A GENERAL MATHEMATICAL MODEL FOR INFORMATION RETRIEVAL SYSTEMS

Abraham Bookstein and William Cooper

This paper presents a mathematical model of an information retrieval system thought to be general enough to serve as an abstract representation of most document and reference retrieval systems. The model is made up of four components that, in one form or another, appear in every functioning system. It is proved that the basic organization of documents that the system provides for a user on receipt of a request follows from the properties and interrelations of the four components. Each component is then discussed in turn and it is seen that much of the existing theory regarding information systems can be viewed as an elaboration of this model.

Introduction

Information retrieval systems (by which will be understood here reference and document retrieval systems as opposed to data retrieval systems) have frequently been compared to communication systems in which an author communicates with a patron of a system by means of a document [1, chap. 1]. The patron, perhaps anticipating the retrieval mechanisms of the system, translates a felt need for information into a verbal request which he brings to the system; by means of the request, perhaps further transformed, and a processed version of the document collection (the "document surrogates" or "index records"), the system predicts the extent to which each document is "relevant" to the user, that is, would be judged by the user to be an appropriate response to his request. The system then acts as a switching mechanism, connecting its patrons to those documents likely to be of interest to them.

For a given request and a given document, different systems may have any of a number of different response capabilities. For example, some systems

[Library Quarterly, vol. 46, no. 2, pp. 153–67]

simply respond by predicting whether or not the document will be found relevant by the patron; at the opposite extreme a system may attempt to assign to the document a number indicating its degree of predicted relevance, or its probability of relevance, to the request [2]. The end result of this process is that the documents are given a structure that guides the user in his search for information; the set of documents is broken up into subsets, and these subsets are arranged in order of predicted relevance to the request as perceived by the system [3]. The patron is in effect advised to search the collection by first examining the documents in the highest ranking set, then those in the next set, and so on until either his need for information is satisfied or else he gives up. The partitioning of the collection, permitting a search that is more efficient than a random search of the collection, is the fundamental service provided by the system for its users.

We shall here provide a formal model of such a system. This model will abstract from functioning systems those mathematical characteristics that determine how these systems provide a structure for a set of documents in response to a request; it will also indicate, precisely, which features of the system are essential for that structure to be imposed. Models such as this are useful in that they help provide a unified view of a wide variety of systems that superficially differ considerably from each other, thereby contributing to the establishment of a theoretical foundation for the study of these systems.

In the first part of the paper, then, the model will be precisely defined and it will be shown that some of the well known characteristics of information retrieval systems follow necessarily from a small number of assumptions. By making explicit, in terms of our model, the assumptions underlying properties commonly accepted as characteristic of information retrieval systems, we hope to provide a necessary but often neglected stage of analyses of such systems. In the second part of the paper we deal more fully with each component of our model in order to show the wide range of systems it can describe. The types of systems described in this section of the paper are all well known; what we show is that these systems can be described precisely in terms of a uniform vocabulary provided by our model.

Definition of Model

On the basis of the above discussion, we formally define (see Appendix) an information retrieval system S as a quadruple $S = (I, R, V, T)$, where:

I = set of all possible "index records" that might be created by the system, each index record potentially being a surrogate for a document; these index records represent documents for the purpose of manipulation in the system. A specific collection of documents is represented by a subset of I.

R = set of "search prescriptions" that are recognized by the system. A search prescription is the system's version of the request for information as submitted by a patron. It may take the form of the patron's request or it may

require modification. For the purpose of conciseness, we shall assume that users offer the system requests directly in a form it can manipulate, and accordingly we will use the term "request" to refer both to the verbal statement offered to the system by a patron and the final search prescription.

V = set of "retrieval status values"—a simply ordered set (see Appendix) whose elements represent the possible decisions the system is capable of making. Though these values can be implemented in many ways in an operating system, we here consider them as constituting an abstract set structured by means of an order relation. We are interested in how this order relation is used to impose a structure on the set of documents (or more precisely, on I).

T = "retrieval function"—a function taking points in $R \times I$ to points of V. This function formalizes the retrieval mechanism of the system. For each request and each index record it determines a value from V, indicating the degree to which it is predicted that the document represented by the index record will be found relevant to the request by the patron. Alternatively, for each request, $r \epsilon R$, T defines a function $T_r: I \rightarrow V$.

