An Autonomous Reading Machine

RICHARD G. CASEY AND GEORGE NAGY, MEMBER, IEEE

***Abstract*—An unconventional approach to character recognition is developed. The resulting system is based solely on the statistical properties of the language, therefore it can read printed text with no previous training or a priori information about the structure of the characters. The known letter-pair frequencies of the language are used to identify the printed symbols in the following manner.**

First, the scanned characters are partitioned into distinct groups of similar patterns by means of a distance measure. Each class (at most 26 are permitted) is assigned an arbitrary label, and an intermediate tape, containing these temporary labels of the symbols in the original sequence, is generated.

In the second phase of the program, the matrix of bigram frequencies of the labels is compared to a frequency matrix obtained from a large sample of English text. The labels are then assigned alphabetic symbols in such a way that the correspondence between the two matrices is maximized.

The method is tested on a 100 000-character data set comprising four markedly different fonts.

***Index Terms*—Adaptive, character recognition, clustering, cryptograms, linear categorizer, pattern classification, reading machine.**

Introduction

THE SYSTEM to be described presents a striking contrast to the usual methods of character recognition. The processor is exposed to an unlabeled representation of each character in a passage of printed text, but it is given no information regarding the significance of the various elements in these patterns. Unlike conventional classifiers, it is never "trained" on identified samples. In principle, this processor has no stored data to aid it except contextual information in the form of a table of letter-pair frequencies derived from other samples of text.

In a sample of text several thousand words long, the observed letter transition frequencies may be expected to match the stored values quite closely. For printed text in such quantity, it seems quite reasonable to base recognition on the relatively invariant transition frequencies, rather than on preconceptions of the structure of the characters.

This idea is applied to reading text in a manner similar to the solution of a simple substitution cryptogram puzzle. Tentative identities are assigned to groups of similar characters. These identities are then permuted until the text reads sensibly; good sense in this case means an admissible set of transition frequencies. Letter-pair probabilities are actually used by way of compromise between singlet frequencies, which are insufficient on a sample of only a few thousand words to identify any but the most common letters (e, t, a, o), and longer sequences, which require a prohibitive amount of computation.

Manuscript received February 13, 1967; revised December 5, 1967.

The authors are with the IBM Watson Research Center, Yorktown Heights, N. Y. G. Nagy is currently on leave at the University of Montreal, Montreal, Quebec, Canada.

In solving a cryptogram, each type of character is labeled with the same (though incorrect) symbol to start with. The processor described here, however, must first perform the nontrivial task of determining which samples should be classed together by the same symbol. The system achieves this by means of a series of cluster-seeking algorithms operating in the sample space.

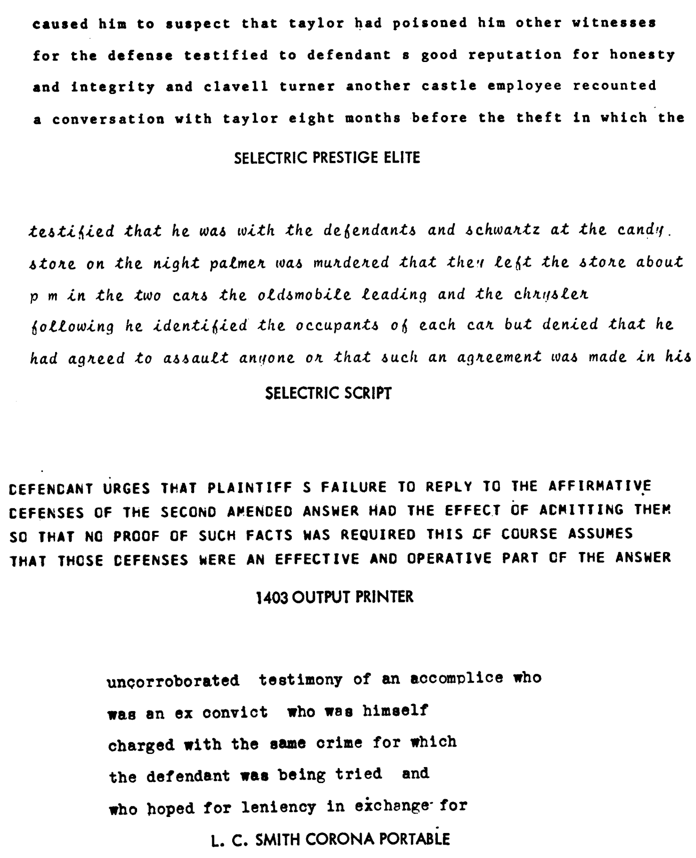
Each point in the sample space is a 96-dimensional binary vector obtained by applying quasi-random n-tuples in all "shifts" to the binary version of a scanned character. This device reduces the dimensionality of the sample space (from 360 to 96) and provides registration invariance without eliminating the distinction between classes.

There are very few restrictions as to the source of the printed material. Whereas systems whose parameters are determined on identified samples suffer when a previously unseen font style is introduced, this categorizer relies only on certain properties of consistency within the particular portion of the text to be read. The "geometric" properties required of the input data can be roughly stated as: 1) samples of the same alphabetic category should be similar, and 2) samples of different categories should be dissimilar.

The two phases of the recognition procedure, clustering of similar characters and assignment of identities, have both been implemented by essentially the same technique: minimization of a distance function. In the cluster-seeking phase, the mean-square distance from the samples to the cluster centers is minimized. To decipher the cryptogram, letter identities are assigned to the clusters so as to maximize the statistical likelihood of the corresponding letter-pair frequencies in the text. A detailed description of these procedures is given in Sections II and III.

The performance of the system is evaluated on 20 000–30 000 character segments of text in a single type-case in each of four dissimilar fonts (Fig. 1). A brief description of the experimental system, a table of the overall error rates, and a detailed illustration of the action of the algorithm on one particular font style form the contents of Section IV.

The text was scanned on the experimental print reader of the IBM Watson Research Center. The clus-


caused him to suspect that taylor had poisoned him other witnesses
for the defense testified to defendant s good reputation for honesty
and integrity and clavell turner another castle employee recounted
a conversation with taylor eight months before the theft in which the

SELECTRIC PRESTIGE ELITE

testified that he was with the defendants and schwartz at the candy.
store on the night palmer was murdered that they left the store about
p m in the two cars the oldsmobile leading and the chrysler
following he identified the occupants of each car but denied that he
had agreed to assault anyone or that such an agreement was made in his

SELECTRIC SCRIPT

CEFENCANT URGES THAT PLAINTIFF S FAILURE TO REPLY TO THE AFFIRMATIVE
CEFENSES OF THE SECOND AMENDED ANSWER HAD THE EFFECT OF ACMITTING THEM
SO THAT NO PROOF OF SUCH FACTS WAS REQUIRED THIS CF COURSE ASSUMES
THAT THOSE CEFENSES WERE AN EFFECTIVE AND OPERATIVE PART CF THE ANSWER

1403 OUTPUT PRINTER

uncorroborated testimony of an accomplice who
was an ex convict who was himself
charged with the same crime for which
the defendant was being tried and
who hoped for leniency in exchange for

L. C. SMITH CORONA PORTABLE

Fig. 1. Samples of text used as input. The 1403 chain printer tends to lighten the right-hand riser of several characters; D-O-C confusions are common. The Smith-Corona typewriter, a vintage model (circa 1920), produces almost closed-in c's; hence c-o is a problem here too.

