COMMUNICATION SCIENCES

AND

ENGINEERING

VII. PROCESSING AND TRANSMISSION OF INFORMATION[*]

Academic and Research Staff

Prof. P. Elias	Prof. M. E. Hellman	Prof. R. S. Kennedy
Prof. R. G. Gallager	Prof. E. V. Hoversten	Prof. C. E. Shannon

Graduate Students

P. Alexander	M. A. Freedman	P. F. Moulton
J. R. Clark	J. G. Himes	A. J. R. Muller
S. J. Dolinar, Jr.	J. E. Just	B. Neumann
J. C. Eachus	M. J. Marcus	R. S. Orr
R. A. Flower	G. Q. McDowell	T. A. Welch
	L. S. Metzger	

A. SEARCH LENGTHS IN FILE SYSTEMS WITH DIRECTORIES[†]

The quality of a directory (catalog) in a file system is measured by how well it narrows the average field of search in the main file for an ensemble of user questions. An ideal directory will pinpoint every relevant item in the file for every user question, while a less perfect directory will supply a longer list of items, including some irrelevant entries. A quantitative statement of expected search length for a given directory requires an estimate of two parameters:

1. Size of the ideal directory, which reflects the number of file items and the statistics of the descriptive attributes used to form questions.

2. Statistics of expected usage; that is, the relative frequency of all questions that might be asked and the expected recall level (completeness of search) desired by each user.

This report outlines an investigation of these parameters for a typical library file, characterized by (a) low update rates, and (b) extensive user request flexibility, where each file item may be specified by many different user questions. We shall briefly describe a file model developed in doctoral thesis research,[1] and summarize the principal results of this work.

In this mathematical analysis of directories two conceptual tools of some interest are employed. First, we define an ideal directory against which all real implementations can be compared. Second, we define an ensemble of directories, reflecting the various configurations of file data that a file user might expect to encounter. Various directory construction schemes can then be compared as they perform on a given ensemble.

[*]This work was supported in part by the Joint Services Electronics Program (U.S. Army, U.S. Navy, and U.S. Air Force) under Contract DA 28-043-AMC-02536(E).

[†]This work was supported in part by Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Nonr-4102(01).

A general lower bound on search lengths is constructed, based on the difference between the information in the ideal directory and the information in a real implementation. The interpretation of this bound serves as a guide for constructing directories and measuring their performance.

1. Directory System Model

An ideal directory f is defined as a binary matrix having.

r attributes (columns)	$a_1 \ldots a_i \ldots a_r$
s items (rows)	$b_1 \ldots b_n \ldots b_s$
a binary relationship	$f(a_i b_n) = 1$ or 0.

A question ensemble Q is a set of 3^r questions q, each composed of

an r-bit vector of attribute values, each specified to be 1, 0, or $\emptyset$ (don't care)

a probability of occurrence $p_Q(q)$.

Together, Q and f define a file, where an item b_n is said to be relevant to a question, q iff, for every a_i:

$$f(a_i b_n) = 1 \ \& \ q(a_i) = 1 \text{ or } \emptyset$$

or

$$f(a_i b_n) = 0 \ \& \ q(a_i) = 0 \text{ or } \emptyset.$$

Notes. This is the intersection form of question, where an item must fit all specified attribute values of q to be considered relevant, and both 1 and 0 in f fit a $\emptyset$ in q.

The directory is ideal in the sense that the user agrees with the evaluations of relevance so defined, under the assumption that all semantic problems have been overcome.

The use of f enters when an item b_n is found relevant to a user question q; then, the item identifier n is returned to the user to identify the relevant item in the main file.

The ordering of rows of f is determined by the main file, and this is assumed to be uncorrelated with the attribute information.

One restrictive assumption is made, that the items of a file are statistically independent of each other. This permits an item-by-item analysis, which reduces the notational complexity in the following analysis. This assumption causes only small errors, within the context that attribute statistics are already fully specified. The existence of one item in a library cannot prejudice the existence of another item unless they are related because of factors outside the attribute values such as being two volumes in a set.

