Information Processing & Management Vol. 27, No. 1, pp. 27–41, 1991
Printed in Great Britain.

0306-4573/91 $3.00 + .00

RANKING LARGE DOCUMENT COLLECTIONS BY A STATE SPACE SEARCH

MICHAEL D. GORDON
Computer and Information Systems, School of Business,
University of Michigan, Ann Arbor, MI 48109-1234, U.S.A.

(*Received* 9 *August* 1989; *accepted in final form* 20 *June* 1990)

Abstract—An algorithm is described for ordering by probability of relevance overlapping document subsets from which a searcher should choose the next document. The algorithm produces the ordering without assumptions of index term independence, improves its performance with increasing feedback, and estimates most accurately the probabilities of relevance of the subsets most likely to be relevant. The efficiency and effectiveness of the algorithm are analyzed theoretically.

1. INTRODUCTION

A collection of documents described by index terms defines a search space of document subsets. For efficient and effective retrieval on a large document database, a searcher must make decisions about a vast number of such subsets. These decisions, concerning which subset to retrieve a document from next, implicitly involve probability estimations and combinatorics. Unfortunately, people deal poorly with such situations. Kahneman, Slovic, and Tversky (1982) describe pervasive human fallacies in estimating probabilities, even among mathematically sophisticated individuals. Blair (1980) argues that unaided searchers cannot effectively construct queries that explore a large search space of overlapping document subsets.

This paper presents a theoretical rationale for employing a novel form of information retrieval algorithm. The algorithm helps searchers better navigate through large document collections by continually revising probability estimates for document subsets. The algorithm avoids assumptions of index term independence and predicts relevance more accurately with increasing feedback. A mathematical analysis of the algorithm shows it can efficiently provide information about the probability of relevance of many different subsets of documents.

Consider a set of documents relevant to an inquirer's need for information. Suppose the collection containing this set is very large (millions of documents or more), and the relevant set itself contains thousands of documents. Further, suppose the inquirer requires many of these documents to do his or her job effectively. (Legal, scholarly, or business research are examples of situations in which these assumptions apply.) Of course, the entire collection is far too large for the inquirer to search, and he or she needs the collection arranged in an order that reduces the effort required to find the number of relevant documents needed.

Finally, suppose that the inquirer can identify perhaps a dozen or fifteen "good" search terms* but is unable to specify in what combinations they will be best used to retrieve relevant documents. Call this number of terms m.

$2^m - 1$ disjoint, non-empty subsets of these good search terms can thus be defined. A particular subset of terms can be used to specify for retrieval just those documents that are indexed by all these terms but by none of the other m terms. For instance, with $m =$

*Following the model for probabilistic retrieval (van Rijsbergen, 1979), a "good" search term might be one for which $p_i > q_i$, that is, one for which the probability of the term being used in a relevant document exceeds its being used in a nonrelevant document. Such terms contribute positively, for documents which employ them, to a discriminant function whose value increases as its prediction of the relevance of a document increases.

This research was supported by the Graduate School of Business Administration, Division of Research, University of Michigan.

3 "good" terms "computers," "business," and "automation," the subset {business, automation} would retrieve any document indexed by both "business" and "automation," but would not retrieve documents indexed by any of the other $2^m - 2$ other combinations of these three terms. (A document indexed by the terms {business, automation, telecommunications} would be retrieved, because the term "telecommunications" lies outside of the set of m term being considered.**)

The goal of an information retrieval system in this circumstance is to present these $2^m - 1$ subsets in an order that minimizes the effort that an inquirer looking through the documents in the presented order will have to put forth in order to find the number of relevant documents he or she desires.

The "probability ranking principle" (Robertson, 1977) implies that, under certain conditions, the optimal presentation order is to present the subset with the highest (estimated) probability first, the next highest second, and so on. Probabilistic retrieval (van Rijsbergen, 1979) is the most common method mentioned for producing such rankings. It uses estimates of the probabilities that various index terms appear in both relevant and nonrelevant documents, and these probabilities are then combined to provide an estimate of the probability of relevance for documents within a subset indexed by a given set of index terms.

The chief failing of this method is its usual reliance on an assumption that index terms are independently distributed (conditioned on either an assumption of document relevance or document nonrelevance). However, the underlying independence assumption is not valid (van Rijsbergen, 1977; Tague 1984), meaning that the rankings of various subsets of documents will be suboptimal. Unfortunately, no computationally tractable methods for ranking document subsets have been developed that significantly increase the effectiveness of retrieval based on the independence assumption.

Because of the failing of the independence assumption, as the estimates of the probabilities of term occurrences improve, there is no guarantee that the estimate of the probability of relevance of a document employing just those terms will improve. For instance, by basing our estimates on larger samples, we can improve our (separate) estimates of the probability of *occurrence* of term_a, term_b, and term_c in the set of relevant documents as well as in the nonrelevant documents. However, this does not mean that the estimate of probability of *relevance* (calculated using Bayes' rule) for a document containing these three terms will also improve.

In principle, one could obtain samples from all the disjoint subsets described by a set of m terms, and these could be used to estimate the probability of relevance of each of these sets. These sets could then be presented to the user, the sets with the highest estimated probability first. However, with $2^m - 1$ distinct combinations, we could not possibly expect to sample each of them reliably. (By "sample" we mean "select documents from and evaluate their relevance.")

The algorithm we propose in this paper for rank ordering sets of documents attempts to defeat the three objections just raised. That is, the proposed algorithm

- avoids assumptions of term independence;
- improves its estimates of $P(\text{Rel}|)$ with increasing sample size; and
- permits one to sample the document state space far more rapidly than by directly sampling each disjoint subset.

Further, the algorithm produces the most accurate estimates of $P(\text{Rel}|)$ for those subsets most likely to be relevant.

2. STATE SPACE SEARCHING

The algorithm we propose for searching a document space is motivated by the A* algorithm used in artificial intelligence for graph searching (see, for example, Nilsson (1980) for a discussion of this algorithm).

We are describing subsets that are in complete conjunctive normal form. Specified terms are ANDed together with the negation of excluded terms. We refer to these as **disjoint subsets or **complete cnf subsets.**

We will see later (section 4) how our method develops a *document retrieval search tree*.† To understand our algorithm better, we first present the A* algorithm for searching a state space in the form of a tree. This algorithm (adapted from Nilsson, 1980) seeks *goal* nodes accessible from the tree's *start* node (i.e., its root).