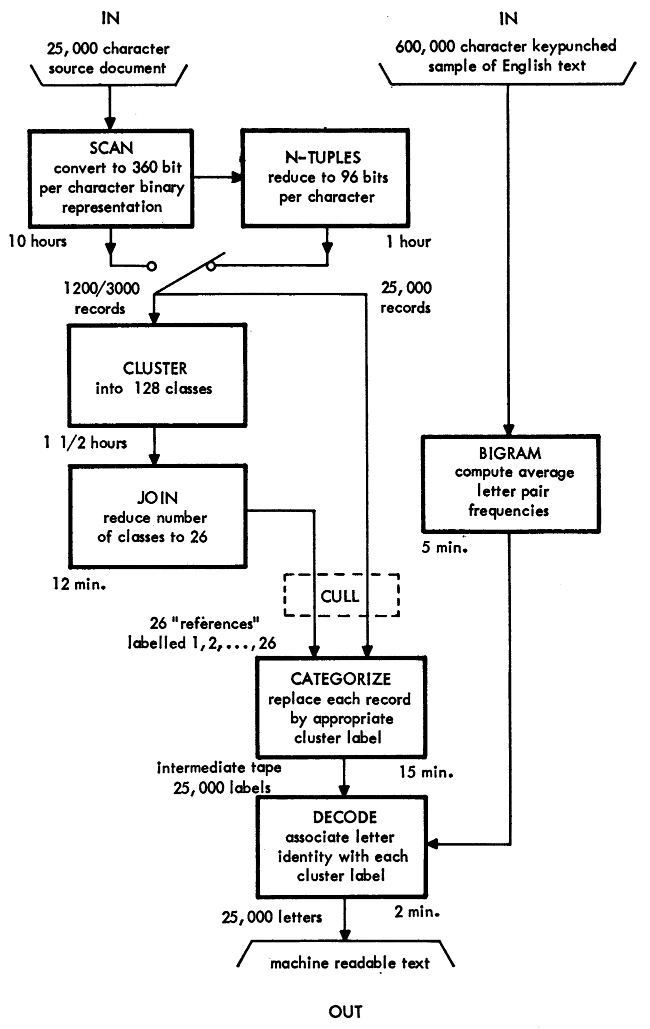

IN
25,000 character source document
IN
600,000 character keypunched sample of English text
SCAN convert to 360 bit per character binary representation
N-TUPLES reduce to 96 bits per character
10 hours
1 hour
1200/3000 records
25,000 records
CLUSTER into 128 classes
1 1/2 hours
BIGRAM compute average letter pair frequencies
JOIN reduce number of classes to 26
5 min.
12 min.
CULL
26 "references" labelled 1,2,...,26
CATEGORIZE replace each record by appropriate cluster label
intermediate tape 25,000 labels
15 min.
DECODE associate letter identity with each cluster label
25,000 letters
2 min.
machine readable text
OUT

Fig. 2. Flow diagram of programs. All but the SCAN and N-TUPLES programs, which use the 1400 computer-scanner system, run on an IBM 7094 II computer. The running times shown refer to processing "features." The dotted CULL block makes use of the CATEGORIZE and JOIN programs to derive references for rare characters.

tering, recognition, and cryptogram decoding operations were performed by an IBM 7094 at the same location. An overall flow diagram, including processing times for the various operations, is given in Fig. 2.

I. Related Work

Rabinow[22] has recently obtained a patent covering the general concept of an autonomous reading machine. He envisages combining an elementary clustering procedure with dictionary look-up techniques. To our knowledge no theoretical or experimental results have been reported so far which would allow a comparison of Rabinow's system with that described here.

An excellent survey of cluster-seeking methods, including an extensive bibliography, has been prepared by Ball.[2] Among the techniques reported, the ISODATA distance-minimizing algorithm of Ball and Hall[3] is of particular interest here, since a modified version of it was used as the basic tool for grouping similar character samples in the experiments to be described.

Abraham[1] has considered the problem of finding clusters of related samples given a similarity matrix whose ijth element is 1 if samples i and j are similar, and 0 if they are not. The similarity matrix can be pictured as an undirected graph with nodes representing the samples; similarity of two samples is expressed by a connection between the corresponding nodes. Each of the disjoint subgraphs of such a graph defines a cluster of related samples.

Bonner[5] has also applied cluster-seeking techniques to character recognition. The objective of his work was to design Boolean logic for reading magnetic characters. As sets of similar characters are defined by means of his cluster-seeking program, logical bit combinations are specified to recognize each cluster. Here clustering is an aid to a conventional design program rather than a step in the recognition procedure itself.

An interesting discussion on the use of cluster-seeking methods to organize data is presented by Ornstein.[18] Ornstein proposes to define a taxonomic structure on given samples with a confidence measure indicating the reliability of classification at each level. This operation is considered primarily as a step in the process of inductive reasoning. Clustering serves the added function of distinguishing between the essential and inessential attributes of the samples. In the present work, also, the aim is to find the "natural" clusters in the input. A taxonomic structure is not necessary, however, since it is known how many clusters exist.

Methods of sorting data so as to minimize several rather sophisticated statistical measures have been devised by Friedman and Rubin.[11] One of their objectives is to take the correlation within groups into account during the partitioning operation. Conceivably these high-powered procedures would be capable of resolving even coarctate clusters.

Algorithms developed for cluster seeking are usually justified on intuitive grounds. Cooper and Cooper,[9] on the other hand, have examined analytically the problem of partitioning a sample space into subpopulations. With their approach the problem is resolved into one of estimating the statistical parameters of a composite population. Results thus far have been limited to the most elementary cases, e.g., two spherical normal distributions.

One of the striking features of the clustering literature is that many of the experiments reported are based on medical diagnostics,[6] botanical measurements,[23] etc., where it is difficult to evaluate or even compare the various techniques since the desired clustering performance on such data is usually not known. In the present work, a clearly defined objective is postulated, namely, that all the samples of each alphabetic category should constitute a distinct cluster. Thus, it would seem that character recognition provides an excellent framework for testing cluster-seeking methods.

For the cryptoanalytic portion of the work several standard references have been consulted,[12],[19],[21] but none of the methods suggested seemed sufficiently general and insensitive to categorization errors. In addition, the detailed nature of the customary procedures renders them tedious to program and difficult to formulate analytically. If others have tried permuting indices to minimize a distance function, the authors are not aware of the pertinent studies.

II. Determination of Subgroups

There is no precise, completely general definition of a "cluster." In intuitive terms a cluster is a subset of samples which are somehow related to one another (e.g., similar) but are not related to the remaining samples. Perhaps the simplest way to formalize this notion is to define a two-state similarity measure by means of which any two samples can be labeled either "similar" or "not similar." The clusters are then identified with the disjoint subgraphs of the graph associated with the similarity matrix for the given samples. The key to this method is the accurate assignment of similarity states.

Unfortunately, in the problem under consideration a similarity matrix is not available a priori, nor can one be readily inferred from the distribution of the samples. It is, in fact, unwise to attempt to use the similarity matrix on individual samples; the individual decisions are so critical that changing the similarity relation between only a few pairs of samples may completely alter the cluster configuration. A more stable approach is to form only the most clear-cut similarity groupings at first, and to work thereafter with averages of the grouped samples in attempting to determine additional similarity relations. This is the strategy adopted here.

Since the cluster seeking must deal with real-world data, it is designed to take in stride possible scanner failures; additional restrictions are imposed by computer running time and internal storage limitations. In other respects the system is designed to be as free of arbitrary assumptions as possible.

The cluster-seeking program operates on only a portion of the text to be read. The sample population selected consists of 3000 96-bit feature vectors, the largest amount of data which can be comfortably stored in an IBM 7094 computer core memory along with the program itself. Each of the 96 bits in a sample vector denotes the presence or absence of a particular feature in the character represented.

The first step in the process (see Fig. 3) is to partition these samples into a large number of subsets of the most similar vectors. This partitioning, based on a distance measure of similarity, divides the samples of each identity into a number of subsets. Some errors—samples of one identity assigned to a subset largely occupied by another identity—may occur. These are sufficiently infrequent, however, to allow further operations to be conducted on each subset as a whole (represented, for example, by its centroid), rather than on the individual samples.

Next, the closest pairs of subgroups are merged. This step is iterated until a sudden increase in the minimum pairwise distance between subgroups indicates that two different identities are about to be joined. At this point there may be more than 26 clusters remaining. The larger ones are likely to consist of the most common letters occurring in the text. The smaller clusters are either rare characters (x's, z's, and q's) or stray samples which cannot be definitely assigned to any of the larger clusters. Since little intelligibility is lost if the former are misidentified, the processor simply selects the 26 largest clusters to represent the alphabetic identities.