Item Configurations. An item can take on a binary configuration c with probability $p_C(c)$. This defines the ensemble of configurations C, which has 2^r members. These probabilities are defined by the user's knowledge of possible file contents, not by a particular file. The user's uncertainty about an item's configuration is measured by

$$H(C) = -\sum_{C} p_C(c) \log p_C(c). \tag{1}$$

(Note that throughout this report log x means $\log_2 x$.) The maximum value of H(C) is r, which occurs when attributes take on values 1 and 0 equally probably and independently within C. When the attributes have known correlations or unequal 1, 0 probabilities, $H(C) < r$ results.

The ideal directory is just s samples from H(C), so this defines an ensemble F of directories (files), and an entropy

$$H(F) = s \cdot H(C) \leqslant r \cdot s. \tag{2}$$

Implementation. An ideal item configuration c is represented in the real world by a representation e. There exists an ensemble E of these e's with each member having a probability of occurrence $p_E(e)$. Thus an entropy H(E) is defined (as in Eq. 1), which in general will be smaller than H(C). The probability of occurrence $p_E(e)$ is based on the probabilities of occurrence of the c's that it represents. In some cases, several e's may be alternative representations for one c, so exact statements about H(E) must await the study of special cases.

The representation of an f is defined to be a g, which with its probability $P_G(g)$ is a member of the ensemble G; $H(G) = s \cdot H(E)$. The physical meaning of H(G) is straightforward, being the number of bits of information used to construct the file directory. If the directory is a table of values, this bit count is obvious. If it is a hash-code algorithm, the number of bits in the stored program, which determines how many different algorithms could be used, is H(G).

Whenever a particular representation e occurs, this means that one of the c's which it represents would have occurred had the implementation been an ideal directory. For each such e, a conditional probability $p(c|e)$ is therefore meaningful. Then $H(C|e)$ is the entropy of these c's, and

$$H(C, E) = H(E) + \sum_{E} p(e) H(C|e) = H(E) + H(C|E). \tag{3}$$

2. Total-Recall Example

To illustrate the nature and meaning of the directory analysis, a simple example will be examined. It has restrictions on the question set and on implementation techniques allowed, to simplify the mathematics. While the restrictions may seem harsh, the model does actually fit a number of interesting situations.

1. Questions

(a) Total Recall – Every user question is assumed to require total recall, so every item that might possibly fit the question is retrieved.

(b) Attribute Values – Every attribute has the same fixed probability of being specified 1 or 0, and all attribute values are specified independently. Thus

$$p_Q(a_i=1) = p_Q(a_i=0) = \frac{1}{2} p_a$$
$$p_Q(a_i=\emptyset) = p_\emptyset = 1 - p_a. \tag{4}$$

2. Implementations

(a) Partition – Only implementations that partition the set F are considered, so that any one c is always represented by the same e, and

$$H(C|E) = 0. \tag{5}$$

(b) Representation – Only those implementations are considered which can be represented by an $r \times s$ matrix whose elements are 1's, 0's, and $\emptyset$'s.

For examples of implementation that fit these restrictions, consider two $r=2$ implementations.

(a) One attribute can be just ignored (with an obvious saving of storage space), so that, say, $c_2 = 10$ and $c_3 = 11$ are stored as $1\emptyset = e_1$, and $c_0 = 00$ and $c_1 = 01$ are stored as $0\emptyset = e_2$.

(b) The two attributes can be combined:

$c_0 = 00$ is stored as $00 = e_1$

$c_1 = 01$ is stored as $01 = e_2$

$c_2 = 10$ and $c_3 = 11$ are stored as $1\emptyset = e_3$.

(Three configurations are cheaper to store than four.)

