A Comparison of Text Retrieval Models

HOWARD R. TURTLE[1] AND W. BRUCE CROFT[2]

[1] *West Publishing Company, St Paul, MN 55164, USA*

[2] *Computer Science Department, University of Massachusetts, Amherst, MA 01002, USA*

Many retrieval models have been proposed as the basis of text retrieval systems. The three main classes that have been investigated are the exact-match, vector space and probabilistic models. The retrieval effectiveness of strategies based on these models has been evaluated experimentally, but there has been little in the way of comparison in terms of their formal properties. In this paper we introduce a recent form of the probabilistic model based on inference networks, and show how the vector space and exact-match models can be described in this framework. Differences between these models can be explained as differences in the estimation of probabilities, both in the initial search and during relevance feedback.

Received December 1991

1. INTRODUCTION

Information Retrieval (IR) is concerned with identifying documents in a collection that best match a description of a user's information need. Research in this field spans many subdisciplines of computer and information science, but central to any effective retrieval system is the identification and representation of document content, the acquisition and representation of the information need, and the specification of a matching function that selects relevant documents based on these representations. A retrieval model specifies the details of the document representation, the query representation, and the matching function.

A number of retrieval models have been proposed since the mid-1960s. They have evolved from *ad hoc* models intended for use with small, highly structured document surrogates (e.g. bibliographic records containing title, author and subject codes) to current generation models that have strong theoretical bases and which are intended to accommodate a variety of full text document types. Current models handle documents with complex internal structure and most incorporate a learning or 'relevance feedback' component that can improve performance when presented with sample relevant documents.

Three main classes of retrieval models are in current use: exact match models which form the basis of most commercial retrieval systems, vector space models which view documents and queries as vectors in a high-dimension vector space and use distance as a measure of similarity, and probabilistic models which view retrieval as a problem of estimating the probability that a document representation matches or satisfies a query. The vector space and probabilistic models have been shown experimentally to offer significant improvements in retrieval performance over exact-match models,* but have only recently been used in commercial products.

The underlying theory is different for each of these model types and each has different performance characteristics, both in terms of retrieval effectiveness and computational requirements. At the same time, all of these models have striking similarities. In this paper we show how the differences between these models can be viewed as differences in the way probabilities are estimated and combined in a probabilistic model. In doing so we show that the estimation problems for the probabilistic and vector space models are essentially equivalent, and that exact-match models simply restrict the range of probability values to be considered. The important question is not whether we view these estimates as weights or probabilities, but what sources of evidence about document and query content can be used to improve the estimates and retrieval performance.

In what follows we will describe retrieval models and introduce a probabilistic model based on inference networks. We then show how exact match and vector space retrieval can be represented within the inference network model, and conclude with a comparison of relevance feedback techniques.

* The most common measures of retrieval performance are precision and recall. Precision is the proportion of a retrieved set of documents that is relevant to the query. Recall is the proportion of all documents in the collection that are relevant to a query that is actually retrieved.

2. RETRIEVAL MODELS

Every information system has, either explicitly or implicitly, an associated theory of information access and a set of assumptions that underlie that theory. We use the term theory here in the mathematical or logical sense in which a theory refers to a set of axioms and inference rules that allow derivation of new theorems. A *model* is an embodiment of the theory in which we define the set of objects about which assertions can be made and restrict the ways in which classes of objects can interact. Models allow us to compare different approaches to information access and make predictions about system performance that can be evaluated.

In the case of IR, a *retrieval model* specifies the representations used for documents and information needs, and how they are compared. For example, many retrieval models have assumed representations based on binary or weighted index terms, and comparison based on a similarity measure. There has been, in the past, less emphasis on models that specify how representations should be extracted from document and query texts, and which representations produce the best performance. Research on retrieval models incorporating probabilistic indexing[9,32] and machine learning[13] explicitly address this issue.

The vector space model and the probabilistic model

have been studied and used for some time by IR researchers and are quite well understood. Even so, only certain aspects of the underlying theories are clearly defined, whereas others, especially those dealing with the user, remain somewhat vague. For example, papers describing various forms of these models usually state what independence assumptions are being made (for example, Ref. 35), but are less clear when it comes to defining 'relevance'. The vector space model[29] does not attempt to define anything to do with the external realities of users and information needs, and instead could be regarded as a mathematical description of one component of an IR system, rather than the IR process as a whole.[1] The models of Robertson *et al.*,[20] Fuhr[9] and Turtle and Croft[32] are somewhat more comprehensive but still leave a number of questions unanswered, such as how different types of information needs and goals are represented in a retrieval theory.

Some common assumptions made in retrieval models are:

- the objects being retrieved are primarily textual;
- the retrieval of an object is independent of whether other objects are retrieved (although this is not true for cluster-based retrieval);
- retrieval is based on representations of textual content and information needs;
- both text and information need representations are uncertain.

These assumptions have led to retrieval techniques that emphasise ranked retrieval, iterative query formulation, relevance feedback, and analysis of simple natural-language queries.

Other types of information systems have different theories and assumptions associated with them. The area of database systems is particularly important and includes theories associated with relational database systems, deductive database systems and object-oriented systems.[38] Typical database systems represent more types of object than typical IR systems, but the objects usually have very well-defined content. Issues involving uncertainty in query specification or object representation have largely been ignored in the database field (with a few exceptions).[11,15] Making these more restrictive assumptions has allowed significant progress in areas such as query optimisation. The same justification cannot be made for many commercial IR systems that also ignore retrieval models based on uncertainty. In this case, the unrealistic assumptions underlying these systems result in poor retrieval performance.

In this paper, we will concentrate on using a retrieval model based on inference nets to compare different approaches to building IR systems. We do this by showing how the vector space model and the Boolean model, which are system-oriented models, can be described using the inference net framework. This has the benefit of showing how weighting techniques and Boolean query formulations can be explained in terms of a retrieval model that describes the processes of representing documents and information needs. It also suggests that, rather than being one of a set of alternative retrieval models, the probabilistic approach is the current best theory for information retrieval.

3. INFERENCE NETWORK MODEL

The inference network retrieval model description we give here focuses on the details necessary to show how exact-match and vector space operations can be accommodated. The model is described more fully in Refs 31 and 32. In this model, retrieval is viewed as an evidential reasoning process in which multiple sources of evidence about document and query content are combined to estimate the probability that a given document matches a query.

An inference network or causal network is a directed, acyclic dependency graph in which nodes represent propositional variables or constants, and edges represent dependence relations between propositions.[16,17] If a proposition represented by a node p 'causes' or implies the proposition represented by node q, we draw a directed edge from p to q. The node q contains a link matrix that specifies $P(q|p)$ for all possible values of the two variables. When a node has multiple parents, the matrix specifies the dependence of that node on the set of parents (π_q) and characterises the dependence relationship between that node and all nodes representing its potential causes. Given a set of prior probabilities for the roots of the DAG, these networks can be used to compute the probability or degree of belief associated with all remaining nodes.

The inference network shown in Fig. 1 consists of two component networks: a document network and a query network. The document network represents the document collection and may incorporate numerous document representation schemes. The document network is built once for a given collection and its structure does not change during query processing. The query network consists of a single node which represents the user's information need, and one or more query representations which express that information need. A query network is built for each information need and is modified during query processing as existing queries are refined or new queries are added in an attempt to better characterise the information need. The document and query networks are joined by links between representation concepts and query concepts. All nodes in the inference network are binary-valued.