The 26 selected clusters are used to sort the remainder of the text characters by means of a linear categorization procedure. At the conclusion of this stage each sample in the whole text has been assigned to a unique cluster number from 1 to 26. The data are then in stan-

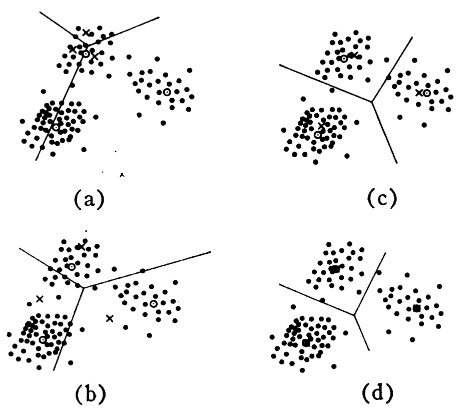


Fig. 3. Clustering to minimize the mean-square distance. The solid dots represent the position of the samples in measurement space. The circled dots indicate the centroids of the clusters (which are fixed), the x's are the variable cluster centers, and the solid lines represent the partitioning boundaries. At each step the cluster center moves to the center of gravity of the corresponding partitioned samples. This defines a new linear boundary halfway between each pair of cluster centers.

dard cryptogram form, ready to be decoded into alphabetic symbols.

The number of similarity classes formed in the initial stage is a compromise between two opposing desiderata. A large number of similarity classes decreases the likelihood that the clusters will be tainted by a mixture of identities. No further partitioning is done after this stage; hence a cluster containing more than one identity is bound to contribute erroneous classifications. On the other hand forming fewer clusters facilitates subsequent operations, in which each cluster is treated as an entity. In the experiments reported here, 128 initial clusters were formed. This number permitted reasonable speed (1–2 hours processing time on a 7094 computer), and also conformed with storage allowances in the follow-up program for combining the initial clusters.

Clustering by Minimization of Mean-Square Distance

The basic method used for partitioning the 3000 binary samples is the following. Let the m samples be represented as n-component vectors $S_1, S_2, \cdots, S_m$. These samples are partitioned into r subsets by associating each S_i with the closest (in Cartesian distance) member of a set of vectors $X_1, X_2, \cdots, X_r$.

The placement of these "cluster centers" determines the partitioning achieved. A useful criterion, adopted here for the X's, is that they minimize

$$d^2 = \frac{1}{m} \sum_{i=1}^{m} | S_i - X_j(_i) |^2$$

where $j(i)$ is the index of the cluster center closest to the ith sample.

The quantity d^2 is the mean-square distance from a sample to its corresponding cluster center. It can also be

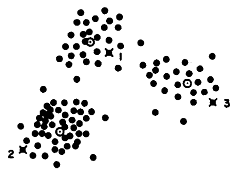


Fig. 4. Initializing procedure. The first cluster center is chosen arbitrarily; the second is the sample furthest from the first. Succeeding samples are chosen to maximize the minimum distance to the existing cluster centers. The sample marked by x and labeled "1" constitutes the first cluster center, "2" the second, and "3" the third. The circled dots indicate the centers of gravity of the clusters.

interpreted as the mean-square error resulting if each S_i is replaced by $X_{j(i)}$.[24] Its primary value in cluster seeking is as an overall measure of the tightness of a set of clusters.

An algorithm for obtaining a local minimum of d^2 is illustrated in Fig. 3 for a two-dimensional example. The sample set, consisting of three well-defined populations, is to be resolved into three clusters.

The first step of the algorithm is to choose three initial cluster centers $X_{1,0}$, $X_{2,0}$, $X_{3,0}$. The samples are then partitioned into three subsets by assigning each sample to the nearest cluster center, as shown in Fig. 3(a). The center of gravity of each subset is chosen as the next cluster center, and a new partitioning is obtained by the minimum-distance rule [Fig. 3(b)]. In this example, the procedure converges after four iterations.

The minimization and convergence properties of this algorithm are discussed elsewhere.[3],[8],[15] In order to cluster binary data efficiently, the X's are constrained to be ternary vectors. The restriction to ternary values lends additional speed and conserves storage with little sacrifice in accuracy. In an extreme case the mean-square error would be 33 percent greater than the value attained with infinite precision in the X's, but for the purpose of partitioning the samples into tight clusters, the lower quantization is adequate.

Initial Cluster Centers

The mean-square distance achieved by the algorithm, as well as the speed of convergence, depends on the location of the starting cluster centers. In general, it is desirable that the X's be distributed over the populated region of the sample space rather than concentrated in one part of it as in Fig. 3(a). The procedure used to obtain such a distribution of cluster centers is the following (see Fig. 4). The first sample in the batch to be processed is designated cluster center number one. The distances of the remaining samples from this one are calculated, and the farthest sample is called center number two. The smaller of the two distances from each sample to these two centers is listed, and the sample having the greatest minimum distance is selected. The remaining centers are chosen in turn to have maximum separation from the existing centers. These initial cluster centers are well scattered over the sample space, an intuitively desirable property. In the example of Fig. 3, this initializing procedure would have led to convergence after at most three steps instead of four.

Sequential Clustering

Applying the preceding algorithm directly to sort 3000 characters into 128 clusters has several disadvantages. First of all, it is time consuming: the (128×3000) distances from samples to cluster centers must be computed at each of the 10–20 iterations required to reach convergence. Secondly, it does not give the best performance in typical cases.

Related schemes which have also been experimentally investigated are the following. The direct cluster-seeking procedure is used to partition the given samples into 2^k clusters, where k may be any integer in the range 1 to 7. Each of these clusters is split in turn into two subclusters by reapplying the distance-minimizing algorithm. This step gives 2^{k+1} subclusters which can each be split into two parts in the same manner again and again until $2^7=128$ clusters are obtained.

Fig. 5 shows how these methods compare on a particular batch of data. The value $k=7$ corresponds to the direct procedure of forming 128 clusters in a single stage. The quantity plotted is the sum of the number of minority characters in each of the 128 clusters. These results are not conclusive, but they do serve at least to justify the more economical scheme, finally adopted, of clustering the data into 32 groups to begin with, then successively splitting each subgroup two times. This procedure was about twice as fast as the direct $(k=7)$ method.

At $k=4$ and below, the groups contain a heavy mixture of identities after the initial clustering. Splitting into pairs tends to split the identities, especially when more than two identities are present in a single cluster. Thus, sorting directly into only 32 subgroups gave 60 misclustered characters, while starting with 2 clusters and splitting in six more steps into 128 gave 254 misclustered samples. It is evidently better to start with a number which is on the order of the "natural" number of subgroups in the data than to start with one which is significantly smaller. This conclusion runs contrary to generally accepted results on hierarchically structured data.[11],[24]

Cluster Joining

The 128 clusters consist of collections of adjacent sample points distributed over the measurement space. A logical approach to combining clusters is to define a "distance" between collections of samples, and to join those which are "closest." One such distance measure is the amount of separation between cluster centers. A more sophisticated function of the structure of the clusters, e.g., one which allows for the size and shape of the clusters, would perhaps allow clusters to be combined with more confidence. If the various categories are well separated to begin with, the distance between means forms a sufficient basis for this operation.

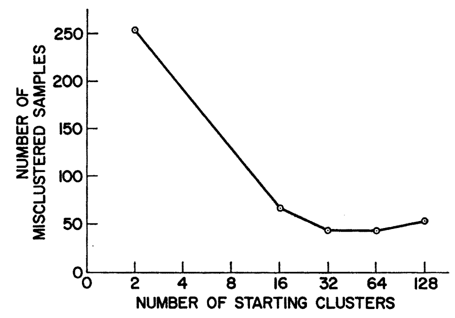


Fig. 5. Clustering performance for a sequential splitting procedure. The sharp initial decrease occurs because the cluster-seeking algorithm will often correctly sort mixed samples of two identities, but is unable to find a good dichotomy for samples of many categories.

