its performance at each interaction point was computed based on the same documents visited by ID5R.)

Figure 3: IR performance comparison: ID3 vs. ID5R



A Database Management System Example

We show a relational DBMS application as a second example. We constructed a sample expertise database containing information about 30 employees. Each record was stored as a tuple and consisted of a unique primary key (employee name), one department field, and three expertise fields. Part of the this employee expertise table is shown below.

Aaron	MIS	NN	C_language	ID3
Abbot	MIS	semantic_net	C_language	NN
Alvin	ADM	english	typing	chinese

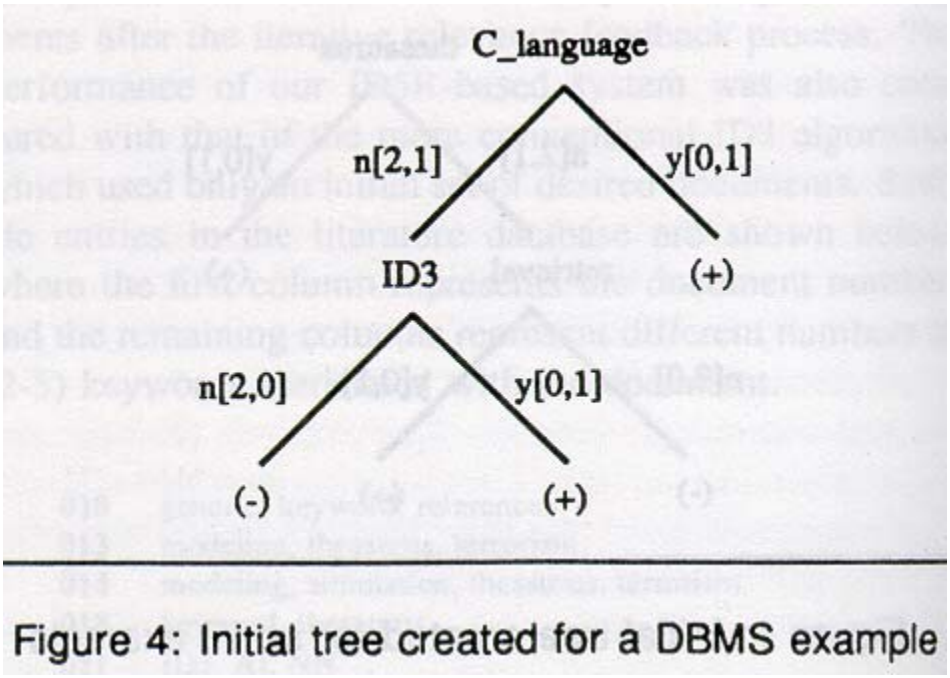
Amanda	MIS	Pascal	C_language	cobol
Andrew	FIN	management	investment	real_estate
Betty	ACCT	taxation	management	accounting
Barbara	ACCT	accounting	finance	chinese
Bind	MIS	NN	X_window	C_language
Ben	MKT	simulation	statistic	C_language
Bob	MIS	NN	Pascal	Unix
Carol	MIS	NN	scheduling	Unix
...				
Hamam	OM	simulation	modeling	scheduing
Hoover	ENGR	software_engineering	OO_programming	database
Harrison	MIS	online_processing	CICS	IR
Lucas	OM	operation_research	NN	data_structure

A possible scenario for this case might arise when a manager wishes to recruit employees for a project team and is looking for employees with expertise similar to that of a few employees whose skills he already knew. The manager could initiate our IQBE system by providing a list of these ``example employees'' and request the system to find similar employees. In this case, he decided that Linda and Lyn were desired employees (+), but not Bill nor Gary (-). When no expertise field was given it was indicated as nil.

