ON THE STORAGE ECONOMY OF ERROR-TOLERATING QUESTION-ANSWERING SYSTEMS

Judea Pearl
School of Engineering and Applied Science
University of California, Los Angeles
Los Angeles, California, U.S.A.

Abstract

The possibility of saving various computational resources is an argument often advanced in favor of permitting question-answering systems to make occasional errors. In this paper we establish absolute bounds on the amount of memory savings that is achievable with a specified error level for certain types of question-answering systems. Question-answering systems are treated as communication channels carrying information concerning the acceptable answers to an admissible set of queries. Shannon's rate-distortion theory is used to calculate bounds on the memory required for several question-answering tasks. For data retrieval, pattern-classification, and position-matching systems it was found that only small memory gains could be materialized from error-tolerance. In pair-ordering tasks on the other hand, more significant memory savings could be accomplished if small error-rates are tolerated. Similar limitations govern the tradeoffs between error and computation time.

1. Introduction

Loosely speaking a question-answering (QA) system 1s any mechanical system which accepts data of a specified type and at a later time produces output symbols in response to queries from some admissible set. The output symbols are interpreted as answers to the queries about the specific data-set and are expected to conform to some acceptable test of correctness. Practically all question-answering systems in operation today are deductive in nature. When a query is presented concerning an assertion not derivable from the data, either the query is declared inadmissible, or the data is declared incomplete.

It is apparent that human information processing is not confined to deductive reasoning; qualifiers such as 'probably', 'perhaps', '(doubtfully', and many more are *very* common in *everyday* language. People quite often assert or deny propositions which are either themselves incomplete, concern incomplete data, or are too complicated to be computed within the available time constraint. We usually interpret such behavior as an attempt to economize some computational resource (e.g., data specificity, storage space, search time) at the expense of answer accuracy. Correspondingly, it has been suggested that a substantial amount of computational resources could be gained if one allowed computerized QA systems to make occasional errors via approximate representations of data or world-models. Zadeh [1] has proposed the use of fuzzy-sets to cope with the "incompatibility principle." In Zadeh*s words: *"the essence of this principle is that as the complexity of a system increases, our ability to make precise and yet significant statements about its behavior diminishes until a threshold is reached beyond which precision and significance (or relevance) become almost mutually exclusive characteristics."* Rabin [2] conjectured that the disparity between the super-exponential complexity of proofs in even the simplest algebras and the apparent simplicity of everyday human planning behavior can perhaps be explained by man's willingness to tolerate a small amount of error.

These notion, while in harmony with one's intuition and experience, have not yet been submitted to a careful quantitative analysis. This paper should be considered as a step toward such analysis. It concerns the economy of storage space and establishes absolute bounds on the amount of storage that can be saved by tolerating a specified percentage of errors in certain simple, yet typical, QA tasks. As a starting point, we ignore the question of computational-complexity or search-time, thus allowing the QA system any required (but finite) amount of time for organizing and filing data into storage, as well as for computing the answers from the condensed data. A more recent work [3] demonstrates that the relative savings of most complexity measures in use are asymptotically bounded by the relative savings of storage space. Hence, the latter should be considered a fundamental characteristic of error-tolerating computations.

Four question-answering tasks are analyzed: data-retrieval, pattern-classifications, position-etching and pair-ordering. The first and second tasks" are concerned with the economical description of one-place predicates, while the third and fourth concern the description of an order relation.

Data-retrieval QA systems accept as data an arbitrary subset of individuals (from some universe), and respond to queries of the type "has this individual been observed before?" A typical example would be the task of determining whether a certain node in a game tree has been expanded before, or whether it has been recognized before as a "forced-win" position. A pattern-classification system accepts data in the form of a subset of individuals classified into ℓ mutually exclusive categories (property-values). It responds to queries demanding the proper classification of a given individual from the observed list, permitting an arbitrary classification of unobserved individuals.

Both position-matching and pair-ordering systems accept data in the form of an ordered list of individuals. Position-matching systems admit queries of the type "What individual stands in the kth position?" or "Who is the immediate successor of individual x?" Pair-ordering systems respond to queries of the type "Does x precede y?". The numerical encoding of the order in position-matching tasks is known as a nominal scale (e.g., the number on the shirts of football players) and that of pair-ordering tasks as ordinal scale (e.g., ranking of players in order of ability).

In all these tasks it is possible to answer all queries correctly if an exact replica of the original data is kept in storage. In many cases,

however, the memory required to hold such a replica would be prohibitively large. Consider, for example, the task of deciding which member of a given pair of chess configurations has proven more promising in past games. Such tasks are never handled by complete enumeration but rather by employing approximate models which allow the computation of an order-preserving scale from some aspects of the configurations. The memory space needed for storing these imprecise models is expected to be smaller than that required for storing the scale itself, and much smaller than that required to store the primary observational data from which the scale was deduced. Naturally, the amount of damage caused by impreciseness depends on the purpose served by the model, as reflected by the type of task-related queries that the model is expected to resolve. We therefore define the measure of the task-related damage (or distortion) as the fraction of queries in the admissible set which are answered incorrectly.