The procedure is implemented as follows. The distances between the mean vectors of all pairs of clusters are listed in a table. The pair having the lowest separation is merged into a single cluster, and a new mean vector is calculated. The pairwise distance table is updated to eliminate entries corresponding to the joined clusters, and to enter the distance from the combined cluster to each of the others. The new minimum entry is located, the corresponding clusters are joined, and the process is allowed to iterate in this manner.

Since there are 26 alphabetic categories it might seem proper to continue joining until only 26 clusters remain. There are two reasons why this must not be done. On the one hand, the rarer classes (x, q, z) may not occur in the 3000 (or fewer) samples. On the other, the sample population often contains poorly scanned, improplerly segmented, or otherwise distorted characters which tend to precipitate into clusters. These specimens are usually quite dissimilar from samples of their own or of any of the other categories; thus, on the basis of distance, a deleterious joining of similar character types (such as O and C) will occur before these stray clusters are combined with others.

A fixed distance threshold for joining is an alternative criterion for stopping. However, the appropriate threshold varies as a function of the font in which the text is printed. Better results are obtained when the machine itself determines a threshold on the basis of the cluster distribution, by keeping track of the distance between clusters to be combined, i.e., the minimum distance in the table at each step. If the various categories are indeed well separated, the minimum entry in the table will begin to increase rapidly at some point in the joining routine. Fig. 6 shows the distribution of these values as the clusters are successively combined almost to the limit of a single collection of 3000 samples. The appropriate threshold for terminating is clearly indicated by the minimum in the distribution. Furthermore, the choice for the threshold is less critical than it might

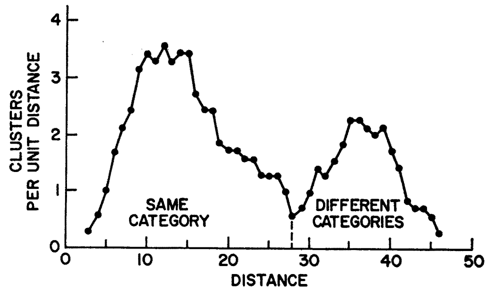


Fig. 6. Determination of the threshold for joining clusters. This plot (illustrating script font) has been smoothed by averaging locally over seven values of the independent variable. As expected, clusters of samples of the same category are close together and join at a low distance threshold, while clusters corresponding to different identities join at a very high threshold. The existence of a dip separating the two regions enables the system to select the proper joining threshold without knowledge of the identities.

appear since only very small clusters are being combined in the neighborhood of the indicated value.

The presence of poor sample representatives, as implied above, leads to serious problems in clustering character data. For example, when such samples are present, the corresponding clusters will be impossible to distinguish from "q" or "x" or "z" clusters, which are equally small in a total population of 3000 characters or less. In addition, a "chain" effect may occur wherein a stray cluster joined to a valid cluster shifts the ensemble cluster center in the direction of a third cluster of a different identity, so that all three are eventually combined.

Such difficulties can be overcome if the characteristics of stray samples are known. Multilated characters, for example, excite very few of the n-tuple masks used for dimension reduction in the experiments. These samples were eliminated prior to joining by deleting clusters whose centers had fewer than five 1 bits. In addition, clusters of high internal variance are suspect. Often these turn out to be composed of several stray characters of different identities. A threshold on the allowable internal variance, as measured by the number of *don't cares* in the ternary cluster center, is effective in eliminating such samples, but was not included in the experiments reported here.

Final Assignment of Cluster Numbers

After completion of the preceding stages, 3000 samples of text have been partitioned into 25–35 clusters. The most populous clusters correspond to letter identities; the sparser ones may still contain either the infrequent letters or stray samples. Since only 26 character types can be presented to the cryptogram stage, the number of clusters is reduced on the basis of size when necessary. Although selecting the 26 largest clusters is liable to result in the loss of some categories, the overall system error rate is not greatly increased by this method.

The 26 remaining cluster centers are used as the decision parameters of a minimum-distance categorizer. The entire text, including the already clustered data, is submitted to this categorizer in order to relabel samples which were weeded out during clustering. Each character is tagged with the number of the closest cluster center, and the numbers are recorded in sequence on an intermediate magnetic tape The minimum-distance classifying technique is in accord with the cluster-seeking method in which distance to a fixed point was the criterion for assigning a sample to a cluster.

III. Identification of the Classes

The class-pair frequencies observed on a 25 000-character sample of unidentified text, and the letter-pair frequencies derived from a 600 000-character sample of identified text, are shown in Fig. 7. What is required is a method of associating the symbols of the alphabet with the numbered classes in such a way as to maximize the resemblance between the two matrices. A precise definition of "resemblance" will be helpful in this endeavor.

The Bigram Distance Function

The observed class-pair frequencies are

$$f_{ij} = \frac{N_{ij}}{N} \qquad i = 1, 2, \cdots, 27; j = 1, 2, \cdots, 27,$$

where N_{ij} is the number of occurrences of a pattern of class i followed by one of class j, and $N-1$ is the total number of patterns, including blanks. The f_{ij}'s will be regarded as independently distributed random variables, despite the slight dependence introduced by the constraint

$$\sum_{i,j} f_{ij} = 1.$$

It is also known that the class-pair frequency corresponding to the alphabetic symbols (k, l), will have the binomial density distribution

$$P\{f_{kl}\} = \binom{N}{Nf_{kl}} p_{kl}{}^{Nf_{kl}}(1 - p_{kl})^{N-Nf_{kl}}$$

where p_{kl} is the frequency of occurrence of the letter-pair (k, l) in a very large sample of the English language.

For large N, and for values of f_{kl} reasonably close to the mean, the above distribution may be approximated by the normal density function

$$P\{f_{kl}\} \simeq [2\pi N p_{kl}(1 - p_{kl})]^{-1/2} \cdot \exp\left[-1/2 \frac{(f_{kl} - p_{kl})^2}{N p_{kl}(1 - p_{kl})}\right].$$

Now consider the one-to-one mappings, $T_s(i) = k$, $s = 1, 2, \cdots, 26,$[1] which assigns a symbol of the alpha-

[1] Note that while bigrams involving blanks are used to facilitate the association of letters and classes, blanks are unequivocally identified by the scanner, and thus do not appear in the mapping proper.