Algorithm 1: A*

1. Put the root node, *s*, on the list *Open*.
2. Being with an empty list, *Closed*.
3. Loop: If *Open* is empty, exit with failure.
4. Set $n :=$ the first node on *Open*. Take n from *Open* and put on *Closed*.
5. If $n =$ a goal node, exit with success.
6. Generate all immediate descendants of node n. Put them on *Open*.
7. Reorder *Open* so that the "best" node becomes the first node on the list.
8. Goto Loop.

At any stage of the A* algorithm (as described above) there is a (possibly empty) set of *Open* nodes, and their descendants do not yet appear on the tree. The most promising of the *Open* nodes is replaced by each of its immediate descendants until a goal node is reached or until it is impossible to expand the tree further.

For the problem we are interested in, we consider a vocabulary, V, of five "good" search terms, {A, B, C, D, E}. The small size of this set permits clear examples to be presented. In general, we imagine $\|V\|$ to be 12, 15, or more.

In searching a tree of document nodes, we envision a start state, *s*, leading through various intermediate states to various goal states (see Fig. 1). The intermediate states each pertain to a *depth* of sampling performed on a particular subset of documents. For example, the node $ABCDE_3$ represents the third time a document from the set of documents jointly indexed by these terms has been retrieved and evaluated by a searcher.‡ Similarly, node ABE_5 represents the fifth time a document indexed by the three terms A, B, and, E and *possibly but not necessarily* indexed by C and or D, has been retrieved and evaluated. The start state of the graph represents the state of no documents having been retrieved and furnished to a user. A *goal state* (not shown in the figure) represents a search state associated with a subset of documents from which an inquirer has retrieved a relevant document *G* times.

To make the explanation of our algorithm more precise, we introduce some terminology.

3. TERMINOLOGY

Search states

The nodes in the search tree resemble ABE_5 or BD_3, etc. We refer to these representations as *search states* or *search nodes*.

Lattice nodes

Each search state, stripped of its subscript, represents a subset of documents. For instance ABE represents that subset of documents indexed with at least terms A, B, and E (and possibly C and D). Given our vocabulary, V, the set of subsets we can represent forms a lattice, $L = \langle V, \subseteq \rangle$, see Fig. 2.†† Search states indicate a depth of sampling from an as-

†Our algorithm could be presented without reference to a state space. However, we feel that such a representation is familiar, easily grasped, and aids discussion.

‡We use sampling with replacement in selecting documents from various subsets. So doing ensures that our algorithm terminates and simplifies our theoretical analysis. In practice, a document drawn for a second time need not be actually presented to a user. The system need only update relevance statistics based on the user's previous assessment. Note, that sampling with and without replacement both give unbiased estimates of the proportion of documents that is relevant, though the latter method has smaller variance.

††We ignore the lattice "top" which corresponds to documents indexed by none of the given terms. Note that subsets deeper in the lattice are subsets of their relatives higher in the lattice, even though deeper subsets are described by more symbols.

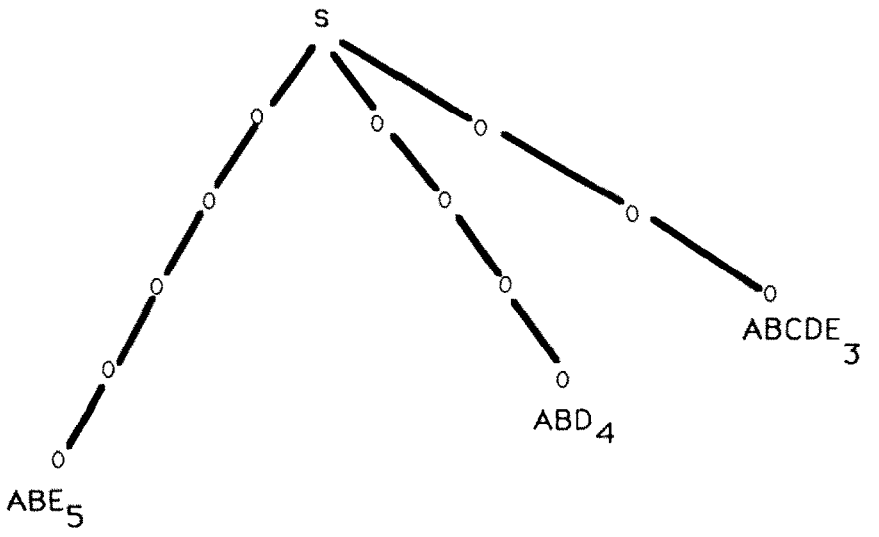

Fig. 1. Document state space.

sociated *lattice node*. For instance, ABE_6 indicates that lattice node ABE has been searched six times.

Intentionally sample

When a document is picked at random subject to the constraint that it be a member of lattice node X (i.e., that its description qualify it as a member of the subset defined by X) and the searcher makes a judgment about the document's relevance, we say that node X has been *intentionally sampled*.