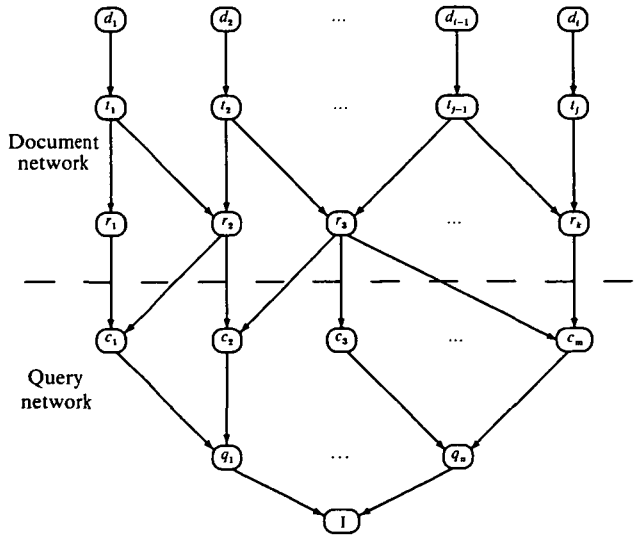

Figure 1. Basic document inference network.

3.1 Document network

The document network consists of document nodes (d_is), text representation nodes (t_js), and concept representation nodes (r_ks). Each document node represents an actual document in the collection. A document node corresponds to the fact that a specific document has been observed. The form of the document represented depends on the collection and its intended use, but we shall assume that a document is a well-defined object and will focus on traditional document types (e.g. monographs, journal articles, office documents).

Document nodes correspond to abstract documents rather than their physical representations. A text representation node or text node corresponds to a specific text representation of a document. We shall focus here on traditional document texts, but one can easily imagine other content types for documents (e.g. figures), and multi-media documents might have several content representations (e.g. audio or video). In these cases a single document might have many physical representations. Similarly, a single text content might be shared by more than one document. While this sharing is rare (an example would be a journal article that appears in both a serial issue and a reprint collection) and is not generally represented in current retrieval models, it is common in hypertext systems. For clarity, we will only consider text representations and will assume a one-to-one correspondence between documents and texts. The dependence of a text upon the document is represented in the network by an arc from the document node to the text node.

The content representation nodes or representation nodes can be divided into several subsets, each corresponding to a single representation technique that has been applied to the document texts. For example, if a collection has been indexed using automatic phrase extraction and manually assigned index terms, the set of representation nodes will consist of two distinct subsets or content representation types with disjoint domains. Thus, if the phrase 'information retrieval' has been extracted and 'information retrieval' has been manually assigned as an index term, two representation nodes with distinct meanings will be created. One corresponds to the event that 'information retrieval' has been automatically assigned to a subset of the collection, the second corresponds to the event that 'information retrieval' has been manually assigned to a (presumably distinct) subset of the collection. We represent the assignment of a specific representation concept to a document by a directed arc to the representation node from each text node corresponding to a document to which the concept has been assigned.

The basic document network shown in Fig. 1 is a simple three-level directed acyclic graph (DAG) in which document nodes are roots, text nodes are interior nodes, and representation nodes are leaves. Document nodes have exactly one text node as a child, and each text node has one or more representation nodes as children.

Each document node has a prior probability associated with it that describes the probability of observing that document; this prior probability will generally be set to 1/(collection size) and will be small for reasonable collection sizes. Each text node contains a specification of its dependence upon its parent; by assumption, this dependence is complete, a text node is observed (t_i = *true*) exactly when its parent document is observed (d_i = *true*).

Each representation node contains a specification of the conditional probability associated with the node, given its set of parent text nodes. While, in principle, this would require $O(2^n)$ space for a node with n parents, in practice we shall generally use canonical representations that will allow us to compute the required conditional probabilities when needed. These canonical schemes are described in Section 3.4 and require $O(n)$ space if we need to weight the contribution of each parent, or $O(1)$ space if parents are to be treated uniformly.

3.2 Query network

The query network is an 'inverted' DAG with a single leaf that corresponds to the event that an information need is met, and multiple roots that correspond to the concepts that express the information need. As shown in Fig. 1, a set of intermediate query nodes may also be used in cases where multiple query representations are used to express the information need. These nodes are a representation convenience; it is always possible to eliminate them by increasing the complexity of the distribution specified at the node representing the information need.

In general, the user's information need is internal to the user and is not precisely understood. We attempt to make the meaning of an information need explicit by expressing it in the form of one or more queries that have a formal interpretation. It is unlikely that any of these queries will correspond precisely to the information need, but some will better characterise the information need than others, and several query representations taken together may be a better representation of the information need than any of the individual queries.

The roots of the query network are query concepts, the primitive concepts used to express the information need. A single query concept node may have several representation concept nodes as parents. A query concept node contains a specification of the probabilistic dependence of the query concept on its set of parent representation concepts. The query concept nodes define the mapping between the concepts used to represent the document collection and the concepts that make up the queries. In the simplest case, the query concepts are constrained to be the same as the representation concepts, and each query concept has exactly one parent representation node. In a slightly more complex example, the query concept 'information retrieval' may have as parents both the node corresponding to 'information retrieval' as a phrase and the node corresponding to 'information retrieval' as a manually assigned term.

The attachment of the query concept nodes to the document network has no effect on the basic structure of the document network. None of the existing links needs change and none of the conditional probability specifications stored in the nodes is modified.

A query node represents a distinct query representation and contains a specification of the dependence of the query on the query concepts it contains. Multiple query representations can be obtained from many sources. It is possible that the user might provide more than one form (e.g. a natural language description and a sample

document), but it is more likely that additional forms will be generated automatically based on the original natural-language query or using information obtained by an intelligent interface. In cases where a search intermediary is used, we may have multiple human-generated query representations.

The single leaf representing the information need corresponds to the event that an information need is met. In general, we cannot predict with certainty whether a user's information need will be met by an arbitrary document collection. The query network is intended to capture the way in which meeting the user's information need depends on documents and their representations. Moreover, the query network is intended to allow us to combine information from multiple document representations and to combine queries of different types to form a single, formally justified estimate of the probability that the user's information need is met.

We will often use a simplified form of the basic model of Fig. 1 in which we assume a one-to-one correspondence between document nodes and text nodes and between representation concept and query concept nodes. Under these assumptions, the network of Fig. 1 can be reduced to the network shown in Fig. 2.

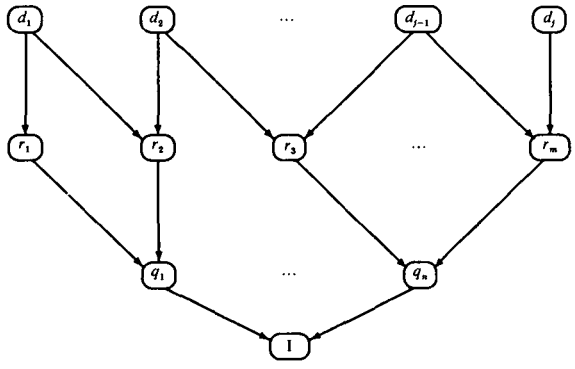

Figure 2. Simplified inference network.