This paper demonstrates that the conversion of a small error allowance into a significant memory saving requires highly redundant query sets, and that such conversion is not feasible for the tasks of data-retrieval, pattern-classification and position-matching unless, perhaps, the data is highly structured.

2. Background and Nomenclature

The model chosen for defining QA systems is represented by the block diagram of figure 1*. It consists of three parts: a storage procedure B,.. , a retrieval procedure B,. ., and a memory S. During the 'filing' phase, B-., examines the elements of a data set and files a 'summary' in S for use by another procedure B-find .. Later, in the 'finding' phase, B_{find}. . uses the information in the memory to make decisions about the queries. Our aim is to calculate the minimum quantity of data that B*... must store in S in order to guarantee a minimally acceptable performance standard.

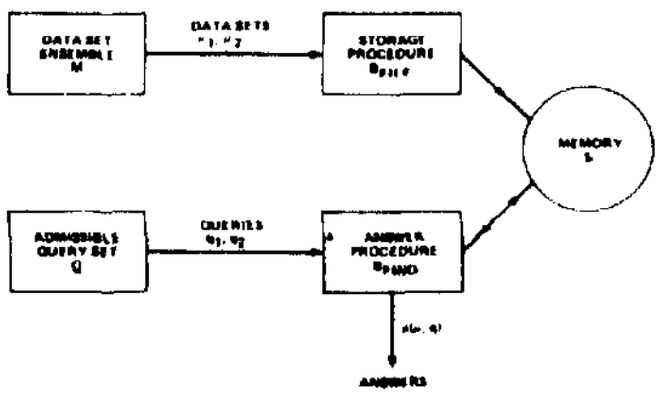

Figure 1. Block Diagram Model of a Question-Answering System.

The model is made precise using the following formulations: Let $\underline{M}$ be a collection of data-sets, $\underline{M} = \{\mu_1, \mu_2, \cdots \mu_M\}$, and $\underline{Q}$ a set of admissible queries on $\underline{M}$, $\underline{Q} = \{q_1, q_2 \cdots q_Q\}$. A QA system is a finite state machine QA with input alphabet $\underline{M} \cup \underline{Q}$, output alphabet $\underline{A} \cup \phi = \{a_1, \cdots a_K\} \cup \phi$, a finite state-set $\underline{S} = \{\sigma_1, \sigma_2, \cdots \sigma_S\}$, and a pair of characterizing functions: an output function f_A: $(\underline{M} \cup \underline{Q}) \times \underline{S} \rightarrow (\underline{A} \cup \phi)$, and a next-state function f_S: $(\underline{M} \cup \underline{Q}) \times \underline{S} \rightarrow \underline{S}$ satisfying

$$f_A(\mu_i, \sigma_j) = \phi \qquad (2.1a)$$

$$f_A(q_i, \sigma_j) = a_k \qquad (2.1b)$$

$$f_S(q_i, \sigma_j) = \sigma_j \qquad (2.2a)$$

$$f_S(\mu_i, \sigma_j) = f_S(\mu_i) \qquad (2.2b)$$

The output symbol ϕ represents the system's output during the data-filing process. (2.1a) and (2.1b) express the fact that during the data-filing process we ignore the system output, while during the query process we wish the output to represent an answer to the query. (2.2a) constrains the machine from changing its state in response to a query, while (2.2b) forces the machine to clear its state before the application of a new data. The storage capacity of the machine is defined to be the size of the state-space S.

It is now necessary to define a measure of machine performance. Let $a(\mu,q) \in \underline{A}$ represent the answer selected by the machine in response to query $q \in \underline{Q}$, with $\mu \in \underline{M}$ being the most recently examined data. We define a real-valued distortion function δ: $\underline{A} \times \underline{M} \times \underline{Q} \rightarrow Re^+$ so that $\delta[a,\mu,q]$ would measure the degree of damage caused by answering 'a' to query 'q' about the data 'μ'. We may assume without loss of generality that every (μ,q) pair possesses at least one $a_T \in \underline{A}$ for which $\delta[a_T,\mu,q] = 0$, representing a "correct" answer to a query. The overall system performance is judged by the mean distortion measure:

$$D = \sum_{\mu \in \underline{M}} \sum_{q \in \underline{Q}} P(\mu,q)\ \delta[a(\mu,q),\ \mu,\ q] \qquad (2.3)$$

where $P(\mu,q)$ is the joint probability that data μ would be presented followed by query q.