In the following we shall prove that T_r induces a structure on I that reflects the order relation defined on V, and that this structure can be used to guide the patron in his search for documents. Some readers may wish to skip much of the rest of this section and continue with its summary.

Induced Structure on I

Let $\leq$ denote the simple order relation on V [4, pp. 208–23]; thus for $v_1, v_2 \epsilon V$ it is meaningful to say $v_1 \leq v_2$, or "v_1 precedes v_2." For a given request, r, let us define a relation, $\leq'$, on I as follows: if $i_1, i_2 \epsilon I$, then $i_1 \leq' i_2$ if $T_r(i_1) \leq T_r(i_2)$. The relation $\leq$ induces a number of properties on $\leq'$.

PROPERTY 1: $\leq'$ is reflexive.

Proof: For any $i \epsilon I$, $i \leq' i$ if $T_r(i) \leq T_r(i)$. But this is always so because $\leq$ is reflexive.

PROPERTY 2: $\leq'$ is transitive.

Proof: Suppose, for $i_1, i_2, i_3 \epsilon I$, $i_1 \leq' i_2$ and $i_2 \leq' i_3$. We want to show $i_1 \leq' i_3$. We know $T_r(i_1) \leq T_r(i_2)$ and $T_r(i_2) \leq T_r(i_3)$ by the definition of $\leq'$. Since $\leq$ is a simple order, it is itself transitive, and accordingly $T_r(i_1) \leq T_r(i_3)$. But this means $i_1 \leq' i_3$, as was to be proven.

PROPERTY 3: $\leq'$ is connected.

Proof: Let i_1 and i_2 be arbitrarily chosen from I. We want to show that if $i_1 \nleq' i_2$, then $i_2 \leq' i_1$. But if $i_1 \nleq' i_2$, then $T_r(i_1) \nleq T_r(i_2)$. Since $\leq$ is connected, this implies $T_r(i_2) \leq T_r(i_1)$. By the definition of $\leq'$, this means $i_2 \leq' i_1$, as was to be shown.

(We note that it is not necessarily the case that $\leq'$ is antisymmetric. For consider i_1 and i_2 where $T_r(i_1) = T_r(i_2)$ and $i_1 \neq i_2$. Then, since $\leq$ is reflexive, $i_1 \leq' i_2$ and $i_2 \leq' i_1$.) We shall call a relation that is reflexive, transitive, and connected a weak order. We have thus shown that the retrieval function, T_r,

from I to V induces a weak order on I (and more generally, any function $f{:}D \rightarrow R$, where R is simply ordered, induces a weak order on D).

Consequences of Weak Order

In this section we shall examine some properties of the set of index records, I, that result from the weak order defined on I. Although we will be proving these properties for I, it is clear that they are valid on any weakly ordered set.

Suppose a weak order ($\leq'$) is defined on a set I. Then we consider two points of I, a and b, as equivalent if $a \leq' b$ and $b \leq' a$. For any point $d \epsilon I$ the set S_d is defined as the set containing d and all points equivalent to d. It is easy to show that following properties hold:

PROPERTY 4: All points in any of the sets S_d are equivalent to each other.

Proof: Let d_1 and d_2 be any two points in S_d. Then $d_1 \leq' d$ and $d \leq' d_2$. Transitivity implies $d_1 \leq' d_2$. Similarly $d_2 \leq' d_1$, so d_1 and d_2 are equivalent.

PROPERTY 5: Every element of I is in one and only one set.

Proof: The point d is certainly in S_d, so we have only to show that each point is in no more than a single distinct set. Suppose $d \epsilon S_{d_1}$, and also S_{d_2}. If d_3 is any point in S_{d_2} then $d_3 \leq' d$ and $d \leq' d_3$ by the preceding property. But since $d \epsilon S_{d_1}$, then $d \leq' d_1$ and $d_1 \leq' d$. By the transitivity of $\leq'$, this means $d_3 \leq' d_1$ and $d_1 \leq' d_3$, so $d_3 \epsilon S_{d_1}$. Since d_3 was an arbitrary member of S_{d_2}, this implies that $S_{d_2} \subseteq S_{d_1}$. Similarly $S_{d_1} \subseteq S_{d_2}$, so $S_{d_1} = S_{d_2}$, which was to be shown.