Linda	MIS	C_language	NN	GA	(+)
Lyn	ENGR	X_window	ID3	nil	(+)
Bill	MIS	X_window	Cobol	nil	(-)
Gary	ENGR	simulation	Cobol	Unix	(-)

The test attributes for these instances included: MIS, C_language, NN, GA, ENGR, X_window, and ID3. The system constructed the initial decision tree shown in Figure 4. This decision tree can be represented as production rules: (1) IF an employee has ``C_language'' as an expertise filed THEN he/she is desired (one +, the rightmost branch); (2) IF an employee does not have ``C_language'' as an expertise, but has ``ID3'' THEN he/she is also a desired employee (one +, the middle branch); (3) IF an employee does not have ``C_language'' or ``ID3'' as expertise THEN he is an undesired employee (two -, the leftmost branch). Using this result, the system generated three desired employees: Aaron, Abbot, and Amanda. Their employee records were then presented to the manager, who confirmed that Aaron and Abbot were indeed desired but not Amanda (shown below).

Figure 4: Initial tree created for a DBMS example



Aaron	MIS	NN	C_language	ID3	(+)
Abbot	MIS	semantic_network	C_language	NN	(+)

Amanda	MIS	Pascal	C_language	Cobol	(-)

The system then updated the tree to the new tree shown in Figure 5. The system-induction and user-feedback process continued until the whole database was traversed. ID5R's recall performance at each interaction point and the performance of ID3 are shown in Figure 6. ID5R was able to reach 100% recall level, but ID3 was able to reach only 50% level.