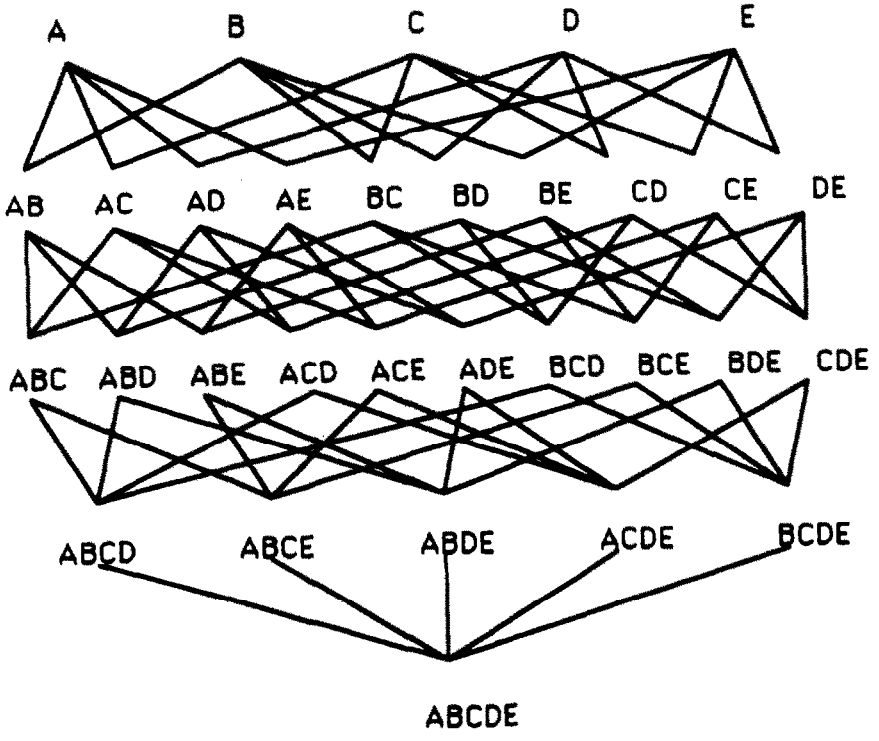

Fig. 2. Lattice of document subsets.

Actually sample

Since the document sets represented by lattice nodes are not disjoint, a document retrieved when one lattice node is intentionally sampled may actually have a description that belongs to several lattice nodes. For instance, in intentionally sampling lattice element ACE, the retrieved document may have description ABCDE. Thus, the retrieved document is an element of each of the following lattice nodes (i.e., subsets) contained in ACE: ACE, ABCE, ACDE, ABCDE. We say that if one intentionally samples lattice node X and retrieves a document, d, then all lattice nodes Y such that $Y \subseteq X$ and $d \in Y$ are actually sampled (see Fig. 3).

THEOREM

Let S_Y *be the documents actually sampled from lattice node* Y *and let* $S_Y \cap \text{Rel}$ *be the documents from* S_Y *that are relevant. Then,* $\#(S_Y \cap \text{Rel})/\#(S_Y)$ *is an unbiased estimate of* $P(\text{Rel} \mid Y)$.

Proof. Choose any $X \supseteq Y$. Let S_X be the set of documents intentionally sampled from X. By the definition of intentional sampling, S_X is a random sample (with replacement) from X. Hence, $S_X \cap Y$ is a random sample from Y. Consequently, $\#(S_X \cap Y \cap \text{Rel})/\#(S_X \cap Y)$ is an unbiased estimator for $P(\text{Rel} \mid Y)$, since it is a sample mean from a random sample. But $S_Y = \cup(S_X \cap Y)$, over all $X \supseteq Y$. Therefore, $\#(S_Y \cap \text{Rel})/\#(S_Y)$ gives an unbiased estimate of $P(\text{Rel} \mid Y)$ by averaging the separate estimates $\#(S_X \cap Y \cap \text{Rel})/\#(S_X \cap Y)$, each of which is unbiased. □

Multiple accrediting

The application of this theorem lies in its ability to help us sample more efficiently the set of lattice nodes. Again, suppose that one intentionally samples node ACE and retrieves a document, d, with description ABCDE. We may then update our estimates of the probability of relevance for the following lattice nodes: ACE, ABCE, ACDE, and ABCDE. For

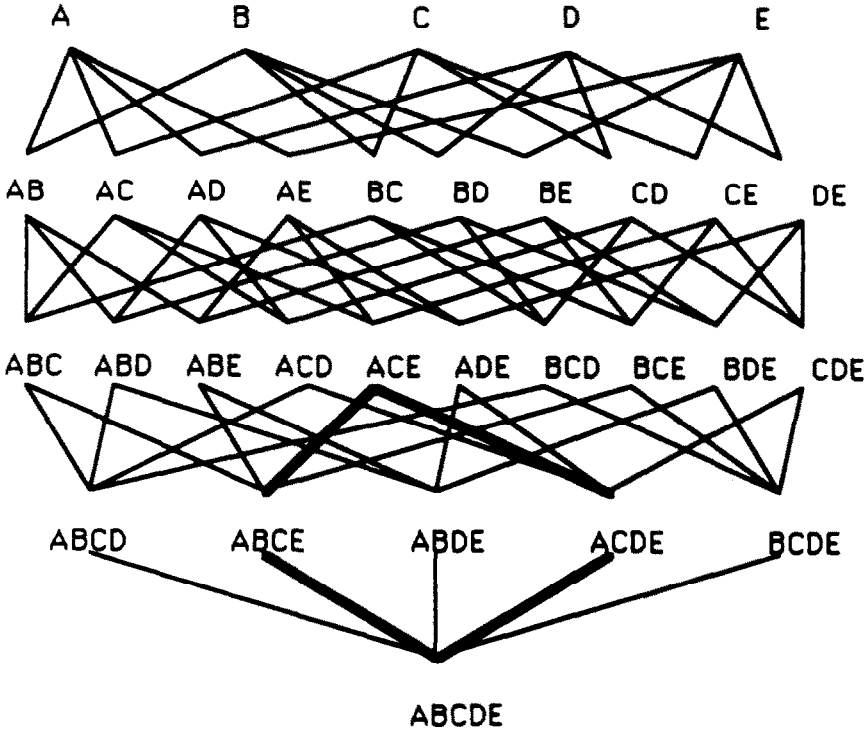

Fig. 3. Actual sampling. The lattices nodes joined by heavy lines are actually sampled when lattice node ACE is intentionally sampled and a document with description ABCDE is retrieved.

instance, if d is relevant, and ABCE had previously been actually sampled four times and found relevant twice, we would update our estimate of P(Rel|ABCE) from 2/4 to 3/5 (and similarly for the other nodes). We refer to the process of simultaneously updating the probabilities of relevance of several actually sampled lattice nodes with the retrieval and evaluation of a single (intentionally sampled) document as *multiple accrediting*.

4. SEARCHING A DOCUMENT STATE SPACE

With this terminology in hand, we describe an algorithm for searching a document retrieval state space. The algorithm seeks multiple *goal nodes*. A goal node is a search node associated with a lattice node that has been actually sampled (with replacement) until a specified number, G, of relevant documents has been retrieved.