We conclude that a weak order relation, $\leq'$, uniquely and "naturally" breaks the set on which it is defined into subsets such that each point is in one and only one set.

We next define a new set and a new relation. The new set, D, has as its elements the previously defined subsets of I; the new relation, $\leq''$, is defined as follows: For D_1, $D_2 \epsilon D$, then $D_1 \leq'' D_2$ if there are points $a \epsilon D_1$ and $b \epsilon D_2$ such that $a \leq' b$. The reader can show by arguments similar to those used above that this relation is a simple ordering.

Summary

We can translate these general results in terms of any information retrieval system as follows: Given a request, r, the function T_r uniquely breaks the set of index records, I, into a set of subsets, and induces a simple ordering on these subsets. This structure, in turn, breaks the set of documents in a collection into an ordered set of subsets. This property can be used in a search of the collection by means of the following idealized search procedure. To search the collection a system user begins by examining the "first" subset of documents; if he wishes, he continues by searching the "next" subset of documents, etc. Each subset is searched randomly. The system operates by reducing a random search of the full collection to random searches of the much

smaller subsets defined above; these subsets are presumed to be enriched with relevant documents. In the context of our abstract model, it remains for us to demonstrate that the above procedure is always possible. To do this, it is sufficient to show that there always is, in fact, a first, second, etc., point in the simply ordered sets with which we deal. But since our set of documents is finite, it is readily demonstrated that this is indeed the case.

The search procedure outlined above seems to be implicit in most information retrieval systems. The model makes precise how such a procedure depends on an assumed structure underlying the total system. It does not, however, permit comment on the advisability of such a procedure. To do this it is necessary to introduce decision theory concepts as in [5, 6, and 7]. These papers indicate when the idealized search described here may be considered as optimal.

Elaboration of Model

There has been much interest among information scientists in providing a unifying theoretical foundation for their practice. Certainly a model that makes explicit what, in essence, an information system is doing must be a part of this foundation. But if that model is to be at all attractive, it should provide a natural basis for discussing topics felt to be significant in the field. That the model described here provides such a base can best be shown by considering in greater detail each of its components in turn.

Index Records

The index record is a transformed version of the document. While there is an information loss in going from the full document to the index record, just as the document is itself an imperfect expression of what the author intended to communicate, this loss is currently a necessary cost of creating an entity that the system can efficiently manipulate in the course of a search. Yet, as the model created in the paper emphasizes, the system cannot make any decision as to the probable relevance of a document to a request that requires information not available on the index record.

Anything the system operates upon directly in response to a request can be considered an index record—for example, the document itself if the system is capable of taking into account all the information contained therein. In a typical information system, however, an index record is made up of "index terms" chosen to represent the content of a document; in this representation most of the information contained in the grammatical and logical interconnections among the words in a document is removed. Similarly, requests are also usually represented as a set of index terms. Once the index records have been created, the system searches for relevant documents by means of a se-

quence of steps that includes the matching of the index terms of the records with corresponding terms representing a request. So far as this model is concerned, the index terms can be considered to be a set of abstract marks, some of which are assigned to each document. To describe fully any specific indexing language, at least three aspects of the language must be discussed:

1. Syntax: rules by which the marks are constructed. In some cases these marks are identical to the customary representation of natural language words and phrases, which, when they function as indivisible units, are sometimes called "descriptors." In other cases an abstract notation is used. A more complex index term may be formed by associating weights with descriptors, or by using other devices, such as roles or subheadings, which provide a structure for the terms and can be taken advantage of in the search procedure. Which is the case is determined by what is convenient for the user, and, in part, by the structure of the language. For our purposes, it is important to recognize that ultimately the syntax creates a set of "legitimate" marks, or index terms, a subset of which may define a document for the purposes of the system. A more detailed discussion of possible implementations may be found in textbooks such as [8].