PROBABLE BIGRAM FREQUENCIES X 10

		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	.693	.044	.001	.015	.178	.375	.099	.044	.038	.003	.003	.008	.062	.018	.176	.074	.008	.000	.097	.213	.180	.002	.011	.013	.005	.096	.000
A	.207	.000	.004	.030	.011	.032	.011	.010	.057	.019	.001	.000	.038	.031	.020	.008	.020	.000	.038	.013	.049	.006	.007	.022	.003	.000	.000
B	.079	.011	.000	.000	.000	.001	.000	.000	.001	.005	.000	.000	.000	.002	.000	.003	.000	.000	.001	.000	.000	.008	.000	.000	.000	.000	.000
C	.124	.028	.000	.004	.000	.044	.000	.000	.000	.047	.000	.000	.000	.001	.031	.010	.000	.000	.009	.005	.001	.014	.000	.000	.005	.000	.000
D	.057	.017	.000	.000	.001	.085	.000	.000	.000	.021	.000	.000	.015	.000	.090	.006	.000	.000	.025	.001	.000	.008	.000	.000	.000	.000	.000
E	.057	.000	.033	.053	.063	.025	.016	.021	.216	.022	.003	.008	.045	.039	.037	.003	.043	.000	.124	.074	.085	.012	.037	.020	.003	.003	.002
F	.064	.006	.000	.000	.000	.017	.017	.000	.000	.014	.000	.000	.002	.000	.003	.095	.000	.000	.002	.000	.000	.002	.000	.001	.000	.000	.000
G	.016	.011	.000	.000	.006	.012	.000	.001	.000	.014	.000	.000	.001	.000	.052	.003	.000	.000	.004	.000	.000	.005	.000	.000	.000	.000	.000
H	.047	.001	.000	.032	.000	.003	.000	.013	.000	.000	.000	.000	.000	.000	.001	.005	.003	.000	.001	.016	.258	.000	.000	.022	.001	.000	.000
I	.130	.025	.006	.026	.037	.010	.026	.007	.046	.000	.000	.002	.027	.024	.017	.004	.004	.000	.047	.033	.096	.010	.024	.019	.002	.001	.000
J	.016	.001	.002	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.003	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
K	.004	.004	.000	.004	.000	.001	.000	.000	.000	.001	.000	.000	.001	.000	.002	.001	.000	.000	.006	.001	.000	.000	.000	.000	.000	.000	.000
L	.026	.067	.010	.013	.001	.031	.003	.002	.001	.025	.000	.001	.042	.000	.004	.017	.017	.000	.003	.003	.003	.017	.000	.001	.000	.001	.000
M	.044	.011	.001	.000	.002	.016	.000	.003	.000	.017	.000	.000	.000	.011	.001	.028	.001	.000	.011	.002	.003	.005	.000	.003	.000	.001	.000
N	.047	.118	.000	.000	.000	.091	.000	.003	.001	.168	.000	.002	.000	.000	.005	.143	.000	.000	.008	.000	.002	.023	.000	.004	.000	.000	.000
O	.153	.000	.016	.084	.008	.003	.033	.004	.025	.076	.003	.000	.014	.015	.032	.005	.027	.000	.043	.016	.067	.001	.003	.006	.000	.002	.003
P	.075	.029	.000	.000	.000	.010	.000	.000	.000	.007	.000	.000	.000	.012	.000	.013	.028	.000	.007	.008	.000	.013	.000	.000	.002	.000	.000
Q	.004	.000	.000	.000	.000	.005	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
R	.057	.063	.005	.006	.003	.140	.009	.008	.003	.017	.000	.000	.000	.000	.001	.094	.039	.000	.008	.000	.028	.043	.000	.003	.000	.000	.000
S	.116	.060	.003	.001	.005	.075	.000	.004	.001	.081	.000	.001	.007	.004	.035	.013	.001	.000	.023	.029	.025	.032	.000	.003	.000	.005	.000
T	.307	.099	.001	.043	.000	.022	.004	.001	.008	.078	.000	.000	.005	.000	.087	.032	.007	.000	.043	.072	.011	.024	.000	.000	.002	.000	.000
U	.024	.010	.011	.010	.006	.000	.006	.005	.003	.000	.013	.000	.006	.004	.004	.051	.006	.010	.011	.032	.013	.000	.000	.000	.000	.000	.000
V	.023	.008	.000	.000	.001	.018	.000	.000	.000	.010	.000	.000	.002	.000	.003	.011	.000	.000	.005	.000	.000	.000	.000	.000	.000	.000	.000
W	.075	.007	.000	.000	.001	.007	.000	.000	.001	.000	.000	.000	.001	.000	.002	.018	.000	.000	.001	.001	.004	.000	.000	.000	.000	.000	.000
X	.000	.005	.000	.000	.000	.015	.000	.000	.000	.001	.000	.000	.000	.000	.000	.001	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
Y	.005	.011	.019	.002	.001	.005	.001	.000	.002	.000	.000	.000	.020	.000	.009	.001	.000	.000	.014	.001	.019	.000	.000	.000	.000	.000	.000
Z	.003	.000	.000	.000	.000	.000	.000	.000	.000	.002	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

OBSERVED BIGRAM FREQUENCIES X 10

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1	.001	.006	.000	.215	.064	.009	.146	.191	.001	.054	.008	.005	.008	.035	.002	.012	.044	.024	.077	.044	.014	.105	.043	.213	.001	.364	.002
2	.112	.000	.009	.034	.002	.001	.016	.082	.001	.009	.020	.019	.010	.003	.001	.009	.026	.027	.016	.030	.017	.031	.031	.033	.011	.013	.002
3	.079	.002	.001	.000	.002	.000	.001	.000	.000	.011	.000	.000	.000	.001	.000	.000	.000	.001	.000	.000	.001	.001	.017	.001	.005	.002	.001
4	.049	.024	.000	.001	.000	.000	.118	.000	.001	.011	.000	.001	.000	.000	.001	.000	.015	.001	.001	.002	.000	.013	.031	.002	.009	.101	.001
5	.002	.000	.011	.002	.000	.000	.008	.010	.000	.005	.000	.002	.000	.000	.000	.000	.011	.000	.000	.001	.000	.019	.005	.001	.000	.009	.000
6	.000	.000	.002	.001	.000	.000	.000	.002	.000	.000	.000	.000	.000	.000	.000	.000	.002	.000	.000	.000	.000	.003	.001	.000	.000	.002	.000
7	.038	.143	.000	.001	.000	.000	.003	.007	.001	.133	.005	.001	.000	.002	.001	.001	.001	.000	.001	.003	.000	.007	.158	.001	.019	.094	.005
8	.277	.069	.002	.003	.000	.000	.085	.011	.000	.032	.000	.026	.004	.001	.000	.000	.010	.001	.007	.005	.000	.033	.095	.079	.023	.047	.001
9	.000	.001	.000	.001	.000	.000	.001	.000	.000	.001	.000	.001	.000	.001	.000	.000	.001	.001	.000	.000	.000	.001	.000	.001	.000	.000	.000
10	.142	.052	.016	.005	.003	.000	.034	.055	.001	.015	.007	.006	.026	.012	.001	.001	.018	.006	.029	.021	.015	.050	.000	.027	.001	.011	.002
11	.090	.000	.000	.001	.000	.000	.001	.006	.000	.016	.000	.000	.000	.000	.000	.000	.000	.000	.000	.004	.000	.001	.005	.003	.000	.004	.000
12	.101	.034	.000	.001	.001	.000	.044	.002	.000	.016	.000	.009	.000	.001	.000	.000	.005	.001	.001	.000	.001	.009	.041	.011	.016	.026	.001
13	.079	.005	.000	.000	.000	.001	.001	.000	.001	.014	.000	.001	.008	.000	.000	.000	.000	.001	.001	.001	.011	.007	.010	.017	.012	.016	.001
14	.019	.009	.000	.003	.000	.000	.047	.000	.000	.003	.000	.000	.000	.002	.000	.001	.001	.000	.000	.000	.000	.007	.014	.001	.003	.007	.000
15	.004	.001	.001	.001	.000	.000	.001	.002	.000	.001	.000	.002	.000	.000	.000	.000	.000	.000	.000	.000	.000	.001	.001	.000	.000	.000	.000
16	.005	.002	.002	.000	.000	.000	.002	.001	.000	.007	.000	.009	.000	.000	.000	.000	.002	.000	.000	.001	.000	.005	.008	.005	.000	.002	.000
17	.032	.037	.011	.001	.001	.000	.003	.003	.000	.018	.001	.008	.017	.001	.000	.001	.037	.002	.004	.001	.000	.006	.091	.008	.015	.024	.000
18	.042	.011	.004	.003	.002	.001	.007	.005	.000	.011	.000	.002	.000	.001	.000	.000	.010	.001	.001	.001	.000	.010	.011	.005	.001	.026	.000
19	.049	.016	.001	.000	.000	.000	.005	.000	.001	.078	.000	.001	.000	.001	.000	.000	.001	.000	.013	.000	.000	.001	.005	.001	.002	.021	.001
20	.053	.002	.000	.000	.000	.000	.002	.246	.002	.003	.020	.038	.005	.008	.000	.000	.000	.001	.001	.000	.000	.000	.002	.021	.000	.001	.000
21	.035	.021	.001	.005	.001	.000	.001	.001	.000	.021	.000	.000	.001	.004	.000	.000	.007	.000	.000	.001	.006	.004	.014	.006	.012	.016	.000
22	.052	.022	.002	.005	.000	.000	.001	.026	.000	.079	.001	.018	.043	.011	.001	.000	.001	.001	.006	.004	.001	.021	.067	.003	.028	.145	.002
23	.232	.021	.020	.013	.001	.000	.015	.038	.000	.003	.042	.041	.024	.006	.002	.002	.043	.030	.012	.062	.030	.032	.001	.019	.018	.034	.001
24	.107	.053	.003	.015	.005	.001	.029	.041	.000	.017	.004	.004	.001	.003	.001	.004	.011	.003	.004	.001	.007	.023	.085	.034	.025	.113	.000
25	.019	.007	.014	.007	.000	.000	.006	.009	.000	.055	.001	.011	.005	.009	.000	.000	.026	.004	.004	.001	.003	.004	.006	.023	.000	.003	.008
26	.056	.026	.026	.064	.002	.001	.047	.071	.000	.003	.024	.055	.032	.017	.005	.017	.050	.047	.017	.222	.048	.147	.002	.075	.023	.027	.003
27	.011	.001	.001	.000	.001	.000	.001	.000	.000	.001	.001	.003	.002	.001	.000	.000	.001	.000	.001	.001	.000	.001	.001	.003	.000	.003	.000