When no *a priori* knowledge is available on the statistical properties of $\underline{M}$ and $\underline{Q}$. it is reasonable to assume that all queries and data are equiprobable and independent of each other, thus making $P(\mu,q) = (MQ)^{-1}$, and:

$$D = \frac{1}{MQ} \sum_{\mu \in \underline{M}} \sum_{q \in \underline{Q}} \delta[a(\mu,q),\ \mu,\ q] \qquad (2.4)$$

Let $\underline{a}(\mu) = a(\mu, q_1), \cdots, a(\mu, q_Q)$ stand for the answer-string obtained when an ^{Q} ordered sequence of queries $q_1 \cdots q_Q$ is applied after data-set μ was examined. The mean distortion can be written as:

$$D = \frac{1}{M} \sum_{\mu \in \underline{M}} d[\underline{a}(\mu),\ \mu] \qquad (2.5)$$

where:

$$d[\underline{a}(\mu),\ \mu] = \frac{1}{Q} \sum_{q \in \underline{Q}} \delta[a(\mu,q),\ \mu,\ q] \qquad (2.6)$$

Throughout this paper we will consider only the symmetrical distortion function whereby all deviations from the set of true answers $\underline{A}_T(\mu,q) = \{a_T: \delta(a_T,\ \mu.\ q) = 0\}$ have equal weights. This leads to

$$\delta(a,\ \mu,\ q) = \begin{cases} 0 & \text{if } a \in \underline{A}_T(\mu,\ q) \\ 1 & \text{otherwise} \end{cases} \qquad (2.7)$$

*We are using here the terminology of Minsky and Papert, Reference 4, Chapter 12.

For the purpose of calculating the minimum size of S it is convenient to regard a QA system as a communication channel which receives at Its input the data-set p and reproduces at Its output the answer-string a. Thus, the source alphabet 1s M, and the reproducing alphabet, A, 1s the collection of all possible answer-strings, $A = A^{Q}$.

Let i designate an Input data and j an output answer-string, and let dij be the (per letter) distortion measure between the two. Shannon's rate-distortion function for this source 1s defined by [5]:

$$R(D) = \min_{P(j|i)\in P_D} (M,A) =$$

$$= \min_{P(j|i)\in P_D} \sum_{i,j} P_i\, P(j|i) \log \frac{P(j|i)^*}{P_j} \qquad (2.8)$$

where I is the average mutual information between M and A, and PQ is the set of all conditional probabilities yielding a mean distortion not exceeding D:

$$P(j|i) \in P_D \quad \text{iff} \quad \sum_{i,j} P_i\, P(j|i) d_{ij} \le D \qquad (2.9)$$

Shannons source-coding theorem states that it is not possible with any coding scheme to transmit the source Information through any channel of capacity less than R(D) with average distortion less than D. Conversely, the positive form of the theorem states that given any channel with capacity C > R(D), coding schemes exist which result in an average distortion arbitrarily close to D when used over this channel. If one regards B*., [1n figure 1] as the source encoder, B find, . as the decoder, and the memory S as a noiseless channel with capacity S, then Shannon's source-coding theorem implies that in order for a QA system to exhibit a mean distortion not exceeding D, it must be provided with a memory space at least as large as R(0). Conversely, the theorems guarantee that it is possible to realize a mean error arbitrarily close to D with an average memory space not exceeding R(D). The latter is true only if coding very long blocks of data is permitted, such as would be required for serving many users. If a code is to be assigned to each separate data-set as demanded by (2.2b), then R(D) only provides a lower bound on the memory space required.

To obtain the rate-distortion function for a QA system one needs to minimize expression (2.8) with $P_1 = 1/M$ and dij given by (2.6). Shannon has shown that R(D) is a continuous, strictly decreasing, convex U function for $0 < D < D_{L,}$. where D

— — max max

is the minimum distortion achievable with zero memory:

$$D_{max} = \min_{j} \sum_{i} P_i\, d_{ij} \qquad (2.10)$$

and

$$R(D) = 0 \quad \text{for} \quad D \ge D_{max} \qquad (2.11)$$

The slope R'(D) is always infinitely negative at D = 0 but may be discontinuous at $D = D_{max}$.

*A natural logarithm is used in the definition of R(D). Consequently memory sizes will be given in nats. A division by $\log_e 2$ is required for conversion into bits.

The calculation of the rate-distortion function for an arbitrary distortion matrix Is, in general, a difficult task. For our purposes, however, it would be sufficient to derive bounds on R(D) using well known methods of rate-distortion theory [6]. The derivations of these are omitted in this paper and can be consulted in references [8], [9] and [10].

3. Predicate-Related QA Systems

Data-Retrieval Systems

Consider a collection C of m objects from which a subset C of n objects 1s selected and presented to a QA system during the filing phase. At a later time queries of the type "has object x(x€c) been observed?" are presented to the system for confirmation or rejection. An error 1s committed when either a member in C is declared "unobserved" or an object in C -C fs declared observed".

It is clear that if no error at all can be tolerated, then at least $\log_2\binom{m}{n}$ bits of storage must be devoted to describe the observation set. When memory size is limited and $\binom{m}{n}$ large (a typical example: $m = 2^{100}$ $n = 2^{20}$), one 1s tempted to "summarize" the data at the expense of accuracy. The most straightforward way to reduce storage would be to simply ignore a portion of the data, and produce arbitrary answers to queries referring to the missing objects. We shall call this scheme "data-erasure", and will use it to gauge the performance of more sophisticated description schemes.