2. Structure: the relationship between index terms. This structure may be *extrinsic* in that it is made explicit in a schedule or thesaurus provided by the group responsible for the construction and maintenance of the index language, or it may be *intrinsic* in that it is determined by how the terms are in fact applied to documents. The extrinsic structure most commonly takes the form of a set of trees [9], sometimes called a forest. At one extreme we have the hierarchical schemes, such as the Dewey Decimal classification, whose

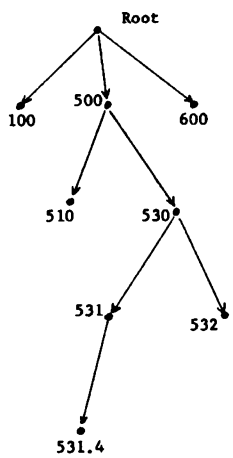

FIG. 1.—Tree structure illustrated by Dewey Decimal classification. Each node represents a class member, and includes the classes represented by the nodes it points to.

structures can formally be described as a single tree (see fig. 1). In such cases the notation tends to be abstract, and in fact is a symbolic representation of the path between the root of the tree and an index term at a node.

A faceted scheme can usually be described as a set of trees, with the index term being a collection of path descriptions, one from each tree or facet. If a facet is "omitted" from a particular expression, it can be considered as equivalent to choosing the root of the tree corresponding to that facet, the root being an extra node added for this purpose (see fig. 2). As just described, a faceted scheme has no implications regarding the order of the facets; an index record for a document is a *set* of nodes, and only the choice of nodes is of concern. This can be considered a strength of such a scheme since it doesn't impose on a patron the order felt important by the creators of the scheme. This freedom from a predetermined ordering overcomes the problem of distributed relatives: if a patron is interested in a term taken from the second facet, he need not consider each term from the first facet under which it may have appeared; he would simply enter the term into the system as a request. Unfortunately such a system is difficult to implement.

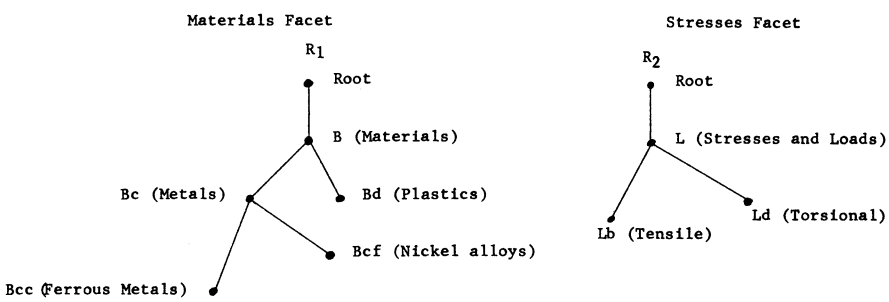

FIG. 2.—Forest of trees, representing a faceted notation. An index term such as *BccLb* represents documents about tensile stress in ferrous metals; it is built up from a node from the materials facet and one from the stresses facet. A document about ferrous metals, but saying nothing about stresses, is represented by *Bcc*, or *Bcc* R_2 if all facets are made explicit. Adapted from [8].

Because of the many attractive properties of a tree structure with regard to implementation, there is considerable pressure to transform a faceted scheme into a single tree. Defining a "natural" order of the facets in effect accomplishes this goal, and the string of path descriptions becomes a single path description of the enlarged tree. A subject heading language which allows form subdivisions and subheadings can be included in this class of languages. We consider that the distinction, sometimes made in discussions of hierarchical and faceted schemes, of precoordinated as opposed to postcoordinated languages becomes blurred when an order of facets is imposed on a faceted

scheme. With such an order, a faceted scheme can be formally represented as a single tree, with all the rigidity that this implies. Once an order is imposed, the main difference between a faceted scheme and its tree representation is one of economy. The explicit factoring of a tree into its facets, should such an operation be possible, is the most economical way to represent the tree. This consideration is recognized in the Dewey Decimal classification in terms of the various subdivision tables included with the schedules.

At the opposite extreme from the hierarchical languages are the uniterm type languages. Here, so far as its originators are concerned, the structure consists of a large set of trees, one for each key word. Each tree has but two nodes, the root indicating that the term is not applied, and the leaf indicating that it is applied to a document. The index record may be made up of a list of natural language terms or else be represented by a binary vector; the latter is in fact a set of path descriptions, one path for each tree, with the trees ordered (see fig. 3).