Consider the lattice in Fig. 2. We desire that our algorithm rank these document subsets (lattice elements) in order of decreasing probability of relevance to a given query. As a first step, we seek as a goal the lattice node with the highest probability of relevance. (We will then look for other goal nodes, unlike the A* algorithm.) We assume in this example that all lattice nodes with two or fewer symbols have a lower probability of relevance than lattice nodes with three or more symbols. Thus, we *initially sample* only the latter lattice nodes, ignoring all lattice nodes with fewer than three symbols. This assumption limits the number of nodes we initially sample. This assumption may be modified to allow initial sampling of other sets of lattice nodes.

Algorithm 2: Document space search

1. For each lattice node employing three or more symbols, intentionally sample that node j ($j = 5$, e.g.) times. Use multiple accrediting for each document retrieved and evaluated. In this way we obtain our initial sample. (At this stage, as a result of multiple accrediting, we might have the associated search tree depicted in Fig. 4.)
2. Put each of the current leaf nodes on the *Open* list. Begin with an empty list, *Closed*.
3. Loop: If *Open* is empty, exit with failure.
4. Select from *Open* any goal node. If such a goal exits, exit with success.
5. Select from *Open* the node, n, that has the smallest value of $f(n)$ (see discussion that follows). Take n from *Open* and put on *Closed*.
6. Intentionally sample one document from the lattice node associated with node n.
7. Multiply accredit each actually sampled node. For each multiply accredited node, add the corresponding nodes to the search tree. Adjust the tree so that all leaf nodes are attached to their "natural" parents, and remove from *Open* and place on *Closed* any node that now has a "descendant" on the tree. (For instance, if the search tree looks like Fig. 4 before step 7 and a document with ABCDE is selected when node ACE is intentionally sampled, then nodes ACE, ABCE, ACDE, and ABCDE are multiply accredited (see again Fig. 3). The search tree following Step 7 is shown in Fig. 5. Nodes $ABCDE_7$, $ABCE_6$, ACE_8, and $ACDE_6$ are placed on *Open*, and $ABCDE_6$, $ABCE_5$, ACE_7, and $ACDE_5$ are removed from *Open* and placed on *Closed*.)
8. Goto Loop

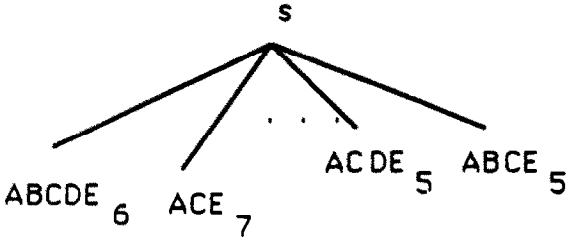

Fig. 4. Search tree from initial sampling.

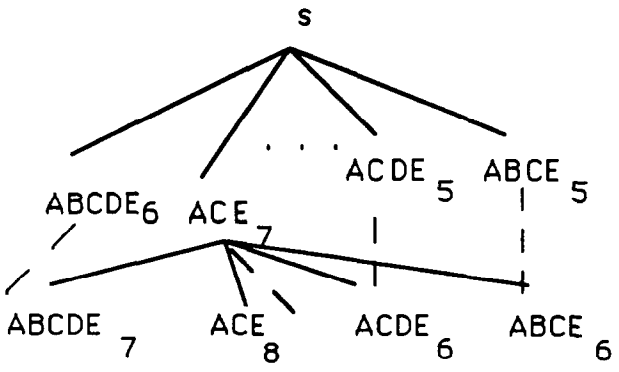

Fig. 5. Multiple accrediting and tree updating. The search tree is adjusted to show a) the lattice nodes actually sampled when ACE is intentionally sampled (solid arcs emanating from ACE_7); and b) the situation after search states are connected to their "natural" parents (dotted arcs).

The algorithm above, like the A* algorithm, selects for expansion (Step 5) the best node, that is the node with the smallest value of $f(\ )$. In the A* algorithm, the "best" search node (Step 7 of Algorithm 1) is that node, n, which minimizes the value of $f(n) = g(n) + h(n)$. The right hand side of this equation is composed of two estimated values. $g(n)$ estimates the least costly path from a start state to node n. In the case of searching a tree, a single path leads from the start state to any search state. Thus, once the cost of this single path is determined (which is done automatically once this node is generated), the estimate of the least costly path to this node will always equal the true cost. The second term on the right-hand side of the equation, $h(n)$, estimates the least costly path from n to any goal node. Thus, $f(n)$ estimates the least costly path from the start state, s, to a goal node by going through n. In our calculation of $f(\ )$, like that for the A* algorithm, we hope to estimate a least cost path from the start state to a goal. However, unlike the A* algorithm, path lengths in our algorithm depend on probabilistic outcomes. We thus choose functions $g(\ )$ and $h(\ )$ accordingly.

We take $g(n)$ to be the path length to node n. That is, node ABC_5 has path length 5, node $BCDE_7$ has path length 7, etc. These path lengths indicate exactly the number of times the corresponding lattice element has been actually sampled—accounting for attaching nodes to their natural parents, as we did in step 8 of our algorithm.

$h(n)$, on the other hand, estimates the distance from node n to a goal. For instance, $h(ABC_5)$ estimates the number of additional times lattice node ABC will need to be actually sampled until it becomes a goal (i.e., until it has been actually sampled with replacement and found to be relevant G times). We take as our estimate of $h(n)$ the expected number of documents yet to be retrieved for this node until the Gth relevant document is discovered.‡‡ Intuitively, selecting for expansion that node, n, with the smallest value of $f(n) = g(n) + h(n)$ is equivalent to selecting the node with estimated shortest (i.e., least cost) path length from the start node, s, through node n to a goal.

LEMMA

For a node, n, with depth b and a relevant documents discovered thus far

$$f(n) = b + (G - a) * (b/a).$$

Proof. By definition, the depth of this node, $g(n)$, is b, and an unbiased estimate of its probability of relevance is $p = a/b$. The expected number of documents yet to be retrieved must account for $G - a$ additional *relevant* retrievals with k *nonrelevant* retrievals preceding the $(G - a)$th relevant document retrieved. That is,

‡‡We estimate the probability of relevance of a node by the sampling statistics (number of times found relevant divided by number of times actually sampled) that describe it. This yields an unbiased estimate of its relevance.

$$h(n) = \mathrm{E}(G - a + k)$$
$$= G - a + \mathrm{E}(k) \ .$$

Since k follows the negative binomial distribution, we have

$$h(n) = G - a + (G - a) * (q/p)$$

where $p = 1 - q$ = estimated probability that node n is relevant. Thus

$$f(n) = g(n) + h(n)$$
$$= b + (G - a) + (G - a) * (q/p)$$
$$= b + (G - a) + (G - a) * [(b - a)/a]$$
$$= b + (G - a) * (b/a).$$ □

5. ANALYSIS

We now analyze the algorithm we have presented. We examine both its effectiveness and its efficiency.