3.3 Use of the inference network

The inference network we have described is intended to capture all of the significant probabilistic dependencies among the variables represented by nodes in the document and query networks. The network, taken as a whole, represents the dependence of a user's information need on the documents in a collection where the dependence is mediated by document and query representations. When the query network is first built and attached to the document network we compute the belief associated with each node in the query network. The initial value at the node representing the information need is the probability that the information need is met, given that no specific document in the collection has been observed and all documents are equally likely (or unlikely). If we now observe a single document d_i and attach evidence to the network asserting $d_i = true$ with all remaining document nodes set to *false* (referred to as instantiating d_i), we can compute a new belief for every node in the network given $d_i = true$. In particular, we can compute the probability that the information need is met given that d_i has been observed in the collection. We can now remove this evidence and instead assert that some d_j, $i \neq j$ has been observed. By repeating this process we can compute the probability that the information need is met, given each document in the collection, and rank the documents accordingly.

In principle, we need not consider each document in isolation, but could look for the subset of documents which produce the highest probability that the information need is met. While a general solution to this best-subset problem is intractable, in some cases good heuristic approximations are possible. Best-subset rankings have been considered in IR[3,30] and similar problems arise in pattern recognition, medical diagnosis and truth-maintenance systems. See Ref. 17 for a discussion of the best-subset or belief revision problem in inference networks.

3.4 Link matrix forms

For all non-root nodes in the inference network we must estimate the probability that a node takes on a value, given any set of values for its parent nodes. If a node a has a set of parents $\pi_a = \{p_1, \ldots, p_n\}$, we must estimate $P(a|p_1, \ldots, p_n)$. Since we are dealing with binary-valued propositions, this estimate can be represented by a matrix of size $O(2^n)$ for a node with n parents and specifies the probability that a takes the value *true* or *false* for all combinations of parent values. The update procedures for inference networks then use the probabilities provided by the set of parents to condition over the link matrix values to compute our belief in a (bel(a) or $P(a = true)$). Encoding our estimates in link matrix form is practical only for nodes with a small set of parents, so our estimation task has two parts: how do we estimate the dependence of a node on its set of parents, and how do we encode these estimates in a compact form?

In this section we shall define canonical link matrix forms that are useful for retrieval networks. By a cononical form we mean that, given an ordering on a set of n parents, we can compute the link matrix value $L[i, j]$, $i \in \{0, 1\}$, $0 \leqslant j < 2^n$, which corresponds to an assignment of truth values to parent variables. When writing a link matrix we use the row number to index values assumed by the child node, and use a binary representation of the column number to index the values of the parents. We use the high-order bit of the column number to index the first parent's values, the second most high order for the second parent, and so on. The three-parent example below illustrates this notation.

We shall describe five canonical link matrix forms. Three of these forms implement the Boolean operators *and*, *or* and *not*; the remaining two forms implement weighted sums that are used for probabilistic retrieval. A number of other forms are possible.

For illustration, we shall assume that a node Q has three parents A, B and C and that

$$P(A = true) = a$$
$$P(B = true) = b$$
$$P(C = true) = c.$$

For *and*-combinations, Q is true only when A, B, and C are all true, and we have a matrix of the form

$$L_{\text{and}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Conditioning over the set of parents we have

$$P(Q = true) = abc, \tag{1}$$

which is the familiar rule for conjunctive combinations of events.

For *or*-combinations, Q will be true when any of A, B or C is true, and false only when A, B and C are all false. This gives a link matrix of the form

$$L_{or} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Again conditioning over the set of parents, we have

$$\begin{aligned} P(Q = true) &= (1-a)(1-b)c + (1-a)b(1-c) + (1-a)bc \\ &\quad + a(1-b)(1-c) + a(1-b)c + ab(1-c) + abc \\ &= a+b+c-(ab+bc+ac)+abc \\ &= 1-(1-a)(1-b)(1-c), \end{aligned} \tag{2, 3}$$

which is the familiar rule for disjunctive combinations of events that are not known to be mutually exclusive.

The *not* operator is defined only for unary propositions or nodes with a single parent. If Q has the single parent A, $Q = true$ exactly when $A = false$, which gives a link matrix of the form

$$L_{not} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

and results in

$$P(Q = true) = 1-a. \tag{4}$$

If we restrict the parent nodes for any of the logic operators to values 0 or 1, Q must also have a value of 0 or 1. If we allow parents to take on probabilities in the range [0, 1] these inference networks provide a natural probabilistic interpretation of the Boolean operators. The use of these canonical forms to simulate Boolean retrieval is discussed in Section 4.1.

A fourth link matrix form arises when our belief in Q depends only on the number of parents that are true. If j corresponds to a link matrix column number for which m parents are true (in which m bits = 1), then

$$L_{sum}[1,j] = \frac{m}{n}$$

$$L_{sum}[0,j] = \frac{n-m}{n}.$$

Thus, for our three-parent example

$$L_{sum} = \begin{pmatrix} 1 & \frac{2}{3} & \frac{2}{3} & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} & 1 \end{pmatrix}.$$

Evaluation of this *sum* link matrix results in

$$\begin{aligned} P(Q = true) &= \tfrac{1}{3}(1-a)(1-b)c + \tfrac{1}{3}(1-a)b(1-c) + \tfrac{2}{3}(1-a)bc \\ &\quad + \tfrac{1}{3}a(1-b)(1-c) + \tfrac{2}{3}a(1-b)c + \tfrac{2}{3}ab(1-c) + abc \\ &= \frac{a+b+c}{3} \end{aligned}$$

In this matrix form all parents are weighted equally; if all parents are observed to be true then $P(Q = true)$ is three times greater than if one parent is observed. A number of other weightings are possible. For example, we can choose weights so that Q is true when any m parents are true to implement an 'm-of-n' operator, or we can choose weights so that the first parent observed has the most influence on our belief in Q and the second and third parents have less influence on our belief (essentially, the *or*-combination is an extreme case in which only the first parent influences our belief). Similarly, we can choose weights so that the first parent observed has little or no influence on our belief in Q and the second and third parents determine our belief.

The final link matrix form we will discuss is a generalisation of the *sum* matrix in which each parent has a weight associated with it, as does the child. In this weighted-sum (*wtd_sum*) matrix, our belief in Q depends on the specific parents that are true – parents with larger weights have more influence on our belief. The weight at Q acts to set the maximum belief that can be achieved at Q. If we let w_a, w_b, and w_c be the parent weights, $0 \leqslant w_q \leqslant 1$ be the weight at Q, and $t = w_a + w_b + w_c$ for our example, then we have a link matrix of the form

$$\begin{pmatrix} 1 & 1-\frac{w_c w_q}{t} & 1-\frac{w_b w_q}{t} & 1-\frac{(w_b+w_c)w_q}{t} \\ 0 & \frac{w_c w_q}{t} & \frac{w_b w_q}{t} & \frac{(w_b+w_c)w_q}{t} \\ 1-\frac{w_a w_q}{t} & 1-\frac{(w_a+w_c)w_q}{t} & 1-\frac{(w_a+w_b)w_q}{t} & 1-w_q \\ \frac{w_a w_q}{t} & \frac{(w_a+w_c)w_q}{t} & \frac{(w_a+w_b)w_q}{t} & w_q \end{pmatrix}.$$

Evaluation of this link matrix form results in

$$\begin{aligned} P(Q = true) &= \frac{w_c w_q}{t}(1-a)(1-b)c + \frac{w_b w_q}{t}(1-a)b(1-c) \\ &\quad + \frac{(w_b+w_c)w_q}{t}(1-a)bc + \frac{w_a w_q}{t}a(1-b)(1-c) \\ &\quad + \frac{(w_a+w_c)w_q}{t}a(1-b)c + \frac{(w_a+w_b)w_q}{t}ab(1-c) + w_q abc \\ &= \frac{(w_a a + w_b b + w_c c)w_q}{t} \end{aligned} \tag{5}$$

The sum matrix is a special case of *wtd_sum* where all weights are 1.