The Pattern-Recognition literature [7] may provide many ideas for such schemes. One may be tempted, for instance, to store only 'representative' samples of the observed set, and base the decisions on the 'proximity' between the query object x and the stored 'representatives'. Another possibility is to approximate the binary representation of the observed sample by a Boolean expression of a limited length, store this expression, and use it in the decision phase. A more popular method is provided by the so-called "Statistical Pattern Recognition". Here the data-set 1s merely used to calculate a set of parameters governing certain statistical models; the parameters are stored and subsequently used *to* generate the answer with the highest probability of correctness. An even more sophisticated description'method can be devised, based on the so-called "Linguistic Pattern Recognition" philosophy. The names of the observed objects are regarded as sentences in some language, a grammar generating this language is Inferred and stored by B_{file}, while $B_f\hat{}_{nd}$ parses the query sentences and decides whether it is in the language. It will be shown that in the absence of any special structures underlying the selection of the subsets C and for large m, all such schemes possess memory vs. error characteristics equal or inferior to those obtained by data-erasure.

The data-set space, for this problem, consists of all subsets of C_m which contain exactly n

A discussion of the storage vs. search-time tradeoff for such tasks is qiven by Mlnsky and Papert [4].

objects, while the query set 1s isomorphic to C . Therefore:

$$M = \binom{m}{n}\ ,\quad Q = m\ ,\quad H = \log\binom{m}{n}\ , \qquad (3.1)$$

Every answer string constitutes a dichotomy of C_m, so $\underline{A}$ is simply the power set of C_m. The distortion measure d_{ij} is given by 1/m times the number of objects in data-set i$\in \underline{M}$ which are misclassified by answer-string j$\in \underline{A}$.

The rate-distortion function governing this system was found to be [9]:

$$R(D) = \log\binom{m}{n} + m[D \log D + (1-D)\log(1-D)] \qquad (3.2)$$

where D stands for the fraction of mislabeled objects. R(D) is depicted in Figure 2 for several values of n, and m = 10^6.

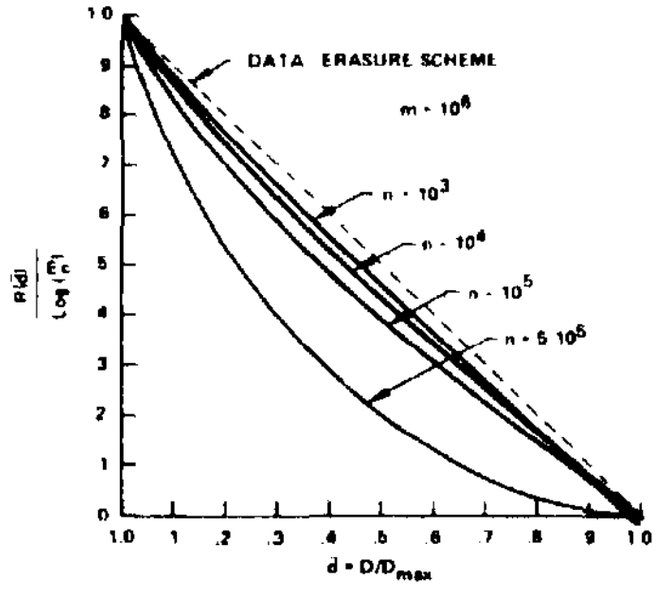

Figure 2. Bounds for Storage vs Error Curves of Item-Retrieval System with Known Sample Size.

It demonstrates that as the ratio m/n increases, the rate-distortion function approaches the data-erasure line 1 - D/D_{max} at a logarithmic rate. Thus, *no amount of error* (below the trivial level of D = n/m obtainable by answering all queries in the affirmative) *can make memory demands grow slower than n log m/n.*

The implications of these results for the prospects of finding a general language that simplifies descriptions of subsets are rather discouraging. They essentially imply that *no filing scheme exists which performs appreciably better than one which simply ignores a certain fraction of the observed data.* Only for sample sizes on the order of m/10 can the statistical dependence among the samples be utilized for achieving a somewhat higher memory saving.

This does not mean, however, that more sophisticated schemes cannot provide better storage-error characteristics under special circumstances. If, for instance, there exists a strong *a priori* knowledge that the observed sample is in some sense compact, then it becomes quite reasonable to store only descriptions of its boundaries, not the entire sample. The results obtained here refer to query sets and data sets which are uniformly and independently distributed. They are valid in cases where no structural knowledge exists (except the sample size n); any combination of n objects from C_m could constitute a data set and any member of C_m could represent a query.