THEOREM

Algorithm 2 *always terminates by finding a goal, given that every lattice node contains at least one relevant document.*

Proof. By steps 1 and 2 of the algorithm, *Open* begins as a non-empty list. *Open* can only become empty (and the algorithm terminate in failure, step 3), if nongoal nodes are removed from *Open* without being replaced. But steps 5 and 7 show that, for any nongoal node removed, at least one new node is placed on *Open*. Thus, the algorithm never terminates in failure.

Further, since the lattice is finite, there must be a finite number of search nodes on *Open*. Thus, if the algorithm does not terminate, at least one lattice node is sampled an infinite number of times without leading to a goal node. That is, a relevant document is never selected from this (lattice) node for the Gth time. But, we are sampling with replacement, and each lattice node has one or more relevant documents by assumption. Thus, the number of draws, k, until the Gth relevant document is discovered for this node is defined by the negative binomial distribution. But the probability that k is finite is 1.0 (Feller, 1968). Thus, no node will be sampled an infinite number of times before a relevant document is discovered for the Gth time. Therefore, the algorithm always terminates.

Since the algorithm always terminates and never terminates in failure, it always terminates by finding a goal. □

The proof above relies on the assumption that every lattice node contains at least one relevant document and the fact that we are sampling with replacement. Since a lattice node describes a set of documents, each of which is indexed with a given set of good index terms, the assumption is quite reasonable. Sampling with replacement ensures that we can retrieve an arbitrary number of relevant documents from any lattice node.

THEOREM

The Open node *with the highest estimated value for* $P(\mathrm{Rel}\,|\,\mathrm{node})$ *will be intentionally sampled next by the algorithm.*

Proof. Let *Open* nodes n and n' have depths of b and b', respectively, and have a and a' relevant documents thus far retrieved, respectively. It suffices to show that f(node) decreases monotonically with increasing estimated values for $P(\mathrm{Rel}\,|\,\mathrm{node})$:

$$f(n) < f(n') \text{ iff}$$

$$b + (G - a) * b/a < b' + (G - a') * b'/a' \text{ iff}$$

$$baa' + Ga'b - aa'b < b'aa' + Gb'a - a'b'a \text{ iff}$$

$$a'b * (a + G - a) < b'a * (a' + G - a') \text{ iff}$$

$$a'b < b'a \text{ iff}$$

$$p_{n'} = a'/b' < a/b = p_n,$$

where p_n and $p_{n'}$ are the estimated values for $P(\text{Rel}|n)$ and $P(\text{Rel}|n')$, respectively. □

This means that to pick the search state most likely to become a goal, we need only look at its estimated probability of relevance and can ignore its path length.

It can be proven that, if $h(n)$ never overestimates the true value of the least costly path from n to a goal node, the A* algorithm always finds the least costly path from the start state to a goal. Next, we develop an analogous proof for Algorithm 2.

THEOREM

The lattice node with the highest (non-estimated) probability of relevance will most often be the first one to place a goal node on Open.

Proof. Because each lattice node contains at least one relevant document, it is, potentially, actually sampled with replacement until it becomes a goal node. The negative binomial distribution

$$f(k; G, p) = \binom{G + k - 1}{k} * p^G q^k$$

defines, for any lattice node, m, with $P(\text{Rel}|m) = \text{p} = 1 - q$, the probability that the Gth relevant document retrieved from that node will be preceded by the retrieval of exactly k nonrelevant documents.

Let the probability of relevance for lattice nodes L_1 and L_2 be p_1 and p_2, respectively, and let k_i be the number of failures until lattice node L_i becomes a goal. We expect a goal node for L_2 to reach *Open* with fewer failures than L_1 if

$$P(k_1 > k_2) > P(k_2 > k_1), \text{ or}$$

$$P(k_1 > k_2) - P(k_2 > k_1) > 0.$$

If the number of failures associated with sampling L_1 and L_2 are independent then

$$P(k_1 > k_2) - P(k_2 > k_1) > 0 \quad \text{iff}$$

$$\sum_{j=0}^{\infty} \sum_{i=j+1}^{\infty} [P(k_1 = i) * P(k_2 = j)] - \sum_{j=0}^{\infty} \sum_{i=j+1}^{\infty} [P(k_2 = i) * P(k_1 = j)] > 0 \quad \text{iff}$$

$$\sum_{j=0}^{\infty} \sum_{i=j+1}^{\infty} \{[P(k_1 = i) * P(k_2 = j)] - [P(k_2 = i) * P(k_1 = j)]\} > 0.$$

Using the formula $f(k; G, p)$, this becomes

$$\sum_{j=0}^{\infty} \sum_{i=j+1}^{\infty} \left[\binom{G + i - 1}{i} * p_1^G q_1^i * \binom{G + j - 1}{j} * p_2^G q_2^j - \binom{G + i - 1}{i} * p_2^G q_2^i * \binom{G + j - 1}{j} * p_1^G q_1^j \right] > 0.$$

Factoring and distributing terms converts the inequality to

$$(p_1 * p_2)^G \sum_{j=0}^{\infty} \sum_{i=j+1}^{\infty} \left[\binom{G+i-1}{i} * \binom{G+j-1}{j} * [q_1^i q_2^j - q_2^i q_1^j] \right] > 0,$$

which reduces to

$$q_i^{i-j} > q_2^{i-j} \qquad \text{iff}$$

$$q_i > q_2 \text{ (since } i > j) \qquad \text{iff}$$

$$p_1 < p_2.$$

The analysis when the number of failures from L_1 and L_2 are statistically dependent reduces to case when they are independent. To see this, consider a statistical sampling process that operates as follows when $L_1 \supseteq L_2$. Intentionally sample from L_1. If the document drawn actually samples L_2 (thus contributing another "success" or "failure" to both nodes), then ignore that draw. Otherwise, 1) update L_1 based on this draw; and 2) intentionally sample and update L_2.