Fig. 7. Bigram frequencies. The matrix on top contains the first three digits of the letter-pair frequencies derived from a 600 000-character sample of legal text. The matrix below shows the class-pair frequencies (script font) from about 21 000 characters. Thus, the frequency of the pair h-e is 0.0216 while the corresponding class pair (which happens to be 20–26) is 0.0222.

bet k to every class i. It is our object to select the most likely mapping in view of the observed bigram frequencies f_{ij} and the known letter-pair probabilities p_{ij}, i.e., to select a T_t such that

$$P\{T_t \mid f_{11}, f_{12}, \cdots, f_{27\,27}\} = \max_s P\{T_s \mid f_{11}, f_{12}, \cdots, f_{27\,27}\}.$$

According to the maximum likelihood principle, this is equivalent to maximizing $P(f_{11}, f_{12}, \cdots, f_{27\,27} \mid T_s)$, which will now be computed according to the independence assumption mentioned earlier.

$$P\{f_{11}, f_{12}, \cdots, f_{27\,27} \mid T_s\} = \prod_{k,l} P\{f_{T_s^{-1}(k), T_s^{-1}(l)}\}$$
$$= \prod_{k,l} [2\pi N p_{kl}(1 - p_{kl})]^{-1/2}$$
$$\cdot \exp \sum_{k,l} -1/2 \frac{(f_{T_s^{-1}(k) T_s^{-1}(l)} - p_{kl})^2}{N p_{kl}(1 - p_{kl})}.$$

To maximize this function, it is sufficient to choose T_s such that the bigram distance function

$$F(T_s) = \sum_{k,l} \frac{(f_{T_s^{-1}(k), T_s^{-1}(l)} - p_{kl})^2}{p_{kl}(1 - p_{kl})}$$

is minimized. The heuristic algorithm used to perform the minimization will be discussed after a brief digression.

Estimation of the Class-Pair and Letter-Pair Frequencies

It is possible to improve upon our rather crude estimates of f_{ij} and p_{ij} by postulating a uniform a priori density function for each pair. The improved estimators may then be derived either by postulating a square-loss function and minimizing the risk, or from the maximum-likelihood formulation.[13]

The resulting correction is particularly helpful in the case of rare bigrams. It tends to reduce the effect of the introduction of anomalous rare bigrams through recognition errors on the intermediate tape, and through typographical errors in the 600 000-character "standard text." For example, without the correction, a nonoccurring letter pair in the "standard text" would play havoc in the denominator of the distance function unless the corresponding frequency were also zero in the "test text."

The a posteriori estimates for the frequencies are

$$f_{ij}' = \frac{N_{ij}+1}{N+2} \quad \text{and} \quad p_{ij}' = \frac{M_{ij}+1}{M+2}$$

where N_{ij} and M_{ij} are the number of occurrences of pair (i, j) in the test and the standard text, and N and M are the total number of pairs in the two sets.

The corrected values are used throughout the evaluation of the distance function $F(T_s)$, but the primes have been suppressed to simplify notation.

The Sorting Algorithm

A precise description of the procedure used to find a satisfactory mapping from the classes defined by the clustering algorithm to the symbols of the alphabet is somewhat tedious; therefore, a quick outline will be offered for the sake of readers lacking in stamina and/or deep-rooted interest.

First, both the classes and the letters of the alphabet are sorted in order of frequency of occurrence. The program then interchanges pairs of classes, starting with pairs close together at the head of the list. An interchange is allowed to stand only if it contributes to a decrease in the distance measure. Only bigram frequencies involving classes (and letters) above the more frequent member of the pair are included in the calculation of the distance measure.

The net effect of this procedure is to expedite the assignment of the most frequently occurring, hence most easily identifiable, classes. Since the singlet frequencies alone constitute a poor identifier for any but the two or three most common letters, consideration of all the bigrams at each step would have a disruptive effect on the ordering. Once the more common characters have been identified, the bigrams in which they occur are sufficient to sort the letters with less dependable distribution.

The final value of the distance measure provides a reliable indication of the success of the whole procedure. If the clustering was inadequate, or the categorization very poor, or if the total number of samples is insufficient for the operation of the decoding algorithm, then the distance figure will remain at a level several orders of magnitude above its normal terminal value.

For a more formal statement of the algorithm, the following definitions will be needed.

Let the classes on the intermediate tape be represented by the index i, $i=1, 2, \cdots, 27$. The blank, recognized and tagged as such by the scanning circuitry, is denoted by $i=1$; the remaining numbers carry no information about the identity of the class they represent.

Let α_j, $j=1, 2, \cdots, 27$, constitute a list of the symbols of the alphabet in order of frequency of occurrence:

$$\sum_{h=1}^{27} p_{h,j} \geq \sum_{h=1}^{27} p_{h,j+1}.$$

Thus α_1 is the blank, $\alpha_2=$e, $\alpha_3=$t, and $\alpha_{27}=$z.

The series of one-to-one transformations (from i to j) evaluated by the program are denoted by T_n. The initial mapping, depending only on the marginal frequencies, is:

$$T_1(i) = j, \qquad \sum_{h=1}^{27} f_{h,T_1^{-1}(j-1)} \geq \sum_{h=1}^{27} f_{h,i}.$$

The rule for the formation of successive mappings by exchanging the assignments corresponding to the pair of entries k and l is:

$$T_{n+1}(i) = T_n(i), \qquad i \neq k,\ i \neq l,\ k < l$$

$$\left.\begin{aligned} T_{n+1}(k) &= T_n(k) \\ T_{n+1}(l) &= T_n(l) \end{aligned}\right\} \text{ if}$$

$$\begin{aligned} &\sum_{i=1}^{k} \frac{[f_{i,k} - p_{T_n(i),T_n(k)}]^2}{p_{T_n(i),T_n(k)}[1 - p_{T_n(i),T_n(k)}]} + \sum_{i=1}^{k-1} \frac{[f_{k,i} - p_{T_n(k),T_n(i)}]^2}{p_{T_n(k),T_n(i)}[1 - p_{T_n(k),T_n(i)}]} \\ &\leq \sum_{i=1}^{k} \frac{[f_{i,k} - p_{T_n(i),T_n(l)}]^2}{p_{T_n(i),T_n(l)}[1 - p_{T_n(i),T_n(l)}]} + \sum_{i=1}^{k-1} \frac{[f_{k,i} - p_{T_n(l),T_n(i)}]^2}{p_{T_n(l)T_n(i)}[1 - p_{T_n(l)T_n(i)}]}, \end{aligned}$$

and

$$\left.\begin{aligned} T_{n+1}(k) &= T_n(l) \\ T_{n+1}(l) &= T_n(k) \end{aligned}\right\} \text{ otherwise.}$$

Each possible interchange is tried by permuting the indices as follows:

$$k(1) = 2$$

$$l(1) = 3$$

$$\left.\begin{aligned} k(n+1) &= k(n) \\ l(n+1) &= l(n)+1 \end{aligned}\right\} \text{ if } l(n) \neq 27$$

$$\left.\begin{aligned} k(n+1) &= k(n)+1 \\ l(n+1) &= k(n)+2 \end{aligned}\right\} \text{ otherwise.}$$

A sample of the behavior of the algorithm, listing the successive mappings, is shown in Fig. 8.