Preliminary investigation of the effect of structure (using Markovian model) shows that the essential features of the memory vs. error characteristics remain unaltered with the introduction of structure into the data. While the memory required to achieve zero error is substantially reduced, the additional percentage reduction in memory requirement induced by error-tolerance remains close to that of structureless data.

Pattern Classification Systems

Consider a collection C of objects (or patterns) which is partitioned into three classes: n objects are in class C , n in C", and the remaining m-2n objects in class C°. During the filing phase a QA system is shown the elements of C and C" with their proper class identity. During the query phase an arbitrary object from C is presented, and the system is asked to classify it either as C or C". An error is counted whenever an element of C or C" is misclassified, but not when an element of C° is labeled (+) or (-). C^+ and C* stand for what is commonly called the "training-set", while C° represents unobserved patterns. The neutrality of C° reflects our commitment to evaluate system performance strictly on the basis of observed data, leaving aside considerations of inductive-power.

It is clearly possible to answer all queries correctly if we store the identities of the n elements of C , We simply label every query (+) if it is in storage and (-) if it is not. This scheme requires on the order of n logo m bits of memory and thus could become prohibitively large for practical values of the sample-size and the data dimensionality (e.g. $m = 2^{10}$° $n = 2^{20}$). Most pattern classification methods, however, are motivated by the belief that classification systems lend themselves to a greater storage economy. It is prompted by noting that unlike data-retrieval systems the exact reconstruction of the data is not required; only information pertaining to the difference between the classes need be stored. It follows that one can always add any number of elements from C to either C or C~ at no extra cost, and s> simplify the description of the separating rule. It will be shown that this argument is justified; the number of dichotomies needed for correct separation of all pairs of classes is much smaller than the number of all possible choices of C . Thus, storing the description of any one such dichotomy could be made more economical than storing the exact description of C .

Consider the multi-category classification problems whereby a description of an ℓ-chotomy is to be found, and the sample set contains n objects in each class C_i, i = 1, 2, ···, ℓ, and m - ℓn_i objects in C_0. The rate-distortion function for this problem is given by [9]:

$$R(\bar{d}) = \ell n[\log \ell + \bar{d} \log \bar{d} + (1-\bar{d}) \log (1-\bar{d}) - \bar{d} \log (\ell-1)] \qquad (3.3)$$

where $\bar{d}$ is the number of misclassification errors per sample:

$$\bar{d} = \frac{m\ D}{n\ \ell} \qquad (3.4)$$

It 1s depicted in Figure 3; as the number of categories Increases the normalized rate-distortion curves approach the line 1-7.

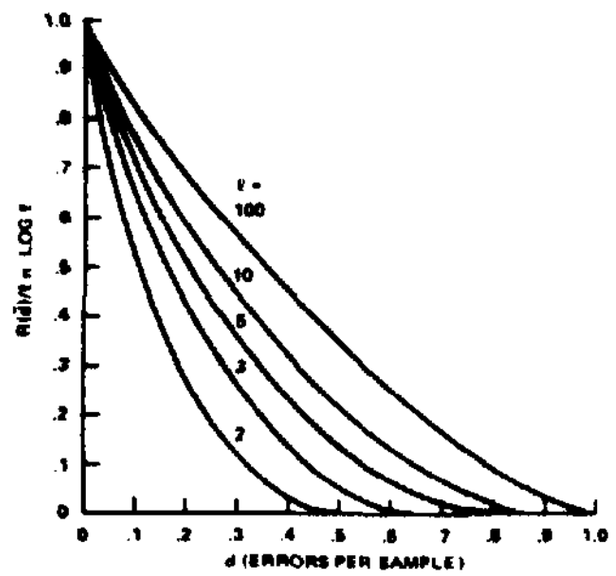

Figure 3. Normalized Rate-Distortion Function for ℓ-Category Classification.

In particular, for $\ell = 2$ we obtain:

$$R(\bar{d}) = 2n[\log 2 + \bar{d} \log \bar{d} + (1-\bar{d})\log(1-\bar{d})] \quad (3.5)$$

Note that the memory required for classification is linear with n, the sample size, but is independent of the size of the sample space, m.

It is not at all obvious that storage requirements should remain finite as $m \to \infty$. Consider, for instance, the requirement for $\bar{d} = 0$, the zero error condition. The straightforward scheme of storing one complete class, say C^1, requires a storage capacity of $\log_2\binom{m}{n} \approx n \log_2 m$ bits, while equation (3.5) claims a storage of $2n$ bits, which amounts to only 1 bit per sample. Indeed, the complete storage scheme is far from optimal, and schemes such as decision trees could achieve a perfect classification with a mean storage of only 1 bit per sample.

For a simple demonstration of such schemes consider n = 1, where two binary sequences each N bits long ($N = \log_2 m$) are presented with their (+) and (-) labels. At a later time an arbitrary binary N-sequence is presented for classification with the requirement that if the third sequence is identical to any of the original two, it must be classified correctly. Rather than storing one of the original samples (which requires N bits of storage), a more economical scheme can be employed: one need store only the index of the bit where the two samples first disagree, and the content of that bit. This requires at most only $1 + \log_2 N$ bits of storage. Moreover, if the two samples are arbitrary, the probability is quite high that disagreement would occur in the first few bits of the sequences leading (e.g., using Huffman Code) to a mean storage requirement of about three bits - slightly more than the two bits predicted by (3.5).