Ignoring draws that would multiply accredit both L_1 and L_2 is equivalent to picking a new goal number of relevant documents, G'-G = number of "successes" from common draws, and then considering statistically independent draws from L_1 and L_2 to reach this new criterion. Therefore, since $P(k_1 > k_2) > P(k_2 > k_1)$ in the independence case for any G, it holds in the case where $L_1 \supseteq L_2$, too. Thus, as in the independent case, the probability that $k_1 < k_2$ depends only on p_1 and p_2. The case where neither $L_1 \supseteq L_2$ nor $L_2 \supseteq L_1$ also reduces to the case of statistical independence by using a sampling procedure similar to that just described which incorporates the probabilities of documents belonging to common supersets of both L_1 and L_2.

Thus, one lattice node is more likely to place a search node on *Open* with fewer failures than another if it has a higher probability of relevance. Together with the fact that the node with the highest estimated probability of relevance is the node that will be sampled by the algorithm, we conclude that the lattice node with highest probability of relevance will most often be the first node to place a goal node on *Open*. □

To this point, we have seen that selecting for expansion the node with the smallest value of $f(\ )$ is equivalent to selecting, on average, the node with the highest probability of relevance, and, by continuing to pick in this way, that node will be the first to have an associated Goal node placed on *Open*. By step 4, the algorithm will terminate with success upon locating this node.

Continuation

After locating a goal node and "terminating" with success, the algorithm resumes by trying to locate other goal nodes. Specifically, the search tree is kept intact, along with all statistics on the number of times each lattice node has been actually sampled and found relevant. The algorithm then continues by considering for expansion only those search nodes that have not yet become goal nodes. The best of these is intentionally sampled with multiple accrediting. Eventually, a second goal node will be discovered. The entire procedure is then continued as governed by a stopping rule: repeat until a predetermined number of goal nodes has been discovered; or until the total number of documents in the union of the set of goal nodes reaches some threshold. Once the stopping rule is reached, the documents in lattice nodes associated with goal nodes are presented to the searcher in order of predicted probability of relevance—each document within a lattice node receiving the same estimated probability. By regarding the search for each new goal node as a new problem, and maintaining all information concerning the depth of (actual) sampling and relevance of each node that has not yet become a goal, each of the theorems above can be applied at each new stage.

Speed up

By multiple accrediting, we can sample the set of lattice nodes far more efficiently than we could otherwise. Here, we examine the "speed up" in sampling that is achieved, subject to certain assumptions. Again, for the lemmas and theorem that follow, we assume a vocabulary of "good" search terms, V, where $\|V\| = n$. For illustrative purposes, we refer to Fig. 2, which uses $\|V\| = 5$.

Note: we use the expression $X \subseteq Y$ to mean that "the set of documents indexed by X is a subset of those indexed by Y." For example, ABCDE $\subseteq$ ABC.

LEMMA 1

A lattice node, m, described by σ symbols has $2^{n-\sigma}$ *descendants.* (X *is a descendant of* Y *iff* $X \subseteq Y$. *For instance, in Fig.* 2, *lattice node* ABC *has descendants*: ABC, ABCD, ABCE, and ABCDE.)

Proof. Let $k = n - \sigma$. Then we may "adjoin" to lattice node m any combination of these k symbols to represent a lattice node that is a subset of m. (For instance, we may "adjoin" "D" to lattice node ABC to create lattice node ABCD.) There are $\sum_{j=0}^{k} \binom{k}{j} = 2^k$ such combinations we can adjoin. □

LEMMA 2

Let X *be a lattice node and* Y *one of its descendants k levels deeper in the lattice.* (*For instance, lattice node* ABCDE *is* 3 *levels deeper in the lattice of Fig.* 2 *than* BC.) *Then, when one intentionally samples* X *and draws a document with description*§ Y, 2^k *lattice nodes are actually sampled.*

Proof. We seek lattice nodes, R, such that $Y \subseteq R \subseteq X$. (That is, the description of R must contain all the symbols in the description of X, and the description of R must contain only symbols in the description of Y. X = ABC, R = ABCE, and Y = ABCDE jointly satisfy these conditions.) Each of these nodes (i.e., subsets of X) will be actually sampled in the situation that we have described.

Let W be the set of symbols "adjoined" to X in forming Y. Since Y is k levels deeper in the lattice, $\|W\| = k$. Then we may adjoin any combination of these symbols to X and obtain a lattice node with the property we demand of R. As in the previous proof, there are 2^k such combination. □

Assumptions. These two lemmas help us prove our "speed up" theorem. This theorem is based on the following two assumptions: 1) Each disjoint (complete cnf) subset formed by the terms in the underlying searching vocabulary, V, has an equal number of documents. For instance, AB~C~DE, ABCD~E, ~AB~CD~E, etc., all have an equal number of documents. 2) Each of these subsets has the same probability of relevance.

SPEED UP THEOREM

On average, multiple accrediting actually samples

$$\frac{(5/2)^n - (3/2)^n}{2^n - 1}$$

lattice nodes for each node intentionally sampled, where $\|V\| = n$.

Proof. Consider a node, m, with w "levels" of descendants excluding itself. (For example, ABC has two levels of descendants excluding itself: level 1 descendants = {ABCD, ABCE}; and level 2 descendants = {ABCDE}.) By Lemma 2, a lattice node k levels deeper in the lattice actually samples 2^k nodes when it describes a document that is retrieved when m is intentionally sampled. But a node with w levels of descendants has w symbols that can be adjoined to its description to create descendants. (For instance, ABC, with two levels of descendants, can adjoin the two symbols D or E.) Each adjoined symbol corresponds to another level deeper in the lattice. Thus, there are $\binom{w}{i}$ descendants of m that are i lev-

§That is, Y is the subset of X deepest in the lattice that contains the document.

els deeper in the lattice. Because, by assumption, each complete cnf subset of m has the same number of documents, it follows that each descendant of m has the same probability of being the deepest lattice node that describes a document intentionally sampled from m. By Lemma 1, m has 2^w total descendants. Thus, the average number of nodes multiply accredited when node m is intentionally sampled is:

$$\frac{1}{2^w} * \sum_{i=0}^{w} \binom{w}{i} * 2^i ,$$

which equals

$$\frac{1}{2^w} * (2 + 1)^w = (3/2)^w$$

by the binomial theorem.