In the network model, the distinction between the Boolean operators and the probabilistic sum operators begins to blur. The operators are, after all, only specialised link matrix forms, and the model allows Boolean and probabilistic operators to be freely mixed in expressions. The ability to mix operator types is required to allow us to combine query forms and is useful in representing phrases and for developing relevance feedback strategies.

While these five canonical forms are sufficient for the retrieval inference networks described here, many others are possible. Further, when n is small (say, less than 5 or 6) we can use the full link matrix if the dependence of a node on its parents is known not to fit a canonical form.

3.5 Estimating the probabilities

In the network of Fig. 2 the values of the root nodes are fixed during instantiation, but the conditional probabilities at the remaining nodes must be estimated. Estimates are required at representation and query concept nodes and at the information need.

The estimates used in the query network are either fixed by the operator type (Boolean or sum) or require estimation of the relative contribution of the parent nodes (weighted sum). When a weighted sum is used at a query node, the contribution of the parent concepts is based on the frequency of the parent concept in the query. A sum estimate is used most often for the information need, although a weighted sum can be used if we have externally supplied information about the importance or quality of individual queries.

The estimates at the representation concept nodes also make use of a weighted sum, but setting the parent weights is somewhat more involved. A number of term-weighting techniques that have been developed in previous IR research can be adapted for use here, but consistently good results are obtained if we assume that our belief in a representation concept depends upon two factors: the frequency with which the concept occurs in an instantiated document and the 'surprise' associated with observing that concept assigned to a randomly selected document. These two factors, when combined, provide a good estimate of the term weight in Equation (5). If the probability that concept t is assigned to a randomly selected document is

$$p_t = \frac{\text{number of documents containing } t}{\text{number of documents in the collection}}$$

the surprise[22] associated with observing t in a randomly selected document is

$$s_t = -\log p_t.$$

This surprise factor is equivalent to the familiar inverse document frequency (*idf*) measure[29] used in the vector space model.

These estimates are subjective in that they are not based solely on the frequency of events in representative samples. The Boolean and sum functions are based on the properties of the underlying operators and make no assumption about the collection distribution. The estimates used with the weighted-sum function are based on an expected distribution of features in a collection and can often be improved using sample data. For example, the relevance feedback techniques described in Section 6 use a sample of relevant documents to improve the estimates at the query node that were originally based on the frequency of terms in the query.

It has been suggested that probabilistic models require samples of relevant documents to be useful,[25] but this claim is based on a narrow view of the nature of probability in which only frequency-based estimates are allowed. Estimates based on representative samples are preferred when they are available, but in many cases representative samples are not available, and techniques similar to those described above are required.

4. EXACT-MATCH MODELS

Exact-match retrieval models use matching functions that, given a query, partition the document collection into two sets, those that match the query and those that do not. Documents in the matching set are generally not ranked, although they may be ordered by date, alphabetically, or by some other criterion. Exact-match models are generally simple and efficient and form the basis of most commercial retrieval packages.

By far the most common exact-match model is the Boolean model. It is important to distinguish here between the use of Boolean operators in queries (which does not imply an exact-match model) and the use of Boolean logic as the interpretation of those operators. As we shall see, a Boolean query can be interpreted using Boolean logic under an exact-match model, using the probability operators of the last section, or using a distance function that is described in Section 5.1.

4.1 Boolean retrieval

In the Boolean model we have a set of binary-valued variables which correspond to features that can be assigned to documents. These features include text terms extracted from documents, but more complex features (e.g. dates, phrases, personal names or manually assigned descriptors) are sometimes used. A document is an assignment of truth values to the set of feature variables; all features which are 'correct' descriptions of document content are assigned *true* and all others are assigned *false*.

A query is a Boolean expression involving feature variables and the operators *and*, *or* and *not*.* The matching function is defined by the normal rules of Boolean logic. Any document that represents an assignment of truth values that satisfies the query expression is said to match the query, and all others fail to match.

If we restrict estimates for $P(r_i|d)$ to $\{0, 1\}$ the inference networks can be used to simulate Boolean retrieval precisely. If we allow probability estimates in the range $[0..1]$ they provide a natural interpretation of the semantics of Boolean operations in probabilistic terms. We first show how Boolean retrieval can be simulated and then how the probabilistic interpretation of the Boolean operations can be relaxed to produce document rankings.

Using the canonical link matrix forms of Section 3.4 we can simulate Boolean retrieval as follows. For clarity, we assume that the query and representation vocabularies are identical, so we can omit query concepts from the network.

(1) Use a canonical *or* matrix at each representation node. When a document is instantiated, all representation concepts to which it has been attached will have $\text{bel}(r_i) = 1$. All remaining representation concepts have $\text{bel}(r_j) = 0$.

(2) Build an expression tree for the query. The root of the tree is the query, and all arcs in the tree are directed towards the root. The leaves of this tree will be representation concepts and the interior nodes will correspond to expression operators. At each operator

* In nearly all commercial implementations, *not* is really an *and not* operator. Unary *not* and *or not* are rarely supported.

node use the canonical link matrix form for that operator. Attach this tree to the document network. A simple example for the query

(*information* and *retrieval*) or not *satellite*

is shown in Fig. 3.

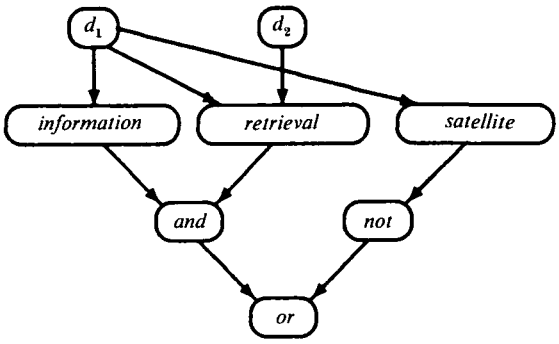

Figure 3. Inference network for (*Information* ∧ *retrieval*) ∨ ¬*satellite*.

(3) Using the evaluation procedure described in Section 3.3, instantiate each document in turn and record the belief in the query node. Any document for which bel(Q) = 1 satisfies the query, any document for which bel(Q) = 0 does not.

Under the assumptions above and using binary indexing, bel(Q) can only have values 0 or 1, and the inference network simulates conventional Boolean retrieval exactly.

The same probabilistic interpretation of the Boolean operators applies equally well to weighted indexing. Using the approach described in Section 3.5 we can incorporate indexing weights by replacing the *or* link matrix in the representation concept nodes with a *wtd_sum* matrix incorporating the appropriate weights. In this case, when a document is instantiated, all representation nodes to which it is attached take on the term weight for that term/document pair and all remaining representation nodes take on a node-specific default belief. These weights are then combined using the closed-form expressions of Section 3.4. In short, the term weights are interpreted as probabilities and are combined using the normal rules for negation and for disjunctive or conjunctive combination of sets in an event space. As a result, the inference network model provides a natural probabilistic interpretation of the Boolean operators and of indexing weights.