Items are retrieved whenever the question fits the representation e, with a $\emptyset$ in e being interpreted to fit both a 1 and 0 in q. In Example (b), if $q = 11$ is asked of a file represented by e_3, the item involved will be retrieved whether its true identity is 11 or 10.

Now we consider the probability of relevance of a particular item b_n when it is retrieved through a particular representation e. The probability of relevance, p_n, is the probability that the item, when retrieved, actually fits the question asked, averaged over all questions and all configurations:

If e contains no $\emptyset$'s, $p_n = 1$ (or else it is not retrieved).

If e contains one $\emptyset$, in attribute a_i,

$$p_n = 1 \quad \text{if } a_i(q) = \emptyset$$

$$p_n = p_C(a_i = 1) \quad \text{if } a_i(q) = 1$$

$$p_n = p_C(a_i = 0) \quad \text{if } a_i(q) = 0$$

$$\text{Avg}(p_n) = 1 \cdot p_\emptyset + p_a\left[\tfrac{1}{2} p_C(a_i = 1) + \tfrac{1}{2} p_C(a_i = 0)\right]$$

$$= p_\emptyset + \tfrac{1}{2} p_a = 1 - \tfrac{1}{2} p_a.$$

(Note that this simplification is possible only because $p_Q(a_i = 1) = p_Q(a_i = 0)$; other cases are more complicated.)

If e contains x $\emptyset$'s,

$$\text{Avg}(p_n) = \left(1 - \tfrac{1}{2} p_a\right)^x.$$

It is more convenient to deal with log p_n, so if e has x $\emptyset$'s,

$$\log \overline{p_n}(e) = x \log \left(1 - \tfrac{1}{2} p_a\right). \tag{6}$$

But now observe that $x \geq H(C|e)$, since with x $\emptyset$'s in e there are at most 2^x c's represented by e; if these are equally probable, then $H(C|e) = x$, but if they are unevenly distributed, $H(C|e) < x$ because $\log \bar{z} \geq \overline{\log z}$. Therefore, from Eq. 6,

$$\log \overline{p_n}(e) \leq H(C|e) \log \left(1 - \tfrac{1}{2} p_a\right). \tag{7}$$

Averaging this over all e's, we obtain

$$-\sum_E p(e) \log \overline{p_n}(e) \geq -H(C|E) \log \left(1 - \tfrac{1}{2} p_a\right)$$

$$\geq -\log \left(1 - \tfrac{1}{2} p_a\right) [H(C)-H(E)]. \tag{8}$$

Relation (8) occurs because

$$H(C|E) - H(E|C) = H(C) - H(E),$$

which is the result of taking the average logarithm of

$$p(c|e) \cdot p(e) = p(e|c) \cdot p(c),$$

and $H(E|C) = 0$ by 4.5. Now Eq. 8, summed over all items retrieved in a file, using

$$H(G) = s \cdot H(E) \quad \text{and} \quad H(F) = s \cdot H(C),$$

gives

$$-\sum_n \text{average } (\log \overline{p_n}) \geq \log \left(1 - \frac{1}{2} p_a\right)[H(F)-H(G)]. \tag{9}$$

Simple checks show how this bound works. If $H(G) = H(F)$ (the directory contains all information needed for ideal access), then Avg $(\log \overline{p_n}) = 0$. This means that $p_n = 1$ for all items accessed, and hence it is the mathematical description of perfect retrieval. If $H(G) = 0$, then

$$\overline{p_n} \leq \left(1 - \frac{1}{2} p_a\right)^{H(F)/s}.$$

This turns out to be the average probability (not proved here) that a randomly selected item fits a question. It is the level of relevance that would be measured if all items were retrieved randomly in an exhaustive search of the file.

This gives a definite upper bound on $\overline{p_n}$. It also helps describe the trade-offs available for deciding how to allocate directory information among attributes and items. A principal result here is that the error level in the directory performance is linear with the difference between H(F) and H(G).

In summary, the derivation is threefold.