It is possible, of course, to devise many other heuristic algorithms for obtaining the best mapping. Several of these have been tried on segments of a 100 000-character keypunched text artificially scrambled by a separate program. The chief merits of the final algorithm, com-

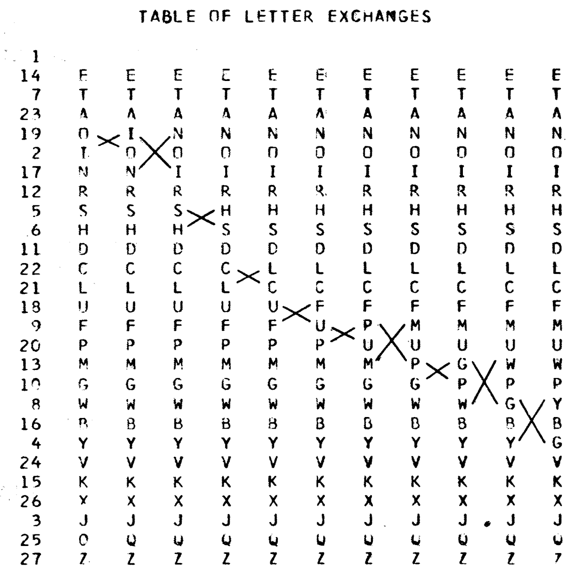


TABLE OF LETTER EXCHANGES

1											
14	E	E	E	E	E	E	E	E	E	E	E
7	T	T	T	T	T	T	T	T	T	T	T
23	A	A	A	A	A	A	A	A	A	A	A
19	O	I	N	N	N	N	N	N	N	N	N
2	I	O	O	O	O	O	O	O	O	O	O
17	N	N	I	I	I	I	I	I	I	I	I
12	R	R	R	R	R	R	R	R	R	R	R
5	S	S	S	H	H	H	H	H	H	H	H
6	H	H	H	S	S	S	S	S	S	S	S
11	D	D	D	D	D	D	D	D	D	D	D
22	C	C	C	C	L	L	L	L	L	L	L
21	L	L	L	L	C	C	C	C	C	C	C
18	U	U	U	U	U	F	F	F	F	F	F
9	F	F	F	F	F	U	P	M	M	M	M
20	P	P	P	P	P	P	U	U	U	U	U
13	M	M	M	M	M	M	M	P	G	W	W
10	G	G	G	G	G	G	G	G	P	P	P
8	W	W	W	W	W	W	W	W	W	G	Y
16	B	B	B	B	B	B	B	B	B	B	B
4	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	G
24	V	V	V	V	V	V	V	V	V	V	V
15	K	K	K	K	K	K	K	K	K	K	K
26	X	X	X	X	X	X	X	X	X	X	X
3	J	J	J	J	J	J	J	J	J	J	J
25	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
27	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z

Fig. 8. Successive mappings for *script* features. The leftmost column represents the classes labeled in the previous stage; on the right appear the alphabetic identities assigned on the basis of the bigram frequencies. New mappings are entered whenever successful exchanges take place; therefore, adjacent columns always differ in exactly two entires.

pared to the others, are simplicity, ability to decipher relatively short samples, and ease of implementation on a digital computer (on the average, a 25 000-character sample requires 150 seconds on an IBM 7094).

IV. Experiments

The data used to test the foregoing ideas were scanned at the experimental character recognition facility of the IBM Watson Research Center. Before passing on to the behavior of the algorithms and the performance levels obtained, the salient features of this system, copiously described elsewhere,[4],[14],[16],[20] will be summarized.

Data Acquisition

The conversion of the printed character to a 15×24 binary matrix representing the black and white points is performed by a cathode ray tube reflection scanner. Careful optical design and electronic compensation circuitry ensure adequate positional linearity and signal-to-noise ratio over an 8 by 10 inch area of the page. Details of the scanning unit may be obtained from earlier work.[4],[20]

The location of the text on the page, document changes, separation of adjacent characters, threshold and line width control, and the suppression of stray noise bits are accomplished by an IBM 1401 computer through a buffered interface.[16] Samples of digitalized characters are shown in Fig. 9.

To achieve registration invariance and to reduce the number of bits to be processed, the video bits from the scanner are shifted through every position of a long shift register. A set of 96 AND gates, each with five to nine inputs, are wired to the shift register to set latches whenever the geometrical configurations they represent have occurred anywhere on the character.[16]

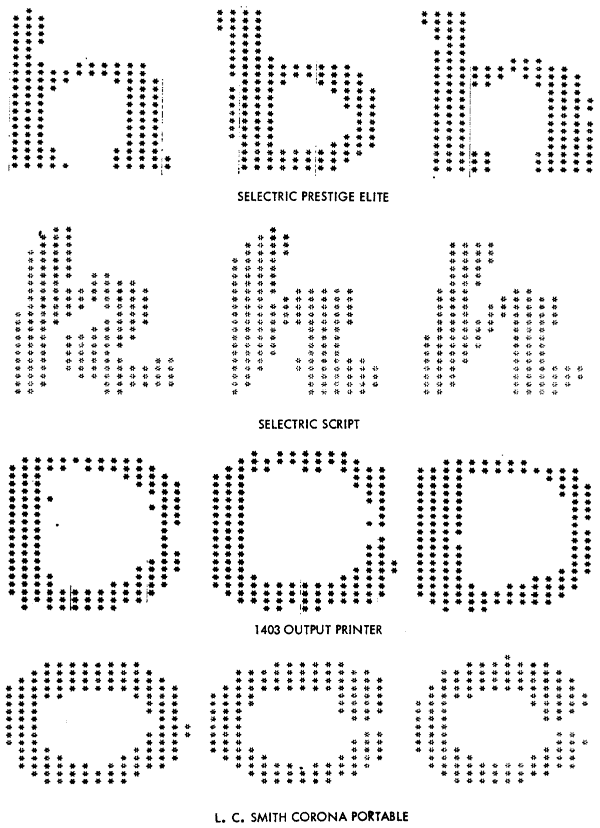


Fig. 9. Scanned characters. The print on the input documents is transformed into a binary number by comparing the intensity of the light reflected from various parts of the character to a threshold. The threshold is adjusted from character to character by means of a formula based on the nominal width of the lines.

TABLE I

Experimental Results

The first two columns refer to the number of displaced samples in the clustering operations on the 3000 characters of primary input. The third column refers to the classification on the 20 000 samples, while the fourth column includes all errors in the text as printed out after the decode phase.

Data	Percent Error Rates			
	Initial Clusters	Merged Clusters	Categorizer Output	Decoder Output
Prestige Elite	0	0	0.2	0.2
Script	0.2	0.2	2.0	2.0
Smith-Corona Manual Typewriter	1.6	9.7	—	—
IBM 1403 Printer	2.2	8.1	—	—

Results

The principal results of the experiments are summarized in Table I, where the successive columns represent the overall error rate at the end of each stage. Although only four fonts were tested, they are sufficiently diverse to span a considerable portion of the space of impact-printed characters.