4. Order-Related QA Systems

Position-Matching Systems

Here we will consider systems which accept data-sets in the form of ordered lists of m distinct items, and respond to queries of the type: "What item stands in the k^{th} position?" In such systems the admissible queries can be represented by the set of integers from 1 to m, the answer-vocabulary $\underline{A}$ is the set of items $\{a_1, a_2, \cdots a_m\}$, and $\underline{M}$ is the set of permutations of the a's. This leads to: $M = m!$, $Q = m$, and $K = m$. Since every m-tuple from $\underline{A}^m$ can represent an answer-string, $\underline{A} = \underline{A}^m$, and $|\underline{A}| = m^m$. The distortion measure d_{ij} is given by the mean number of symbols in answer-string j which are not matched to their corresponding symbols in i. Clearly, a mean error of 1-1/m is achievable with no memory at all simply by producing an answer-string with one repeated symbol, thus, $D_{max} = 1-1/m$.

The rate-distortion function of this system can be shown to satisfy [9]:

$$R(D) \geq \log m! - m[H_b(D) + D \log(m-1)] \quad (4.1)$$

where

$$H_b(D) = - D \log D - (1-D) \log (1-D) \quad (4.2)$$

For large m one may use Stirling's approximation $\log m! \approx m \log m$, and obtain:

$$\frac{R(D)}{R(0)} \geq 1 - D - \frac{1}{\log m} H_b(D) \quad (4.3)$$

Thus, as m becomes very large the lower bound on R(D)/R(0) approaches the data-erasure line I-D at a logarithmic rate.

These results can be summarized by the following conclusions. *As the size of the data increases, the amount of storage needed to represent an ordered sequence of m items with a per-query error D is given by (l-D) m log m. Thus, no amount of error (below unity) oan make memory demands grow slower than m log* m, *and no data-reduction scheme exists which achieves a memory economy higher than that of data-erasure**

Note that the mathematical structure of position-matching systems remains essentially unaltered if $\underline{Q}$ is replaced by queries of the type, "Who is the immediate successor of a_1?" The only change required would be the inclusion of a null symbol ϕ 1n the answer vocabulary to represent the answer "no successor". With this modification we have $K = m+1$ instead of $K = m$, which for large m induces only a slight change in R(D). We thus conclude that successor-finding tasks are subject to the same basic memory vs. error limitations as those limitations found for position-matching tasks.

Pair-Ordering Systems

Let us consider a question-answering system which examines an ordered array of m items $(a_1, a_2, \cdots a_m)$ during the filing phase, and later responds to queries of the type, "Does a_i precede a_j?". The data-space $\underline{M}$ 1s again the set of permutations on $(a_1, a_2, \cdots a_m)$. The set of queries, however, Is homomorphlc to the set of all $\frac{m(m-1)}{2}$ pairs of elements, and the answer vocabulary $\underline{A}$ is (0, 1), with 1 and 0 standing for affirmation and denial of the query respectively. We assume that only *one of* the two q u e $a_i > a_j$, $a_j > a_i$ s in $\underline{Q}$. Thus, we have:

$$M = m!, \quad Q = \frac{m}{2}(m-1), \quad \underline{A} = (0, 1), \quad K = 2 \qquad (4.4)$$

An answer-string can be any binary Q-tuple, and so, $|\underline{A}| = 2^Q$. The distortion measure d_{ij}, between data-set i and answer-string j, is simply given by 1/Q times the number of pairs in i whose relative order conflicts with that stated by j. Clearly, if no information at all is retained in storage, any arbitrary answer-string would leave (on the average) 50% of the answers in error, so, $D_{max} = \frac{1}{2}$.

The lower bound on R(D), obtained [10] by the method leading to (4.1), gives:

$$R(D) \geq \log m! - \frac{m(m-1)}{2} H_b(D) \; . \qquad (4.5)$$

For large m this becomes:

$$\frac{R(D)}{R(0)} \geq 1 - \frac{m}{2 \log m} H_b(D) \qquad (4.6)$$

Thus, given any non-2ero D, there is always an m sufficiently large to make the bound zero. This can be attributed either to the fact that the system permits increasing amounts of memory savings, or simply to the looseness of the bound. We will later demonstrate that the former is true, by using an upper bound on R(D).