Further, there are $\binom{n}{w}$ lattice nodes with w levels of descendants. Because, by assumption, all lattice nodes have equal probability of relevance, each will be intentionally sampled equally often. Thus, the average number of nodes actually sampled when an arbitrary node in the lattice of $2^n - 1$ nodes is intentionally sampled is:

$$\frac{1}{2^n - 1} * \sum_{w=0}^{n-1} \binom{n}{w} * (3/2)^w.$$

Again, the binomial theorem can be used to rewrite this as

$$\frac{(5/2)^n - (3/2)^n}{2^n - 1}.$$ □

We deem this the "speed up" from multiple accrediting. A graph showing speed up plotted against search set size, $\|V\|$, is shown in Fig. 6.

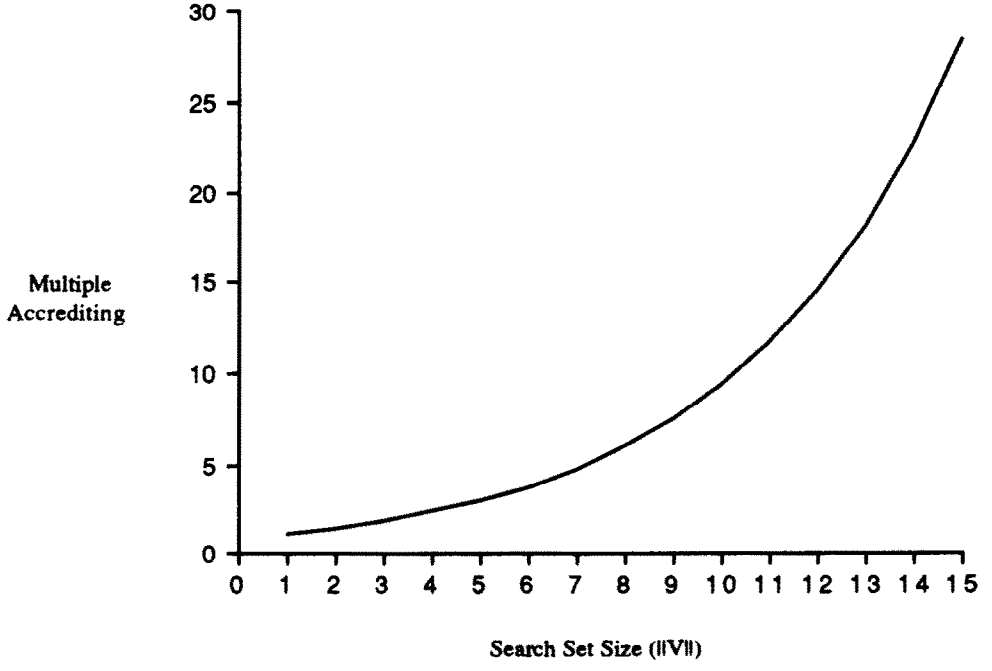

Fig. 6. Speed up via multiple accrediting.

6. DISCUSSION

We stated in the introduction to this paper that our proposed algorithm:

- avoids assumptions of term independence;
- improves its estimates of $P(\text{Rel}|\,)$ with increasing sample size;
- permits one to sample the document state space far more rapidly than by directly sampling each disjoint subset; and
- produces the most accurate estimates of $P(\text{Rel}|\,)$ for those subsets most likely to be relevant.

We now reexamine these assertions.

First, we obtain estimates of form $P(\text{Rel}|\text{X})$ for all lattice nodes, X (excluding those too "high" in the lattice—i.e., those that are not "initially sampled"). These unbiased estimates are obtained by sampling X and do not rely on any assumptions of index term independence.

Second, the estimate for $P(\text{Rel}|\text{X})$ improves with increasing sample size. In particular, we are sampling with replacement and taking $Y =$ the proportion of retrieved documents indexed by X that is relevant as our estimate of $P(\text{Rel}|\text{X})$. Thus, if the true proportion of relevant documents in X is $p = 1 - q$, then the variance of the sampling distribution of Y is $\text{Var}(Y) = pq/n$, where n is the number of actual samples from X. So, our estimates improve with increasing n. Thus, by choosing larger values for G (the number of relevant documents until a node becomes a goal) the ranking of lattice elements improves and we obtain smaller confidence intervals around our estimates of $P(\text{Rel}|\,)$ for the nodes in the lattice.

Third, our speed up theorem indicates that each sampled document serves to sample an exponential number of lattice elements simultaneously. Thus, we obtain large sample sizes for estimating the probability of relevance of each lattice element.

Fourth, the algorithm selects for sampling (steps 5 and 6) the node with the lowest value of $f(\text{node})$ first, which is equivalent to selecting, on average, the node with the highest probability of relevance. Thus, as the algorithm discovers multiple goal nodes, it samples most intensively (and thus produces the most accurate estimates of $P(\text{Rel}|\,)$ for) those nodes most likely to be relevant. It is these nodes we hope to rank most accurately, because a searcher will likely only retrieve documents associated with nodes with the best ranks.

Today, optical storage, subscription information retrieval systems with a single interface to multiple document databases, and other technological advances are making the storage of tens of millions of documents a reality. In such large document databases, a searcher desiring a significant fraction of the documents relevant to his or her need should expect to have to exert considerable effort in providing feedback for his or her search. For our searcher who has identified 12 "good" search terms, there will be over 4,000 lattice nodes by which such a database will be partitioned. Even if we wish to actually sample each of these nodes 15 times to produce a ranking, our speed up theorem suggests that 4000 (total) retrievals and evaluations will suffice. Such an effort may be extremely worthwhile when one needs to find all the documents needed in a lawsuit, or to determine whether to grant a new patent, or to conduct an exhaustive scientific literature review, etc. By limiting the "initially sampled" set of documents to those, say, at least half way down the lattice (for instance, to documents containing at least 6 of the 12 good search terms in our example), the number of documents necessary to sample can be sharply reduced, or, for the same searching effort, the estimates for $P(\text{Rel}|\,)$ can be improved.