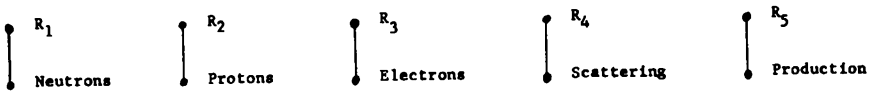

FIG. 3.—Forest representation of a uniterm type of system. A document about electron scattering may be represented by the terms (electron, scattering), or explicitly as (R_1, R_2, electrons, scattering, R_5). The binary vector (0, 0, 1, 1, 0) is a commonly used compact form of this representation.

A distinction may be made between indexing a document and indexing a subject which may possibly be one of a number dealt with in a document. The distinction produces no difficulties for a classification scheme, as each subject is represented by a single term. However, in a uniterm type of scheme, when a subject may be described by several descriptions, confusion may result. Uniterm systems can be generalized by means of devices by which several descriptors describing a single subject are consolidated into what is effectively a single complex index term. These devices are referred to as "links." An example is the concatenation of notations from different facets of a faceted scheme into a single term.

Thus we see that so far as extrinsic structures are concerned, a wide variety of languages can be described in a unified manner as a forest of trees. For some languages this forest consists of a single luxuriant tree; in others, of a large number of modestly adorned trees consisting only of a root and a single leaf. Further we see that there is an intimate relation between the marks associated with a document and the extrinsic structure of the language. Imposing a structure on a language has a number of advantages. For example, it guides the indexer in assigning terms to documents; it helps the patron, or the system acting in his behalf, choose terms with which to probe this collec-

tion and to plan a search strategy; it helps clarify the meaning intended for a term.

The intrinsic structure of a language has received much attention as part of experiments on automatic indexing and searching. Graphic representations of the intrinsic structure have been called "semantic roadmaps" in the literature [10]. This structure can be used directly in guiding a search, or indirectly, as in the construction of clusters of similar documents [11, 12], or clusters of similar terms as in a thesaurus [13].

3. Semantics: rules for how marks are to be applied. The semantics of a term so far as a system is concerned is the potential set of documents to which it is applied. It is what the term "means" to a system, described by how it applies the term to documents coming into the collection. Here the evocative picture of a many-dimensional concept space, in which each point represents a precisely defined idea, is useful. In this space index terms and documents are represented by sets of points. When indexing a document, one tries to choose index terms collectively representing regions of concept space that cover as much as possible of the set representing the document, while covering as few points as possible outside that set. The structure of a language guides the user from one term to related terms, leading him to the most appropriate ones, and also controls for the fact that different individuals will consider a term as covering somewhat different regions.

From the point of view of our model, it is important to recognize that the choice of index terms determines the retrieval capability of the system and that ultimately the rule governing whether or not an index term ought to be applied to a document should follow from considerations as to the consequences of that act for the future operation of the system.

These rules assume a wide variety of forms. They are informally contained in the scope notes of classification schedules and authority lists. Attempts to make these rules precise, as must be the case for automated systems, tend to depend on word counts. An early and simple example involves assigning to a document all words not in a stop list that occur most frequently in the document [14]; here the set of permissible index terms is the words of a natural language that are not on the stop list. It is also possible to collect groups of natural language words into concept classes, and the index terms assigned to documents will represent these classes rather than individual words [15]. More elaborate procedures involve comparing how often a word occurs in text with how often it is expected to occur on the basis of the total number of occurrences of the word in a collection [16]. These well-known devices can all be seen as ways of defining the "semantics" of the system.

Under some methods words selected from a natural language are designated as "content bearing" or "informing" words because of their syntactic or other properties, and if such a word occurs at least once in a document, that word is used as an index term for that document [17]. The set of content bearing words, which may be derived automatically, makes up the indexing

language. Recently attempts have been made to relate explicitly the indexing decision to the expected consequences of the decision [5, 18]. This approach provides a bridge between the other two approaches. The set of possible index terms is all the words in a natural language, but a criterion can be deduced that allows one to assess the likelihood of any term ever being used, in effect permitting a list of content bearing, or informing, words to be defined.

Search Prescriptions

The user seeking information transforms his "information need" into some linguistic expression or request. The request may be further transformed into a search prescription, an object that the system can manipulate. In the case of SDI systems, the search prescription is commonly called an "interest profile." The end result may be a record parallel to that produced when documents are transformed into index records, in which case the considerations voiced there apply here as well.