Note that the essential difference in the behavior of equations (4.1) and (4.5) comes from the increased size of the query set *Q.* The two are special cases of a general bound [8]:

$$R(D) \; M. \log M - Q \; H.(D) \qquad (4.7)$$

which holds for all question-answering systems with symmetric distortion measure (2.7). We may conclude that in order for R(D)/R(0) to vanish for all D > 0 it is necessary that Q increases (with m) faster than log M. This can be expressed by the general statement: *a necessary condition to enable a QA system to convert any small error tolerance into a slower growth of memory demand is that its query set be increasingly redundant.* Position-matching, for instance, has a query set with only a slight redundancy; one must know the position of m-1 items before the position of the remaining m^{tn} item can be deduced with certainty. In pair-ordering systems, on the other hand, knowing the correct precedence of only a small fraction (m-1) of selected pairs may be sufficient to deduce (by transitivity) precedence in all the remaining (m-l)(m-2)/2 pairs.

We now derive an upper bound on R(D) and demonstrate that a small error tolerance can reduce memory demands by a factor of log m. To that end we calculated the rate-distortion function R (D) of a restricted system where only transitivity-preserving answer-strings are allowed. This yields a parametric representation for $R_r(D)$ [10]:

$$QD = \frac{m\,z}{1-z} - \sum_{k=1}^{m} \frac{k\,z^k}{1-z^k} \qquad (4.8a)$$

$$R_r = \log m! + QD \log z + m \log(1-z) - \sum_{k=1}^{m} \log(1-z^k) \qquad (4.8b)$$

where z is the independent parameter varying between 0 and 1. For values of z not too close to 1 the summations in (4.8) converge rapidly, permitting numerical computations of $R_r(D)$ even for very large values of m. Equation (4.8) is depicted by the solid lines of Figure 4 for m * 100, 1000, and 10,000. It is seen that the (normalized) curves undergo a sharp drop in R near D = 0, and that this drop continues to grow with increasing m.

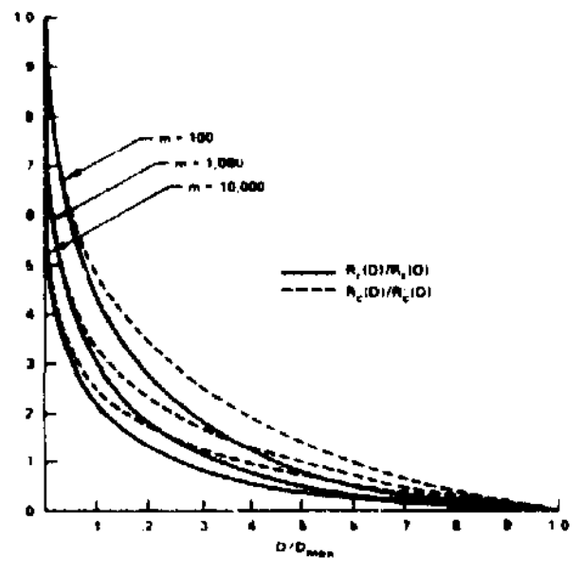

Figure 4. Rate Distortion Function for Pair-Ordering Task (solid lines), Compared with Memory vs Error Characteristics of Clustering Scheme (broken lines).

For large m, the asymptotic behavior of $R_r(d)$ is given by:

$$R_r(D) \simeq m \log 2/D \qquad (4.9)$$

Thus, *while the memory required to answer all queries correctly grows as m log m, only a linear increase with m is required when a small but finite error rate is tolerated.*

Recalling that Shannon's theorems guarantee only the existence of codes for very long data blocks, it remains to show that a memory reduction similar to (4.9) could also be achieved by schemes which encode each data-set separately. This can be demonstrated using the popular filing scheme known as "clustering". Assume that the sequence $(\alpha_1, \alpha_2, \cdots \alpha_m)$ represents the correct order of elements. Divide the sequence into ℓ groups each containing exactly m/ℓ consecutive elements, and retain in memory the name of each Item coupled with its group name. In the answering phase, adopt the following strategy: 1f α_i and o^ belong to two different groups, use the group label to decide precedence; if α_i and α_j belong to the same group, choose an arbitrary answer. It 1s easily shown that the memory vs. error characteristics of this scheme are governed by:

$$R_c(D) \simeq m(\log_2 \frac{1}{D} - 1) \qquad (4.10)$$

implying a linear memory growth similar to that predicted by the restricted rate-distortion function $R_r(D)$. The two are compared in Figure 4.

It is tempting to relate these results to fuzzy-sets representation [1] schemes. Here too, only a fixed number of linguistic labels such as *"very* small", "medium", "large", and "extremely large" are used to characterize linear orders of any size. Equation (4.10) indicates that for the purpose of pair-ordering the scheme is close to optimal. However, clustering methods employ sharp partitioning of the element-set into non-fuzzy subsets. It is not clear whether additional memory

gain could be achieved by the use of numerical indicators for 'the degree of set membership'. It is conceivable that the computational advantage of fuzzy-set coding schemes can only be demonstrated in more complex data structures such as those containing relations among several orders.