4.2 Extensions to the Boolean model

Boolean logic, by itself, is too weak for large full-text collections. Some linguistic structures that searchers find natural are represented awkwardly in the Boolean model (e.g. morphological variants) or cannot be represented at all (e.g. phrases). As a result, most commercial retrieval packages provide proximity operators and techniques for combining related word forms. Both of these extensions can be modelled naturally in an inference network.

Proximity operators are used most often to represent phrases and occasionally to represent higher-order structure (e.g. two phrases in the same paragraph). As an example, requiring that the terms *information* and *retrieval* occur within two words of each other is likely to find a better (smaller) set of documents dealing with information retrieval than would be obtained with the Boolean query '*information* and *retrieval*' which simply requires that the two terms occur in the same document. In terms of the inference network model, proximity operators create new representation concept nodes that are added to the network in response to the query. Most of these 'virtual' concepts are not created when the document network is built, but the underlying representation contains the information necessary to create them when needed. As a result, these proximity or phrase operators are fundamentally different from the canonical operators we described earlier, since they alter the structure of the inference network. See Ref. 7 for a more complete discussion of phrase representation in the inference network model.

The two principal techniques for handling morphological variants are stemming, in which variant forms are algorithmically mapped to a single form when the collection is built, and the use of 'wildcards', which allow users to specify query terms that will match several different document terms. Root forms that are produced by stemming can be built into the network as documents are added. Wildcards are handled much like phrases, and result in the creation of a virtual representation concept that exists only for the duration of query evaluation.

5. VECTOR SPACE MODEL

In the vector space model, documents and queries are represented as vectors in a k-dimension hyperspace where each dimension corresponds to a possible document feature. Vector elements may be binary-valued, but they are generally taken to be weights that describe the degree to which the corresponding feature describes the document or query. A weight of 0 is taken to mean that the corresponding feature does not describe a document or query, with weights greater than 0 representing the degree to which the feature describes the document. These weights are constrained to lie in some fixed interval, say [0..1], so that documents and queries represent points in a k-dimension hypercube. The matching function is then a distance metric that operates on the document and query vectors.

Bookstein[2] showed that the vector space model can be described in probabilistic terms. Croft[4] compared the vector space model with a probabilistic model and showed that under certain conditions they are equivalent. The inference network framework allows us to clarify the relationship between these two approaches to retrieval.

The most common distance metric used in the vector space model is the cosine of the angle between the document and query vectors, given by

$$\mathrm{sim}(Q, D_i) = \frac{\sum_{j=1}^{k} w_{qj} w_{d_{ij}}}{\sqrt{\left(\sum_{j=1}^{k} w_{qj}^2 \sum_{j=1}^{k} w_{d_{ij}}^2\right)}}, \qquad (6)$$

where w_{qj} is the weight assigned to feature j in the query vector and $w_{d_{ij}}$ is the weight assigned to feature j for the ith document D_i. The numerator of this matching

function is the inner product of the query and document vector elements, and the denominator is essentially a normalisation for document and query lengths.

The cosine matching function of Equation (6) is quite similar to the weighted-sum combining function of Equation (5). If the document weights are interpreted as probabilities, the functions are equivalent except for the normalisation employed. In the inference network model the normalisation is probabilistically motivated and results from conditioning over the set of query concepts. The inference network normalisation is independent of the specific document that is instantiated. In the vector space model the normalisation arises from the cosine definition and is dependent on the characteristics of each document. The cosine matching function could be simulated in the network model if we let the query weight of Equation (5) be

$$w_q = \frac{t}{\sqrt{\left(\sum_{j=1}^{k} w_{qj}^2 \sum_{j=1}^{k} w_{d_{ij}}^2\right)}},$$

which would effectively incorporate a length normalisation in the link matrix at the query node. While this kind of normalisation is certainly possible, it would violate our assumption that documents can influence belief in queries only through influence on representation concepts. It is conceptually and computationally simpler to build normalisations that depend on document length into our belief estimates for the representation concept nodes, and to build normalisations that depend on query length into the query node.

A wide variety of weights have been used for w_{qj} and $w_{d_{ij}}$ in Equation (6) (see Ref. 26 for a comprehensive review), which are generally based on information about the frequency with which each term occurs in the document and in the collection as a whole. Although the justification for these weights differs for the two models, the weights are computationally equivalent.

One of the most frequent criticisms of the vector space model is the implicit assumption that the set of document features (terms) forms an orthogonal basis for the vector space within which documents and queries are represented. For this assumption to hold we would expect that the distance between two documents that are identical except for a single term will be the same regardless of the identity of that term. This is similar to assuming the independence of terms in probabilistic models, but the failure of the assumption is somewhat more problematic for the vector space model – probability theory provides a rich set of tools for detecting and dealing with dependencies, whereas it is difficult to see how we would establish whether a set of terms does or does not form a basis.*

The Generalised Vector Space Model (GVSM)[36] replaces terms as the basic document features with a set of 'generalised' terms, each of which is represented as a Boolean expression that describes how the generalised term should be recognised in documents. The GVSM thus provides a mechanism for describing correlations between terms that is not possible in the basic vector space model. Conceptually, these new document features are equivalent to query concepts in the inference network model. The Boolean expression specifies the set of parent representation concepts and defines the form of the link matrix to be used at the query concept node.

* The argument here is not whether the set of term vectors has the desired mathematical properties. Clearly, the set of n unit vectors spanning an n-dimension vector space forms an orthogonal basis. The question is whether this model of distance matches our intuition.

5.1 Extended Boolean model

The binary nature of the retrieval decision in Boolean systems is frequently cited as a drawback.[5,29,24,14] Intuitively, we would like a document containing all but one term of an n-term *and* to be judged nearly as likely to match the query as a document containing all n terms and substantially more likely to match than a document containing none of the terms.

One well-known model which supports weighted Boolean retrieval is the extended Boolean or p-norm model.[24,28] In this model, the similarity between a document (represented as a vector $D = (d_1, d_2, \ldots, d_n)$) and a query Q consisting of the conjunction or disjunction of all terms in the vector is given by

$$\text{sim}(D, Q_{\text{and}}) = 1 - \left[\frac{(1-d_1)^p + (1-d_2)^p \ldots + (1-d_n)^p}{n}\right]^{\frac{1}{p}}$$

$$\text{sim}(D, Q_{\text{or}}) = \left[\frac{d_1^p + d_2^p \ldots + d_n^p}{n}\right]^{\frac{1}{p}}.$$

These similarity functions represent a measure of the distance between the document and a point corresponding to the presence of all query terms (for *and*) or the presence of no query terms (for *or*). Thus, for *and*, the similarity is at a maximum for a document containing all terms, and decreases with increasing distance. For *or*, the similarity function is at a minimum for a document containing none of the query terms, and increases with increasing distance. The parameter p is an arbitrary constant in the range $1 \leqslant p \leqslant \infty$. The best value is determined empirically for a collection, but is generally in the range $2 \leqslant p \leqslant 5$.

When document and query weights are restricted to 0 and 1 and $p = \infty$ the p-norm model, like the inference network model, simulates normal Boolean logic. For $p = 1$, the p-norm model behaves much like the normal vector space model. For intermediate values of p, the p-norm model gives a 'soft' Boolean matching function.

Experiments have shown that the inference network model using the link matrix forms of Section 3.4 performs at least as well as the p-norm model[34] and does not require determination of any parameter values. However, in order to clarify the relationship between the p-norm and inference network models we shall describe link matrix forms that give performance much like that of the p-norm model.