There are several problems with the approach we have outlined. First, the speed up theorem is based on questionable assumptions. The assumption that each disjoint (complete cnf) subset formed from the underlying vocabulary, V, is of the same size may be a crude approximation to an "ordinary" document database partitioned by a dozen index terms in the way we suggest. The accuracy of this assumption depends on the co-occurrence pattern of the terms in V. Such co-occurrence patterns may vary considerably depending upon the depth of indexing (from a few manually chosen terms to a full-text representation) and the homogeneity of the document collection.

The second assumption, that each lattice element have the same probability of relevance, is clearly inaccurate. If it were not, then there would be no need to choose among the various subsets of documents at all! We can reanalyze the speed up associated with multiple accrediting under different assumptions. Let p_i be the probability that a lattice node whose description fails to use i search terms in its description has the current highest estimated probability of relevance. (For instance, ABD fails to use the i = two search terms "C" and "D" from the search vocabulary V = {A, B, C, D, E}.) Further, let p_{ji} be the probability that, when a node with i unused search terms in its description is intentionally sampled, the description of the document actually selected will lie j levels deeper in the lattice. Then, the speed up arising from multiple accrediting is:

$$\sum_{i=0}^{n-1} p_i \sum_{j=0}^{i} p_{ji} * 2^j \ .$$

Another problem that arises with this algorithm is that, even if lattice elements are correctly ranked by probability of relevance, the optimal presentation order may be different than such an ordering of lattice nodes. For instance, a ranking may properly rank lattice node ABC before AB, and AB before AC. Since AB and AC are both supersets of ABC, presenting all documents from lattice element ABC, then all those from AB, and then all from AC is equivalent to first presenting documents with descriptions ABC, then those with descriptions AB~C, and then those with descriptions A~BC (it is not necessary to present the same document twice). Suppose ABC has 1000 documents, 100 of which are relevant; AB~C has 100 documents, none of which is relevant, and A~BC has 1000 documents, 50 of which are relevant. Then, $P(\text{Rel}\,|\,\text{ABC}) > P(\text{Rel}\,|\,\text{AB}) > P(\text{Rel}\,|\,\text{AC})$. But, $P(\text{Rel}\,|\,\text{AB}\sim\text{C}) < P(\text{Rel}\,|\,\text{A}\sim\text{BC})$.

To properly utilize our algorithm, conditions are required for suggesting when an ordering of lattice element is consistent with an ordering of disjoint elements. For instance, $P(\text{Rel}\,|\,\text{ABC}) > P(\text{Rel}\,|\,\text{AB}) > P(\text{Rel}\,|\,\text{AC})$ implies $P(\text{Rel}\,|\,\text{ABC}) > P(\text{Rel}\,|\,\text{AB}\sim\text{C}) > P(\text{Rel}\,|\,\text{A}\sim\text{BC})$ *if* (but **not** only if) $P(\text{AB}) > P(\text{AC})$.§§ A partially ordered set of lattice nodes can be presented in a single order by a topological sort (Knuth, 1973) when necessary and sufficient ordering conditions cannot be met.

The algorithm suggested in this paper has many variations, including the choice of: G, the number of relevant documents for a node to become a goal; the set of nodes to be initially sampled; the number of goal nodes sought until the algorithm terminates; and the decision to sample with or without replacement. In addition, new search terms may be added to the lattice based on evidence of their usefulness part way through a search.

It is clear that this algorithm is not suited to searches on small document databases where a searcher would not tolerate having to evaluate many prospective relevant documents just to improve the order in which a larger set of documents is presented. On the other hand, in very large document databases it is necessary for searchers to have assistance to navigate through overlapping document subsets (lest they make these decisions randomly) and to use information about both retrieval successes and failures wisely in deciding which subset to retrieve a document from next. The method we propose presents a theory for accomplishing these ends. In addition, this algorithm may provide a basis for other search methods. For instance, tentative, initial evaluations of relevance may some day be made satisfactorily by machine, with our algorithm controlling which subsets the machine evaluates next and, thus, minimizing the number of such evaluations. The multiple accrediting from this algorithm may also be incorporated in other retrieval techniques.

Acknowledgements—Choon Lee and Peter Lenk helped me present certain theoretical results more simply. Dave Blair gave me feedback that I have incorporated into this paper. Manfred Kochen, who died recently, pointed out the ties between these ideas and other work. All are from the University of Michigan. I also thank Richard Belew, UCSD, and the anonymous referees for their comments.

§§A less intuitive **necessary** and sufficient condition can be derived, too:

$$\frac{P(\text{ABC})}{P(\text{AB})} > P(\text{Rel}\,|\,\text{AC}) - P(\text{Rel}\,|\,\text{AB}) + \frac{[P(\text{Rel}\,|\,\text{AB}) - P(\text{Rel}\,|\,\text{ABC})] * [P(\text{Rel}\,|\,\text{ABC})/P(\text{AC})]}{P(\text{Rel}\,|\,\text{AC}) - P(\text{Rel}\,|\,\text{ABC})} \ .$$

REFERENCES

1. Blair, D.C. (1980). Searching biases in large interactive document retrieval systems. *Journal of the American Society for Information Science*, July, 271–277.

2. Feller, W. (1968). *An Introduction to probability theory and its applications*. Vol 1. New York: John Wiley & Sons.

3. Kahneman, D., Slovic, P., & Tversky, A. (Eds.) (1982). *Judgment under uncertainty: heuristics and biases*. Cambridge: Cambridge University Press.

4. Knuth, D.E. (1973). *The Art of Computer Programming*. Vol 3. Reading, MA: Addison-Wesley.

5. Nilsson, N.J. *Principles of Artificial Intelligence*. (1980). Palo Alto, CA: Tioga Publication Co.

6. Robertson, S.E. (1977). The probability ranking principle in IR. *Journal of Documentation*, *33*, 294–304.

7. Tague, J., McClellan, C. & Nelson, M. (1984). The hyperterm model of a bibliographic database. *Canadian Journal of Information Science*, *9*, 37–58.

8. van Rijsbergen, C.J. (1979). *Information Retrieval*, Second Edition. London: Butterworths.

9. van Rijsbergen, C.J. (1977). A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation*, *33*, 106–119.