5. Conclusions and Relations to Computational Complexity

Four question-answering tasks were analyzed to test the conjecture that significant memory savings could be materialized with only a slight tolerance for errors. It was shown that the tasks of data-retrieval and pattern-classification, which in the past have absorbed considerable efforts toward economy of descriptions, are subject to a basic limitation. No data-reduction scheme exists which exploits error-allowances in an appreciably more effective way than a straightforward data-erasure. Position-matching, namely, the task of representing a linear order for the purpose of rank identification (or successor identification), is subject to a similar limitation. In applications where the basic decision unit served by the order is the determination of precedences on pairs of elements, a more substantial reduction in memory size can be achieved when a small but finite error is permitted. This difference in behavior is attributable to the fact that in pair-ordering applications the elementary decision tasks are highly interdependent (via transitivity), whereas those in position-matching applications are almost independent.

In a more recent work [3] it was found that the results reported in this paper go beyond our original intent of limiting the storage-error exchange. The work demonstrates that many complexity measures on a variety of computation models can be effectively bounded using the rate-distortion function. Given a computational task (e.g., sorting, function computation, sequence generation), a class of algorithms for accomplishing that task (e.g., straight line algorithms, finite state machines) and a complexity measure on the class (e.g., program length, time or space complexity, combinational complexity), there exists a characteristic function C[R(D)] which provides a lower bound on the mean complexity necessary to produce a mean distortion not exceeding D. Moreover, in almost all interesting cases the function approaches a straight line C[R(D)] * *a* R(0) as the task size becomes very large. This implies that the relative reduction in the mean complexity induced by tolerating a mean distortion D cannot exceed the relative change of the rate-distortion function R(D).

Typical results obtained by this method are:

1. The mean number of binary comparisons required for sorting N-sequences is bounded by

$$C(D_s) > N \log N (1 - D_s) \quad , \quad N \text{ - -->} \quad (5.1)$$

where $D_\$$ is the fraction of terms in the output sequence which are followed by wrong successors. It implies that no sorting scheme exists which matches a fixed percentage of the term with their correct successors and which requires less than 0(N log N) comparisons.

2. If D is the fraction of pairs in the output sequence Which are out of order, then the mean number of comparisons is bounded by

$$\overline{C}(D_p) \underset{N \to \infty}{\geq} \begin{cases} N \log N & D_p = 0 \\ N \log 2/D_p & D_p > 0 \end{cases} \quad (5.2)$$

Equation (5.2) allows for the possibility of sorting N sequences in linear time if one permits any finite fraction of pairs to remain out of order. Such a sorting scheme indeed exists using QUICKSORT with a predetermined number of iterations.

3. The mean combinational complexity of Boolean functions in N variables is bounded by

$$\overline{C}_B(D) \geq \frac{2^N}{N} H_b(D) \quad , \quad N \to \infty \quad (5.3)$$

where D is the probability of output errors.

4. The average number of states over the ensemble of minimal-state sequential machines needed to reproduce binary N-sequences with probability of errors not exceeding D is bounded by:

$$\overline{C}_s(D) \geq (N - \log N) H_b(D) \quad (5.4)$$

It is conjectured that the relative insensitivity of R(D) to errors of the type treated in this paper will also limit the complexity vs. error exchange of language-recognition and theorem-proving. It appears that if inexactness could permit drastic computational savings the effect may surface only in cases where the data is highly structured and/or the queries are highly redundant.

Acknowledgment

This work was supported by the National Science Foundation under Grant No. GJ 42732.

References

1. Lotfi A. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," *IEEE Trans act-Lone on Systems, Mzn, and Cybernetics*, Vol. SMC-3, No. 1, Jan. 1973.
2. Michael 0. Rabin, "Theoretical Impediments to Artificial Intelligence," *Proceedings of IFIP Congress I9?4y* August 5-10, 1974, Stockholm, Sweden.
3. J. Pearl, "On the Complexity of Inexact Computations ," *UCLA-ENG-PAPER-0775,* May, 1975.
4. M. Minsky, and S. Papert, Perceptrons, Cambridge, Massachusetts and London, The M.I.T. Press, 1969 (Chapter 12).
5. C. E. Shannon, "Coding Theorems for a Discrete Source with a Fidelity Criterion," *1959 IRE National Convention Record** Pt 4, pp. 142-163.
6. T. Berger, Rate Distortion Theory, Englewood Cliffs, New Jersey: Prentice-Hall, 1971.
7. Ranan B. Banerji, "Some Linguistic and Statistical Problems in Pattern Recognition," Pattern Recognition, Vol. 3, p. 409-419, Pergamon Press", 19*71.
8. J. Pearl, "Error-Bounds for Inferential Question-Answering Systems with Limited Memory," *UCLA-ENG-748U* October, 1974.
9. J. Pearl, "An Application of Rate-Distortion Theory to Pattern Recognition and Classification," *UCLA-ENG-7501,* January, 1975.
10. J. Pearl, "Memory vs. Error Characteristics for Inexact Representations of Linear Orders" *UCLA-ENG-7491,* December, 1974.