Retrieval Status Values

The retrieval status values define the extent to which the system is capable of distinguishing documents with regard to its expectation of their being relevant to a request. They may be realized in a number of different ways. The simplest instance is when the system can only distinguish between whether or not a document is relevant. The conventional library is an example. A request is here defined in terms of a subject heading, or a classification number. The listing of a book under the subject heading, or its placement on the shelves according to the class number, corresponds to assigning the status value "retrieved"; all other books are implicitly assigned the value "not retrieved." In other systems the status value may be defined by whether or not a document is identified on a list given to a patron. Thus the mechanisms provided for retrieval by conventional libraries can be discussed by terms of the formalism introduced in this paper.

More complex systems will allow distinctions to be made between predicted degrees of relevance. These distinctions may be explicit, as when a number indicating the degree of relevance is printed alongside the bibliographic description of a document; or it may be implied, as when the system provides a list of documents printed in order of assessed relevance.

We see that retrieval status values may be realized in a variety of forms, but regardless of implementation all share one basic characteristic: in one way or another they provide a simply ordered set of categories for distinguishing documents. The assignment of retrieval status values to documents assists the patron by suggesting an idealized search order, and also by providing clues as to when the search should terminate.

Retrieval Function

The retrieval function represents the mechanism by which, given a request, the system decides how to assign retrieval status values to documents. As a wide variety of alternatives are possible, a few examples are in order. For each case a simplified model of an information retrieval system, that allows the principles behind the retrieval function to be most clearly enunciated, is presented.

1. *Subject catalog model.*—The simplest and most widely used reference retrieval system is the subject catalog of a conventional library; the index terms, often called "subject headings," are taken from a set, A, usually referred to in this context as an authority list; possible structural complexities of these terms are ignored in this simplified model. Several terms may be assigned to a document. The patron in turn chooses a *single* term from the authority list to represent his need. The retrieval status values are "retrieved" or "not retrieved," here denoted by "1" or "0" respectively. The retrieval function, T, assigns the value 1 to those documents whose index records are filed behind the index term, or subject heading, representing the request. Other implementations of this system take such forms as subject book catalogs, and KWIC and KWOC indexes.

More formally, the system can be defined as follows, where 2^A will denote the set of all possible subsets of the set A:

$$R = A, \tag{1A}$$
$$I = 2^A, \tag{1B}$$
$$V = \{1, 0\}, \tag{1C}$$
$$\text{For } r \epsilon R,\ i \epsilon I\text{: } T_r(i) = \begin{cases} 1 \text{ if } r \epsilon i \\ 0 \text{ otherwise} \end{cases} \tag{1D}$$

2. *Inverted file systems.*—These systems, sometimes implemented as "optical coincidence" or "peek-a-boo" systems, are similar to the subject catalog systems described above, except that now the user may form a request made up of several terms. A document is retrieved if its index record contains all the terms making up the request. This system can be formalized as follows:

$$R = 2^A, \tag{2A}$$
$$I = 2^A, \tag{2B}$$
$$V = \{1, 0\}, \tag{2C}$$
$$T_r(i) = \begin{cases} 1 \text{ if } r \subseteq i \\ 0 \text{ otherwise} \end{cases} \tag{2D}$$

An overlap system can be defined by allowing a document to be retrieved even though the request is not contained in the index record, provided that the overlap of terms satisfies certain conditions. For example, the size of the overlap may have to exceed some threshold, t, in size:

$$T_r(i) = \begin{cases} 1 \text{ if } \|r \cap i\| \geq t \\ 0 \text{ otherwise} \end{cases}, \tag{2D'}$$

though some normalized form taking into account the magnitudes of $\|r\|$ and $\|i\|$ would be preferable. Here $\|r\|$ denotes the number of elements in the set r.

3. *Ranked output systems.*—A higher level of complexity results if documents are ranked according to their assessed relevance to requests. It is convenient here to represent the documents and requests as binary vectors taken from a space whose dimension is equal to the number of terms in the index vocabulary. Let B denote this space, and $\|r\|$ denote the size of a vector r in this space. A formal description of one such system would be:

$$R = B, \tag{3A}$$
$$I = B, \tag{3B}$$
$$V = [0, \infty], \tag{3C}$$
$$T_r(i) = \frac{(r, i)}{\|r\| \cdot \|i\|}. \tag{3D}$$

Here (r, i) denotes the usual inner product between the vectors r and i. Further elaboration would permit weights to be assigned to terms, thereby indicating varying degrees of relevance of a term to a document. Aside from changing B to the general vector space of n dimensions, the model remains as above. Such systems are sometimes called "weighted indexing systems."