Figure 5: Updated tree after relevance feedback

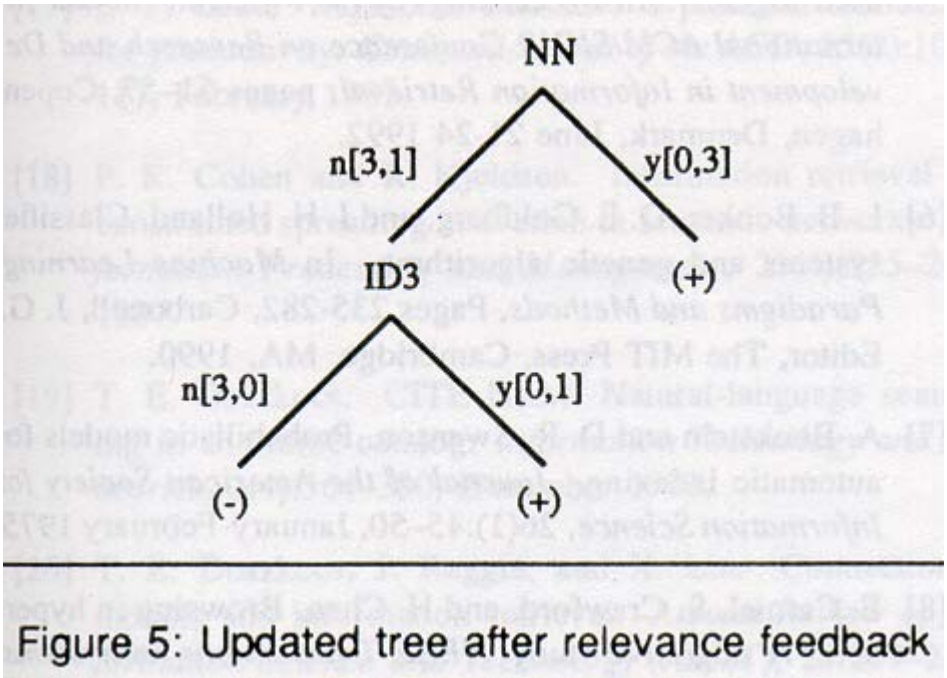


Figure 6: DBMS performance comparison: ID3 vs. ID5R

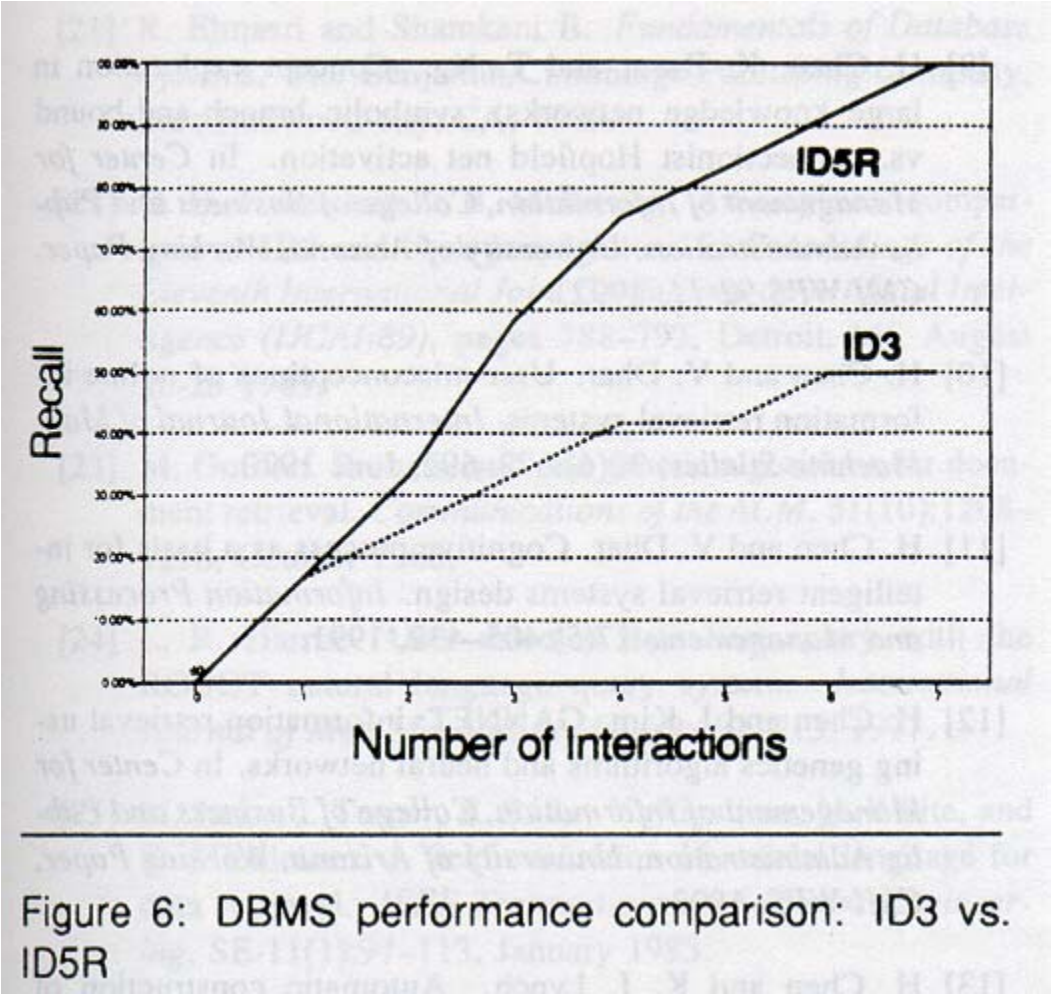


Figure 6: DBMS performance comparison: ID3 vs. ID5R

System Testing

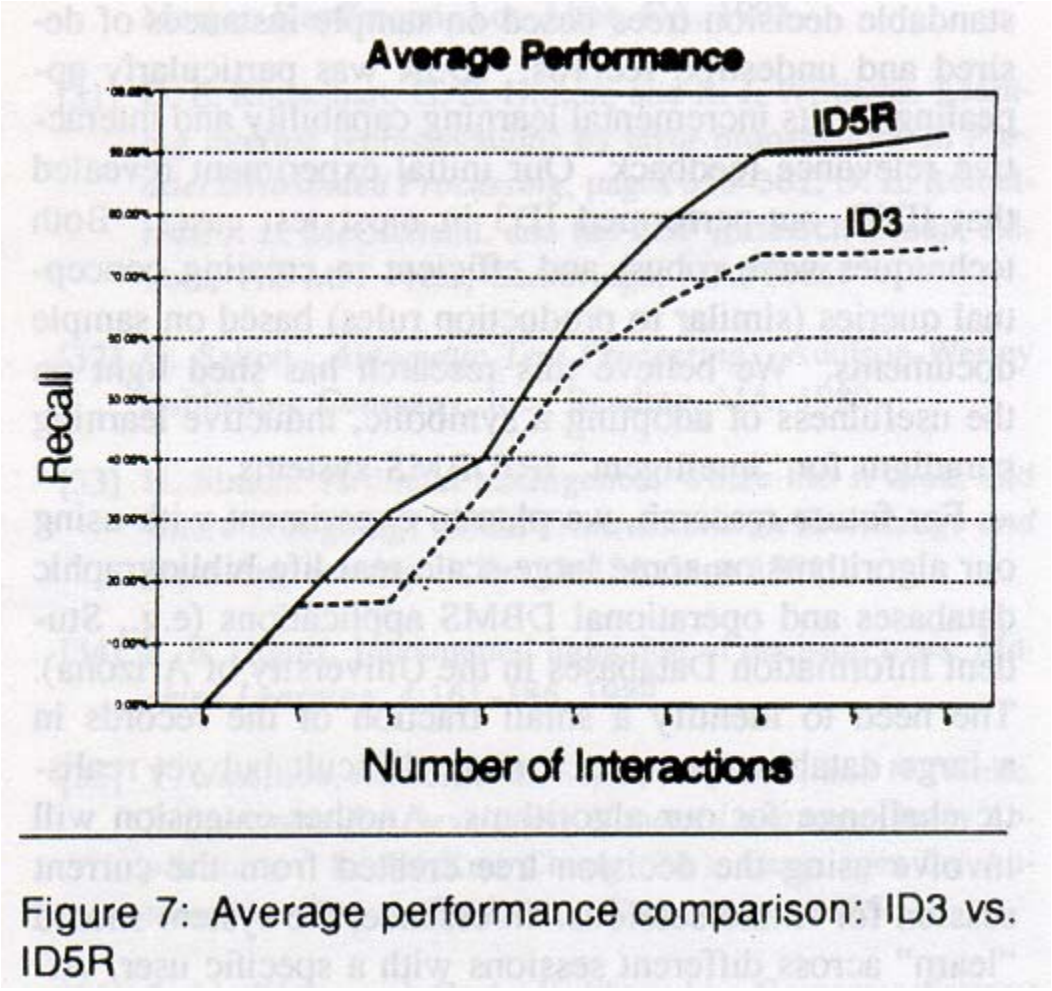
We developed a test database of about 1000 documents from the 1992 COMPEN CD-ROM collection of computing literature. We then identified 10 research topics, each of which had between five and 20 relevant documents in the database (manually identified). The testing was conducted by comparing the recall of the ID3 algorithm and that of the ID5R incremental approach using the 10 research topics. The experimental process was similar to that in the two ID/DBMS examples described earlier. However, due to the size of the test collection, identifying a small set of desired records from a large database posed a more difficult challenge to the IQBE system.

ID5R and ID3 achieved the same levels of performance for five of the ten topics. This was because the initial documents presented had very precise keywords assigned to them. New instances provided during relevance

feedback were consistent with the initial documents. For the other five topics, ID5R's performance increased gradually until it reached about 93%. ID3, on the other hand, was able to reach 74%. These research topics tended to have more diverse keywords in the initial documents provided. The average performance for the ten test cases are shown in Figure 7.

In conclusion, both inductive learning techniques worked surprisingly well even for this large collection of records. However, ID5R's incremental learning and relevance feedback characteristics made it even more robust and appealing for large-scale, real-time IR/DBMS applications.