The link matrix forms of Section 3.4 assert complete certainty given the evidence available at a node, that is

$$P(Q_{\text{and}} = \text{true} | \text{all parents} = \text{true}) = 1$$
$$P(Q_{\text{or}} = \text{true} | \text{any parent} = \text{true}) = 1$$
$$P(Q_{\text{not}} = \text{true} | \text{parent} = \text{true}) = 0.$$

We can easily relax this interpretation of the probabilistic semantics of the Boolean operators if we choose a value

$n \leqslant c \leqslant \infty$, where n is the number of parents at a given node and interpret the *and* operator to mean

$$P(Q_{\text{and}} = \text{true}|n \text{ parents} = \text{true}) = 1$$
$$P(Q_{\text{and}} = \text{true}|k \text{ parents} = \text{true}) = \frac{k}{c}, \quad 0 < k < n$$
$$P(Q_{\text{and}} = \text{true}|\text{no parents} = \text{true}) = 0$$

and the *or* operator to mean

$$P(Q_{\text{or}} = \text{true}|n \text{ parents} = \text{true}) = 1$$
$$P(Q_{\text{or}} = \text{true}|k \text{ parents} = \text{true}) = 1 - \frac{n-k}{c}, \quad 0 < k < n \tag{7}$$
$$P(Q_{\text{or}} = \text{true}|\text{no parents} = \text{true}) = 0$$

Under this interpretation, when $c = \infty$ the operators have their normal Boolean interpretation. As c decreases, our belief in Q depends increasingly on the number of parents that are true. When $c = n$ the distinction between *and* and *or* has disappeared, the link matrices for both operators are the same, and both are equivalent to the *sum* link matrix of Section 3.4. Since a node implementing the *not* operator has exactly one parent, its interpretation is unchanged. These interpretations for the Boolean operators can be implemented as canonical link matrices requiring $O(1)$ space and operating in $O(n)$ time.

When $c = n$, $p = 1$ and $d_i \in \{0, 1\}$ the p-norm and relaxed inference network models produce the same results. For a conjunctive query containing n terms, m of which are true, under the p-norm model we have

$$\text{sim}(D, Q_{\text{and}})_{p=1} = 1 - \left[\frac{(1-d_1)^p + (1-d_2)^p \ldots + (1-d_n)^p}{n}\right]^{\frac{1}{p}}$$
$$= \frac{m}{n}, \quad 1 \leqslant m \leqslant n$$

and for the network model we have

$$P(Q_{\text{and}} = t|m \text{ parents true}) = \frac{m}{n}, \quad 1 \leqslant m \leqslant n.$$

For a disjunctive query using the p-norm model we have

$$\text{sim}(D, Q_{\text{or}})_{p=1} = \left[\frac{d_1^p + d_2^p \ldots + d_n^p}{n}\right]^{\frac{1}{p}}$$
$$= 1 - \frac{n-m}{n},$$

and using the network model we have

$$P(Q_{\text{or}} = t|m \text{ parents true}) = 1 - \frac{n-m}{n}.$$

Similarly, when $c = p = \infty$ both models produce the same results

$$\text{sim}(D, Q_{\text{and}})_{p=\infty} =$$
$$P(Q_{\text{and}} = t|m \text{ parents true}) = \begin{cases} 1 & \text{if } m = n \\ 0 & \text{otherwise} \end{cases}$$

$$\text{sim}(D, Q_{\text{or}})_{p=\infty} =$$
$$P(Q_{\text{or}} = t|m \text{ parents true}) = \begin{cases} 0 & \text{if } m = 0 \\ 1 & \text{otherwise} \end{cases}$$

For values $n < c < \infty$ and $1 < p < \infty$, both functions are monotonic in the number of true parents (number of non-zero vector elements) m, but they are not equivalent since $P(Q = t)$ is linear in m while $\text{sim}(D, Q)$ is not. Note that we could choose to redefine our probability function to produce equivalent results. If, for example, we used

$$c = \sqrt[p]{\frac{n-m}{n}}$$

for *or* operations, the models would produce results that are equivalent in the sense that, for any value of p we can find a corresponding value of c that produces the same values for $\text{bel}(Q)$ and $\text{sim}(D, Q)$. There is, however, no theoretical basis for this redefinition.

The inference network model handles weighted indexing as a natural extension and is again equivalent to the p-norm model for $p = 1$ and $p = \infty$ and is similar but not equivalent for $1 < p < \infty$.

6. RELEVANCE FEEDBACK

Relevance feedback is the name given to a process where, based on user feedback about documents retrieved in an initial ranking, the system automatically modifies the description of the information need (i.e. the query) and produces a new document ranking. User feedback usually takes the form of simple yes/no judgements on the relevance of documents, although more detailed feedback about concepts in those documents is also possible.[6,12] Relevance feedback is a well-established and effective technique in IR, and any discussion of retrieval models would be incomplete without it.

Much of the original work on feedback was done in the context of the vector space model,[27,29] so it is appropriate to present it first in these terms. The general method for producing a new query given an old query and relevance judgements is as follows:

$$Q_{\text{new}} = Q_{\text{old}} + \beta \sum_{\text{rel}} \frac{D_i}{|D_i|} - \gamma \sum_{\text{nonrel}} \frac{D_i}{|D_i|}, \tag{8}$$

where the summation is taken over the known relevant and nonrelevant documents, and the D_i represent document vectors. A particularly common form of this query modification is known as 'dec hi', where weighted terms are added directly to the queries, and only one nonrelevant document is used, i.e.

$$Q_{\text{new}} = Q_{\text{old}} + \sum_{\text{rel}} D_i - D_{\text{topnonrel}}. \tag{9}$$

In the vector space model, then, relevance feedback involves changing weights associated with query terms and adding new terms to the original query.

In the probabilistic model described by Robertson and Sparck Jones[21] and Van Rijsbergen,[35] relevance feedback is described in different terms. In this model, documents are ranked using a (generally) linear discriminant function in which each term corresponds to a representation concept in the collection. Typically, only the representation concepts found in the query have non-zero values, and the coefficients of these terms are estimated using some model-specific function. A representative function is

$$g(d) = \sum_i \left(\log \frac{p_i}{1-p_i} + \log \frac{1-q_i}{q_i}\right), \tag{10}$$

where p_t is the probability that term i occurs in a relevant document and q_t is the probability that term i occurs in a non-relevant document. The second term in the summation is typically estimated using each term's *idf*, and the first term is based on the characteristics of the set of known relevant documents. This term is initially estimated using a fixed value (e.g. $p_t = 0.5$) or a value based on the frequency of the term in the query.

In relevance feedback, we are given a sample of documents that have been judged relevant and we re-estimate our linear discriminant function based on this sample. This involves computing a new set of p_t values for Equation (10) based on the relevant sample and adding the top n relevant document terms (according to some measure) to the original query terms. Addition of new terms from the relevant documents was not done in early experiments with the probabilistic model, although it has been investigated subsequently. The probabilistic model does, in fact, indicate that the addition of these terms, up to a point, should improve performance. Comparisons of the effectiveness of relevance feedback using the vector space and probabilistic models are difficult to make due to the large number of parameters involved, although Salton and Buckley[27] suggest that the best vector space approaches have an advantage.