4. *Boolean systems.*—In this final example we shall consider requests in the form of Boolean expressions. Since much confusion exists in the literature with regard to this topic, we must make precise what *we* mean by such a system.

First let us define the set E of legitimate Boolean expressions. Let A denote the set of legitimate index terms. Then

if $a \epsilon A$, then $a \epsilon E$, that is, an index term, by itself, is a legitimate expression; (4A)
if $s \epsilon E$, then (not.s)ϵE; (4B)
if s, $t \epsilon E$, then (s.and.t)ϵE; (4C)
if s, $t \epsilon E$, then (s.or.t)ϵE. (4D)
Nothing else belongs to E. (4E)

A member of E will be called a "statement." A statement, s, is true with regard to a document represented by the index record i, expressed by $f(s,i)$ = TRUE, under the following conditions (an index record i is considered to be a subset of A): If

$s \epsilon i$, that is, if the request is simply an index term in i; (4A')

s = (not.t) and t is not true; (4B′)
s = (x.and.y) and x and y are *both* true; (4C′)
s = (x.or.y) and x or y or both are true. (4D′)
Otherwise s is false, written $f(s, i)$ = FALSE. (4E′)

Such a system can now be formalized as follows:

$$I = 2^A, \tag{4A″}$$
$$R = E, \tag{4B″}$$
$$V = \{1, 0\}, \tag{4C″}$$
$$T_r(i) = \begin{cases} 1 \text{ if } f(r, i) = \text{TRUE} \\ 0 \text{ if } f(r, i) = \text{FALSE} \end{cases}. \tag{4D″}$$

It is emphasized that the Boolean operators, for example "not.", are applied to *requests* to convert them into other requests; they are not applied to index terms, and the assertion " 'not.a' is an index term describing a document" is without meaning.

Boolean systems are sometimes combined with weighted index systems, a perilous undertaking so far as the logic of the system is concerned.

Conclusions

In this paper we have defined a model intended to describe a wide variety of information systems. While the model is an extremely simple one, it does provide a perspective that unifies a wide variety of systems, and allows us rigorously to prove interesting properties of these systems. Furthermore, much of the theory of information retrieval can be understood as an elaboration of this model.

Our model and the ensuing discussion emphasize the logic of information retrieval systems. Physical realizations of this logic may take a wide variety of forms. Card and book catalogs and subject bibliographies are well known examples. The organization of books on a shelf, made possible by the tree structure of classification schemes, is another; here a request is represented by the user's going to the appropriate place in the stacks, and the books located in that region are "retrieved." Thus at least some aspects of browsing are included in this model. On the other hand, some aspects of modern retrieval systems, such as feedback mechanisms, are not explicitly included in this model. The user is assumed to enter with a request; the model does not consider whether the request represents an initial attempt on the part of the user to get information, or is the result of previous experience with the system and, perhaps, is formulated with the assistance of the system as well.

A number of other models for information systems have been proposed. One such model, presented by Salton [15, p. 211], is essentially a restricted

version of the model proposed here. Another has been devised by Swets [6]. From the point of view of this paper, the Swets model can be thought of as extending models such as ours to include patron evaluations of the documents. Specifically, documents are thought of as being partitioned by users into those relevant and those not relevant to their requests. Swets then considers the distribution of numeric status values assigned by the retrieval function to documents in each of the two classes. Swets uses this model to provide a criterion for retrieving documents, in effect creating a higher order information system. While Swets assumes that V is the real line and that the distribution of V over the two classes are both normal, other models can and have been examined [7]. We thus see that the approach we take here is consistent with and complements earlier efforts at modeling information systems.

APPENDIX
SUMMARY OF NOTATION

A set is a collection of objects, sometimes called its elements. If c is an element of the set C, we will write $c \epsilon C$. If C_1 and C_2 are two sets, and every element of C_1 is an element of C_2, we will write $C_1 \subseteq C_2$. For any two sets, A and B, we shall define a new set, the "cartesian product" of A and B, as the set of all ordered pairs of the form (a,b), with $a \epsilon A$ and $b \epsilon B$; this set is denoted by $A \times B$.