Figure 7: Average performance comparison: ID3 vs. ID5R



Conclusions and Discussion

Information retrieval and database management systems research has been advancing very quickly over the past few decades. Researchers have experimented with techniques ranging from probabilistic models (in IR), relational databases and object-oriented databases (in DBMS) to the knowledge-based approach and recent machine learning techniques. At each stage, significant insights regarding how to design more useful and ``intelligent'' information systems have been gained.

In this research we first conducted a brief review of the conventional and the query-by-examples techniques used in IR and DBMS. The problem with the conventional techniques has been that users of such systems need to be able to articulate their needs clearly and be able to represent such needs in precise query syntax. The systems play no role in helping users refine their queries or ``learn'' from the users' relevance feedback.

Quinlan's ID3 and Utgoff's ID5R were selected from among the various learning paradigms to help perform ``Inductive Query by Examples'' (IQBE) for IR and DBMS applications. Both techniques induced simple, understandable decision trees based on sample instances of desired and undesired records. ID5R was particularly appealing for its incremental learning capability and interactive relevance feedback. Our initial experiment revealed that ID5R out-performed ID3 in most test cases. Both techniques were robust and efficient in creating conceptual queries (similar to production rules) based on sample documents. We believe this research has shed light on the usefulness of adopting a symbolic, inductive learning paradigm for ``intelligent'' IR/DBMS systems.