Although the inference net model is a probabilistic model, there are differences from earlier models.[32] In particular, because this model does not use Bayesian inversion, there are no probabilities that correspond directly to p_t and q_t. This means that feedback in the inference net model is not the same as in the model described by Robertson and Sparck Jones. There are two basic ways in which feedback can be incorporated in an inference network model: adding evidence or altering the dependencies represented in the network. The two approaches are fundamentally different. Adding evidence always leaves the probability distribution represented in the network unchanged, but alters beliefs in the network to be consistent with that distribution. Altering the dependencies, either by changing the topology of the network or by altering the link matrices, changes the underlying probability distribution, which in turn alters belief. The use of evidence is appropriate when we known that the distribution is 'correct' (if, for example, the topology is known and the link matrices have been learned from a reliable sample). Evidential feedback is appropriate in the document network, which is largely determined by the characteristics of the collection. Frisse and Cousins[8] use this approach to implement feedback in a hierarchy of index terms associated with a hypertext medical handbook.

Altering dependencies is appropriate when the initial network is known to be an approximation to the correct distribution and we obtain better information about the nature of the true distribution. This is the approach we use to change the query network in response to user relevance judgements. It can also be used in document space modification,[10,37] where we use a set of queries and relevance judgements to learn the 'correct' distribution for documents and representation concepts.

In the network model, queries are represented by links between representation concept nodes, and the query-node and query-term weights are represented using a weighted-sum form of the link matrix at the query node (Equation (5) in Section 3.4). The basic relevance feedback strategy of adding terms to the query and recalculating weights is implemented in the inference net by adding links between the query node and the representation concepts to be added, and re-estimating the link matrix weights based on the sample of relevant documents rather than on the query text. The fact that we are explicitly modifying the query means that the inference net model can accurately simulate the vector space approach to feedback. Since this is a probabilistic model, however, it should be possible to say what probabilities are being re-estimated during feedback instead of talking about changing weights. This issue is clarified if we consider the canonical link matrix associated with the weighted-sum operation, which is what is used to simulate vector space feedback. The weight associated with a query term (for example, w_a in Equation (5)) is used to estimate the probability that an information need is satisfied given that a document is represented by that term. Simple relevance feedback in the inference net model, then, involves re-estimation of that probability instead of the p_t probability in earlier models.

As an example, consider the 'dec hi' feedback strategy used in the vector space model (Equation (9)). To implement this strategy in the inference network, we would first calculate the set of terms in the modified query and the weights associated with those terms (as is done in the vector space implementation). Conceptually, any new terms are added as new parents of the query node, and the link matrix is changed to reflect the presence of these new terms and the new weights (using the canonical form in Equation (5)). In practice, this would mean that document rankings would be calculated using Equation (5), the inverted lists associated with the terms in the augmented query, and the new query-term weights.[33]

6.1 Feedback with structured queries

A number of models have been proposed for using relevance feedback with Boolean retrieval systems.[18,19,23,24] While some of these models have been shown to improve performance significantly when compared to conventional Boolean retrieval, they are not attractive in the context of the network model. These models generally adapt probabilistic relevance feedback techniques to estimate weights for terms in very restricted Boolean query forms (e.g. disjunctive normal form with no negation and *and* terms containing at most three representation concepts). Since these models do not make use of any linguistic or domain knowledge, it is unlikely that they will afford performance gains that cannot be achieved with normal probabilistic relevance feedback.

The development of an effective relevance feedback mechanism for Boolean and structured queries[7] is a potentially important area for further research. Encoding feedback information in a structured query could improve performance more than a simple query, since it is possible to encode information in the structured query that is not representable in the *sum* expression. A feedback mechanism for structured queries could be implemented with the following procedure.

(1) Identify important words and phrases in the sample

of relevant documents. Important words can be identified using the same techniques as for probabilistic feedback, but phrases are a harder problem. Work on the identification of syntactic or statistical phrases should be of use here,[7] and it may be useful to ask the user about the importance of candidate words and phrases.

(2) Identify words and phrases whose absence from the set of relevant documents is important. This is a hard problem, since the sample of relevant documents is usually far too small to allow meaningful statistical estimates. An expert system may be of use here if the domain of interest is known, but for the near term the user is probably the only reliable source of information about concepts that should not be in the relevant set.

(3) Estimate weights for the concepts identified above, probably using both statistical analysis and user input.

(4) Assemble a query. An obvious approach would be to use *and* or a proximity operator to define phrases,[7] use *or* to assemble sets of concepts that should and should not be in relevant documents, and then use *and not* to combine the two sets of concepts.

A number of variations are possible. In particular, initial work suggests that the probabilistic *sum* operator might be more appropriate than the *or* in step (4). This would represent a hybrid approach, in which we start with the basic relevance feedback query and selectively add structure in the form of phrases and negation.

The main reason for considering the use of structured expressions for feedback is to increase the expressive power of the feedback query without incurring the expense of learning a complete link matrix. Clearly, once the set of representation concepts is known we could learn the appropriate link matrix given a large enough sample of relevant documents. Unfortunately, we shall rarely have a large enough sample, and the computational costs associated with learning a 'correct' matrix would be prohibitive if more than a few terms were involved.

In summary, we see that the inference net model can capture the approaches to relevance feedback that have been used in vector space and Boolean models. The ability to express complex query structure using networks opens up the possibility for new directions in research on learning and relevance feedback.

7. CONCLUSION

A clear understanding of the retrieval models underlying current IR systems is important if we are to understand differences in their retrieval and computational performance. We have shown that the differences between current-generation retrieval models can be explained as different ways of estimating probabilities in the inference network model. Given the inference network framework we can now develop estimation procedures that use the best features of the exact-match, vector space and probabilistic models. Further, we can consider IR systems that can alter their performance to better match users and their information needs. For some kinds of searching (e.g. known-item searching) the deterministic and predictable exact-match models perform very well, while for more general searching or for untrained users the vector space and probabilistic models have clear performance advantages.

REFERENCES

1. P. Bollmann and V. Raghavan, The axiomatic approach for theory development in IR. In *NSF Workshop on Future Directions in Text Analysis, Retrieval and Understanding*, pp. 16–22 (1991).
2. A. Bookstein, Explanation and generalization of vector models in information retrieval. In *Proceedings of the 6th Annual International Conference on Research and Development in Information Retrieval, New York, 1983*, pp. 118–132.
3. A. Bookstein, Set-oriented retrieval. *Information Processing and Management*, **25** (5), 465–475 (1989).
4. W. Bruce Croft, A comparison of the cosine correlation and the modified probabilistic model. *Information Technology*, **3** (2), 113–114 (1984).
5. W. Bruce Croft, Boolean queries and term dependencies in probabilistic retrieval models. *Journal of the American Society for Information Science*, **37** (2), 71–77 (1986).
6. W. Bruce Croft and Raj Das, Experiments with query acquisition and use in document retrieval systems. In *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, edited J.-L. Vidick, pp. 349–368 (1990).
7. W. Bruce Croft, H. R. Turtle and D. D. Lewis, The use of phrases and structured queries in information retrieval. In *Proceedings of the Fourteenth International Conference on Research and Development in Information Retrieval*, pp. 32–45 (1991).
8. M. E. Frisse and S. B. Cousins, Information retrieval from hypertext: update on the dynamic medical handbook project. In *Hypertext '89 Proceedings*, pp. 199–212 (1989).
9. N. Fuhr, Models for retrieval with probabilistic indexing. *Information Processing and Management*, **25** (1), 55–72 (1989).
10. N. Fuhr and C. Buckley, Probabilistic document indexing from relevance feed-back data. In *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, edited J.-L. Vidick, pp. 45–61 (1990).
11. H. Garcia-Molina and D. Porter, Supporting probabilistic data in a relational system. In *Proceedings of EDBT*, pp. 60–74 (1990).
12. D. Harman, Towards interactive query expansion. In *Proceedings of the 11th International Conference on Research and Development in Information Retrieval*, edited Y. Chiaramella, pp. 321–332 (1988).
13. D. D. Lewis, Representation and Learning in Information Retrieval. PhD thesis, University of Massachusetts at Amherst (1991).
14. R. M. Losee and A. Bookstein, Integrating Boolean queries in conjunctive normal form with probabilistic retrieval models. *Journal of the American Society for Information Science*, **24** (3), 315–321 (1988).
15. A. Motro, VAGUE: a user interface to relational databases that permits vague queries. *ACM Transactions on Office Information Systems*, **6** (3), 187–214 (1988).
16. R. E. Neapolitan, *Probabilistic Reasoning in Expert Systems*. Wiley, New York (1990).
17. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA (1988).
18. T. Radecki, Incorporation of relevance feedback into Boolean retrieval systems. In *Research and Development in Information Retrieval*, edited G. Salton and H. J. Schneider, pp. 133–150 (1983).
19. T. Radecki, Probabilistic methods for ranking output documents in conventional Boolean retrieval systems. *Information Processing and Management*, **24** (3) 281–302 (1988).