A function, f, is a rule by which, to every element of a set called the domain of the function, is associated a single point in a set called the range of that function. If D is the domain and R the range, we can write $f{:}d \rightarrow R$, for $d \epsilon D$.

A relation on a set, D, is a means of defining a structure on the set. It is formally defined as subset of $D \times D$, where (a,b) is a member of the relation if a is related to b in the desired manner. A "simple ordering relation" is any relation that is reflexive, transitive, connected, and antisymmetric; in defining these terms we shall write $a \leq b$ if (a, b) belongs to the relation.

Reflexivity: the relation is reflexive if and only if $a \leq a$ for all $a \epsilon D$.

Transitivity: the relation is transitive if and only if $a \leq b$ and $b \leq c$ implies $a \leq c$, for all $a, b, c \epsilon D$.

Connectedness: the relation is connected if and only if for any $a, b \epsilon D$, if it is not true that $a \leq b$ (written $a \nleq b$), then it must be true that $b \leq a$.

Antisymmetry: the relation is antisymmetric if, and only if, for all $a, b \epsilon D$, $a \leq b$ and $b \leq a$ imply $a = b$.

A "weak ordering relation" is a relation that is reflexive, transitive, and connected, but not necessarily antisymmetric.

REFERENCES

1. Meadow, Charles T. *The Analysis of Information Systems: A Programmer's Introduction to Information Retrieval.* New York: John Wiley & Sons, 1967.
2. Maron, M. E., and Kuhns, J. L. "On Relevance, Probabilistic-Indexing and Information Retrieval." *Journal of the Association for Computing Machinery* 7 (1960): 216–44.

3. Cooper, William S. "Expected Search Length: A Single Measure of Retrieval Effectiveness Based on the Weak Ordering Action of Retrieval Systems." *American Documentation* 19 (January 1968): 30–41.
4. Suppes, Patrick. *Introduction to Logic.* Princeton, N.J.: Van Nostrand Co., 1957.
5. Bookstein, Abraham, and Swanson, Don R. "A Decision Theoretic Foundation for Indexing." *Journal of the American Society for Information Science* 26 (January 1975): 45–50.
6. Bookstein, Abraham. "The Anomalous Behavior of Precision in the Swets Model, and Its Resolution." *Journal of Documentation* 30 (December 1974): 374–80.
7. Bookstein, Abraham. "When the Most 'Pertinent' Document Should Not Be Retrieved—An Analysis of the Swets Model." (In preparation).
8. Lancaster, F. W. *Information Retrieval Systems: Characteristics, Testing, and Evaluation.* New York: John Wiley & Sons, 1968.
9. Knuth, Donald E. *The Art of Computer Programming.* Vol. 1. Reading, Mass.: Addison-Wesley Publishing Co., 1968.
10. Doyle, Lauren B. "Semantic Road Maps for Literature Searchers." *Journal of the Association for Computing Machinery* 8 (October 1961): 553–78.
11. Van Rijsbergen, C. J. "Further Experiments with Hierarchic Clustering in Document Retrieval." *Information Storage and Retrieval* 10 (January 1974): 1–14.
12. Salton, Gerard. "Cluster Search Strategies and the Optimization of Retrieval Effectiveness." In *The SMART Retrieval System: Experiments in Automatic Document Processing.* Edited by Gerard Salton. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1971.
13. Sparck Jones, Karen. *Automatic Keyword Classification for Information Retrieval.* Hamden, Conn.: Archon Books, 1971.
14. Luhn, H. P. "A Statistical Approach to Mechanized Encoding and Searching of Literary Information." *IBM Journal of Research and Development* 1 (1957): 309–17.
15. Salton, Gerard. *Automatic Information Organization and Retrieval.* New York: McGraw-Hill Book Co., 1968.
16. Damerau, Fred J. "An Experiment in Automatic Indexing." *American Documentation* 16 (October 1965): 283–89.
17. Maron, M. E. "Automatic Indexing, an Experimental Inquiry." *Journal of the Association for Computing Machinery* 8 (July 1961): 404–17.
18. Bookstein, Abraham, and Swanson, Don R. "Probabilistic Models for Automatic Indexing." *Journal of the American Society for Information Science* 25 (September 1974): 312–18.