For future research, we plan to experiment with using our algorithms on some large-scale real-life bibliographic databases and operational DBMS applications (e.g., Student Information Databases in the University of Arizona). The need to identify a small fraction of the records in a large database presents a more difficult but yet realistic challenge for our algorithms. Another extension will involve using the decision tree created from the current session for future sessions. In essence, the system should ``learn'' across different sessions with a specific user. After repeated uses, an ``intelligent'' information system will be able to customize its interaction and suggestions for each user. However, extensive design effort is still required to achieve this long-term goal of developing truly ``adaptive'' information system.

Acknowledgment

This project was supported in part by an NSF grant, IRI-9211418, 1992-1994.

Bibliography

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

2 N. J. Belkin, R. N. Oddy, and H. M. Brooks.
Ask for information retrieval: Part I. background and theory.
Journal of Documentation, 38(2):61-71, June 1982.

3 N. J. Belkin, R. N. Oddy, and H. M. Brooks.
Ask for information retrieval: Part II. results of a design study.
Journal of Documentation, 38(3):145-164, September 1982.

4 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.

5 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.

6 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.

7 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.

8 E. Carmel, S. Crawford, and H. Chen.
Browsing in hypertext: A cognitive study.
IEEE Transactions on Systems, Man and Cybernetics, 22(5):865-884, September/October 1992.

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

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

11 H. Chen and J. Kim.
GANNET: a machine learning approach to document retrieval.
Journal of Management Information Systems, 11(3):7-41, Winter 1994-95.

12 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.

13 H. Chen, K. J. Lynch, K. Basu, and D. 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.

14 H. Chen and G. Mahboob.
Example-based document retrieval: an inductive machine learning approach.
In *Center for Management of Information, College of Business and*

Public Administration, University of Arizona, Working Paper, CMI-WPS, 1992.

15 E. Codd.
How about recently?
In *English Dialog with Relational Data Base Using Rendezvous Version 1, edited by B. Shneiderman*, 1978.

16 E. F. Codd.
Relational database: a practical foundation for productivity.
Communications of the ACM, 25(2):109-117, February 1981.

17 P. R. Cohen and R. Kjeldsen.
Information retrieval by constrained spreading activation in semantic networks.
Information Processing and Management, 23(4):255-268, 1987.

18 T. E. Doszkocs.
CITE NLM: Natural-language searching in an online catalog.
Information Technology and Libraries, 2(4):364-380, December 1983.

19 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.

20 R. Elmasri and Shamkant B.
Fundamentals of Database Systems.
The Benjamin/Cummings Publishing Company, Inc., Redwood city, CA, 1989.

21 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.

22 M. Gordon.
Probabilistic and genetic algorithms for document retrieval.
Communications of the ACM, 31(10):1208-1218, October 1988.

23 L. R. Harris.
User oriented data base query with the ROBOT natural language query system.
International Journal of Man Machine Studies, 9:697-713, 1977.

24 M. Jarke, J. Turner, E. Stohr, Y. Vassiliou, N. White, and K. Michielsen.
A field evaluation of natural language for data retrieval.
IEEE Transactions on Software Engineering, SE-11(1):97-113, January 1985.

25 R. P. Lippmann.
An introduction to computing with neural networks.
IEEE Acoustics Speech and Signal Processing Magazine, 4(2):4-22, April 1987.

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

27 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.

28 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.

29 J. R. Quinlan.
C4.5: Programs for Machine Learning.
Morgan Kauffmann, Los Altos, CA, 1993.

30 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.

31 G. Salton.
Automatic Text Processing.
Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.

32 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.

33 P. E. Utgoff.
Incremental induction of decision trees.
Machine Learning, 4:161-186, 1989.

34 Y. Vassiliou, M. Jerke, E. Stohr, J. Turner, and N. White.
Requirement for Developing Natural Language Query Applications.
In Shi-Kuo Chang (Ed.), Languages for Automation, Plenum Publishing Corporation, 1985.

35 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.

---

hchen@bpa.arizona.edu