20. S. E. Robertson, M. E. Maron and W. S. Cooper, Probability of relevance: a unification of two competing models for document retrieval. *Information Technology: Research and Development*, **1** (1), 1–21 (1982).
21. S. E. Robertson and K. Sparck Jones, Relevance weighting of search terms. *Journal of the American Society for Information Science*, **27**, 129–146 (1976).
22. S. Ross, *A First Course in Probability*. Macmillan, New York, (1988).
23. G. Salton, E. Voorhees and E. A. Fox, A comparison of two methods for Boolean query relevancy feedback. *Information Processing and Management*, **20** (5/6), 637–651 (1984).
24. G. Salton, A simple blueprint for automatic Boolean query processing. *Information Processing and Management*, **24** (3), 269–280 (1988).
25. G. Salton, Developments in automatic text retrieval. *Science*, **253**, 974–980 (1991).
26. G. Salton and C. Buckley, Term weighting approaches in automatic text retrieval. *Information Processing and Management*, **24** (5), 513–523 (1988).
27. G. Salton and C. Buckley, Improving retrieval performance by relevance feedback. *JASIS*, **41**, 288–297 (1990).
28. G. Salton, E. Fox and H. Wu, Extended Boolean information retrieval. *Communications of the ACM*, **26** (11), 1022–1036 (1983).
29. G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill, New York (1983).
30. K. H. Stirling, The effect of document ranking on retrieval system performance: a search for an optimal ranking rule. *Proceedings of the American Society for Information Science*, **12**, 105–106 (1975).
31. H. Turtle, Inference Networks for Document Retrieval. PhD thesis, Computer Science Department, University of Massachusetts, Amherst, MA 01003 (1990). (Available as COINS Technical Report 90-92.)
32. H. Turtle and W. Bruce Croft, Inference networks for document retrieval. In *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, edited J.-L. Vidick, pp. 1–24 (1990).
33. H. Turtle and W. Bruce Croft, Efficient probabilistic inference for text retrieval. In *RIA091 Conference Proceedings*, pp. 644–661 (1991).
34. H. Turtle and W. Bruce Croft, Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, **9** (3), 187–222 (1991).
35. C. J. van Rijsbergen, *Information Retrieval*. Butterworths (1979).
36. S. K. M. Wong, W. Ziarko, V. V. Raghavan and P. C. N. Wong, On extending the vector space model for Boolean query processing. In *Proceedings of the 9th International conference on Research and Development in Information Retrieval*, pp. 175–185 (1986).
37. C. T. Yu and H. Mizuno, Two learning schemes for information retrieval. In *Proceedings of the 11th International Conference on Research and Development in Information Retrieval*, edited Y. Chiaramella, pp. 201–218 (1988).
38. S. B. Zdonik and D. Maier, *Readings in Object-Oriented Database Systems*. Morgan Kaufmann, San Mateo, CA (1990).

Book Review

ERIC DAVALO and PATRICK NAIM
Neural Networks
Macmillan Education Ltd, Houndmills, Basingstoke, Hampshire RG21 2XS
£13.99. 0-333-549961

This is another book in the Macmillan Computer Science Series, which sets out to provide the reader with a basic understanding of the subject – in this case neural networks – and with one or two minor reservations it achieves this.

The book is structured logically to provide the reader with the background followed by some basic principles, and finishes with more detail on some common models and applications.

Upon first reading, I had to ask myself 'who is the target reader?'. There are many books addressed at the neural network professional with which this book does not compete. There are, however, many more readers to whom such a book would be irrelevant and incomprehensible. I suspect the target reader is a professional in the computer or related fields who needs or wants to know more about neural networks because he or she has a problem to solve. It is also likely that this book could form the basis of an undergraduate course in neural networks.

The first chapter examines the biological foundations for neural networks by describing in simple terms the way the brain and its cells are believed to function. This is followed by an interesting description of the peculiarities of the visual system of a frog and mammals.

Chapter 2 explains the historical basis and basic principles of neural models, introducing the Perceptron and the Hebbian learning rule. This is followed by a discussion of multi-layer neural networks and in particular the back-propagation model. The authors consider the use of this particular model and describe its positive attributes and limitations. This is adequately illustrated with descriptions of applications for back-propagation including the classical 'XOR' problem.

Two further models are described in some detail; these are the Hopfield model and the Kohonen model. The theory of the Hopfield model is examined followed by examples of its use, for example in the travelling salesman problem. A further derivative of the Hopfield model is introduced, namely the Boltzmann machine, which utilises an analogy of annealing in metals called simulated annealing to help the network find a 'better' solution. The Kohonen model, although not as well known as back-propagation and Hopfield, is considered in some detail, along with examples of its use.

Probably the most interesting chapters from the practitioner's point of view are the ones on applications of neural networks and neural computers. They look at the reasons for using neural networks and their limitations, and the characteristics of problems which would be suitable for solving by using neural networks. This is followed by descriptions of specific application examples, with one examined in more detail. The chapter on neural computers only covers the area superficially, but does provide an insight into how such computers work.

The two appendices which describe the back-propagation model and the Kohonen model were a disappointment, they just provide the barest minimum of the mathematical basis for the models. Considering the book appears to be aimed at the practitioner, the engineer or the programmer who wants to use and explore neural networks, I was disappointed to find that there was no description of the algorithms for these models which could be used as the basis of exploration. It is left to the practitioner to derive these or to look elsewhere.

Despite the slight reservations, I would have no hesitation in recommending this book as the starting point for the computer (or other) professional who feels the need to know something about neural networks or who wants to start using and exploring neural networks.

M. A. TUREGA
Manchester

Erratum

We have been advised that there was a typesetting error in the letter by P. A. Kalinchenko, **34** (6), 502 (1991). Within the subroutine PPOLYN, in the third column, line 15, the statement should read

```
IF (ISTART+I) 9, 2, 9
```

and not

```
IF (ISTART+1) 9, 2, 9.
```

We apologise to the author, and readers, for the error.