1. Observing how many file configurations are represented by a single implemention state.

2. Observing that an item must be considered for retrieval if any of these possible configurations would fit a given question.

3. Calculating how often the item is erroneously retrieved because the question fitted one configuration under the specified representation, but another had actually occurred. The last calculations are very dependent on the statistics of the question ensemble, Q, especially to the extent that Q distinguishes the various configurations (c's) represented by a single implementation state (e).

3. General Results

The line of reasoning used here can be extended to more general cases in a slightly different form. For a number of file and question ensembles, a rigorous statement can be made about the average of log p_n.

$$\overline{-p_n \log p_n} \geq K[H(F)-H(G)], \tag{10}$$

where K depends on file and question statistics, but not on the form of implementation; and equality occurs under the following conditions:

(a) Each item in F has a unique representation in G; that is, there is no ambiguity in the indexing process that creates the directory.

(b) The members of G can be described without correlations; that is, the information in G is not wasted in describing inter-attribute mutual information.

This relation has not been proved valid for all file situations, but it has been found to be a good estimate in cases in which it is not a strict inequality. Details of the range of proved validity may be found in the author's thesis.[1]

Implications. The relation (Eq. 10) says that a given directory size, H(G), determines a lower bound on $-\log p_n$ averaged over all questions and item configurations. As the bounding value goes up, the values of p_n (relevance) must become smaller to preserve the inequality. The average value of p_n is thus set by H(G) alone, and file organization can only serve to adjust the distribution of p_n values among the items.

For user questions desiring total recall, the best distribution is to have all nonzero p_n equal; this minimizes the number of nonzero p_n's (items that must be searched to achieve total recall). The bound can then be solved to show that, for simple cases,

$$\text{Total-Recall Search Length} \geqslant s^{1-\alpha}, \tag{11}$$

where $\alpha = \frac{H(G)}{H(F)}$, for the case in which each question has an expected number of relevant items $= \Sigma\, p_n = 1$. Note that when H(G) is a small fraction of H(F) (a typical case) and the number of file items s is large (say, 10^5), search lengths will be intolerably long on the average (say, 10^4 documents).

The file designer then has the choice of redistributing his p_n values to achieve different system objectives. First, he can unevenly distribute the p_n's so that some items are more probably relevant than others, but with an increase in the total number of items having a nonzero p_n. Then low-recall questions can be answered with a few high p_n references, giving lower average search lengths for these questions. This is achieved at the cost of increased search length for total recall. Second, the designer can unevenly distribute the p_n's over questions, so that some questions receive preferential treatment and the others have very long searches. This is effective when H(G) is so small that average search length is intolerably long; at least some questions can be given decent handling if the p_n's are unbalanced. At present, libraries use the last approach, thereby giving very good access for questions that may be answered from title or author attributes, while giving poor response to questions that seek specific subject matter. The result of this is that potential users who want "fact" retrieval rather than "document" retrieval take their questions to some other pool of knowledge than a library.

Other design decisions available to the file designer involve making sure that the directory bits are used efficiently, and none of H(G) is wasted. This method involves selecting an efficient coding so that correlations do not waste bits, and striving to reach equality in the bound (Eq. 10) by minimizing the magnitude of the conditions causing inequality. None of these design decisions would surprise an experienced reference librarian.

File Ordering. An interesting conclusion from this model and approach is that the ordering of items in the file (shelf sequence) can be analyzed as part of the directory function. The information used to determine file ordering is indistinguishable from information stored in the directory, so H(G) is actually the sum of these two informations. Thus all trade-offs in determining p_n's apply to the selection of criteria for file ordering, as well as to the selection of attribute information for the directory.

T. A. Welch

References

1. T. A. Welch, "Bounds on Information Retrieval Efficiency in Static File Structures," Ph.D. Thesis, Department of Electrical Engineering, M.I.T., May 1971 (will also be available as Project MAC Report TR-88).