The first font, *selectric prestige elite*, is typical of commercial typescript. The performance of the various algorithms on this material is shown in Fig. 10. While with *prestige elite* 32 classes almost suffice for correct

Majority	32 clusters			64 clusters						128 clusters												after joining clusters			Classification of complete text		Crypt
I.D.	#1	#2	#3	#1	#2	#3	#4	#5	#6	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#1	#2	#3	#1	#2	
A	268			111	157					34	77	98	59									268			1653(14S)		1653 (14S)
B	17	10		8	9	9	1			7	1	5	4	3	6	1						26	1*		288 (1H)		288 (1H)
C	96			42	54					18	24	26	28									96			694		694
D	125	48	4(1Q)	91	34	24	24	4		60	31	7	27	2	22	6	18	2	2			173	2		1063 (4Q)		1063 (4Q)
E	204	202		55	147	107	97			45	10	122	25	23	84	85	12					406			2816		2816
F	80			44	36					15	29	7										80			561		561
G	34(1S)			34						14	20											34			261(1S)		261(1S)
H	100	71		69	31	47	24			37	32	16	15	24	23	12	12					140			1170 (1B)		1170 (1B)
I	169			77	92					49	28	74	18									169			1428		1428
J	5			2	3					1	1	2	1									3	1	1*	41	7	41
K	29(1X)			29						16	13											29			154		154
L	113			59	54					30	29	20	34									113			757 (1I)		757 (1I)
M	78			30	48					21	9	26	22									78			444		444
N	83	137(1H)		26	57	97	40(1H)			38	39	58	12	45	25	1	2(1H)					149(1H)			1581 (2H, 1Q)		1581 (2H, 1Q)
O	194			76	118					75	1	37	81									194			1524		1524
P	17	14	18	10	7	8	6	10	8	5	5	2	5	7	1	3	3	6	4	3	5	39	1*		390		390
Q				1						1												1*			22(10D)		22 (10D)
R	178			55	123					25	30	35	88									178			1275		1275
S	99	71(1X)		1	32	67	71			1	16	16	13	54	58	13						170	1*		1129 (4Z)		1129 (4Z)
T	340			118	222					53	65	82	140									340			2219		2219
U	47			25	22					22	3	17	5									47			427 (1Q)		427 (1Q)
V	19 (3X)			19						12	7											19			215		215
W	61			42	19					21	21	17	2									61			395 (1I)		395 (1I)
X				1	1	3				1	1	1	2									4	1*		48 (3Z)		48 (3Z)
Y	61			25	36					17	8	29	7									61			382		382
Z																											(7J)

Fig. 10. Details of splitting and joining. This table shows the makeup of the various subclusters existing at each stage of the prccessing of the *prestige elite* font. After the final joining the starred clusters were eliminated (losing the only q cluster). Many of the q's were still recognized correctly, however, because of the similarity with the small cluster of d's. The z's, on the other hand, were completely mis-identified in the output text.

Assigned Cluster Number	Identity of unknown sample																									
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	$\underline{1318}$		2					1	1								1	3		1	3					
5		$\underline{203}$																								
13			$\underline{557}$					1	1														1			
4	2		2	$\underline{652}$	3									1				1								
17			3		$\underline{1985}$						1							1				1				5
25						$\underline{339}$						1			1											
8			1				$\underline{199}$								1	1	1			2						
6								$\underline{671}$			1		1							1						
11				1					$\underline{1021}$		1								1							
12				1			1	1	1	$\underline{34}$						8									28	1
7	1				1			2			$\underline{87}$								1							
26					1	1			7			$\underline{580}$								3						1
18	1												$\underline{269}$													
19	2		1										2	$\underline{1128}$												
22					1						1	1			$\underline{1081}$	5	1									
9																$\underline{355}$										
10																	$\underline{36}$									
16	2		1	1	1								2	1	1			$\underline{973}$								
15	2			1	1	1		17	1		1		1	13	2	1	1	3	$\underline{983}$	1	2		1		2	1
1			1															1		$\underline{1453}$						
3	5																	1			$\underline{418}$		2			
20				1	1			1			2		3	1		4		1	6		1	$\underline{206}$	26		15	
2															1								$\underline{216}$			
21									1				1		1			2	1	1			2	$\underline{35}$		11
23															1										$\underline{139}$	
24																									32	$\underline{0}$

Fig. 11. Categorizer output for the *script* font. The true identities of the classified samples are listed on top, while the labels of the classes derived by the clustering procedure appear on the left. Ideally, all the samples of the same identity would be uniquely assigned to a single class. The underlined diagonal terms denote samples correctly identified by the cryptogram algorithm.

```
ME VAMES ADAMS WHO WAS INDICTED BUT NOT TRIED WITH THE DEFENDANTS TESTIFIED
THAT HE WAS WITH THE DEFENDANTS AND SCHWARTE AT THE CANDY STORE ON THE NIGHT
 PALMER WAS MURDERED THAT THES LEFT THE STORE ABOUTS M IN THE TWO CARS THE O
LDSMOBILE LEADING AND THE CHRYSLER FOLLOWING HE IDENTIFIED THE OCCUPANTSOF E
ACH CAR BUT DENIED IHAT HE HAD AGREED TO ASSAULT ANVONE OR THAT SUCH AN AGRE
EMENT WAS MADE IN HIS PRESENCE BV AN ASSISTANT STATE S ATTORNEV AND FOUR DEF
ENSE COUNSEL HE TESTIFIED THAT ALL OF THE CODEFENDANTS WERE AT THE CANDV STO
```

Fig. 12. Final output of processor on the *script* font. The readability of the output on the relatively difficult *script* font is decreased, beyond the mistakes introduced by the processor, by the absence of all punctuation, and the omission of italics, boldface, numerals, and other symbols in the original (Fig. 1).

clustering, even here the salvage of the "x" category yields some indication of the value of extending the preliminary clustering to 128 classes.

Script font represents an additional degree of difficulty. Here also the major portion of the final error rate is introduced in the categorization stage, as shown in Fig. 11. Because the *n*-tuples are less effective in representing *script* characters, the Hamming distance between the cluster centers derived from 3000 characters is less than in *prestige elite*, with a corresponding increase in misclassified letters. The cryptogram program is not, however, affected by these misclassifications, and makes the best possible assignment of the output of the categorizer. The labels assigned to each sample are printed out in the original sequence in Fig. 12.

A major difficulty is presented by the y's, which are distributed among four different classes by the categorizer (two of the groups also contain other identities). The larger "pure" y cluster is eventually called "z," mainly by default.

In order to extend the technique to the limit, two problem fonts were also tried. In the IBM 1403 chain printer outputs, the D's and O's are barely distinguishable by eye, while in the elderly L. C. Smith (1920 vintage) typewriter at our disposal, most of the o's and c's are completely filled in, and so are many of the a's and s's. The initial clustering algorithm is able to resolve all the classes in both fonts, but the merging algorithm is not yet sufficiently sophisticated to keep them separate. Except for the pairs just mentioned, the performance of the clustering algorithm seems adequate in both fonts (1.6 percent and 0.7 percent error, respectively). The cryptogram program could not, of course, make any headway with either of these fonts, since the basic objective of the program—to match the two bigram tables—was clearly impossible.

The legal case histories used in these experiments are particularly challenging to the crypt-algorithm even without the occurrence of misclassification because of the constant repetition of proper names containing unusual bigrams. Perhaps the severest test encountered so far involves the case of Ho*ck*er of Ro*ck*land County and his bla*ck* Bui*ck*. Anomalies of this nature are the main reason for the large number of samples necessary for successful operation of the algorithm.

V. Conclusion

It has been shown that printed characters of several typical fonts are sufficiently well separated in hyperspace, at least in the binary feature representation, to allow an automatic clustering scheme to separate them into distinct clusters corresponding to the letters of the alphabet. The preliminary cluster assignments, based on a simple iteratively minimized distance criterion, have been further refined by "splitting," "joining," and "omit" routines.

The cluster centers obtained from a few thousand characters of a segment of English text were used to classify all of the characters of this text. The transition probabilities of the classes were correlated against the known letter-pair probabilities of the English language in order to establish the correspondence between the class identities and the members of the alphabet. This was accomplished by an iterative algorithm minimizing a distance function defined on the observed and expected bigram probabilities.

The experiments reported here cannot be construed as a demonstration of a practical automatic reading machine. Much remains to be done, for instance, in coping with punctuation, nonalphabetic symbols such as numerals and mathematical symbols, and intermixed upper and lower case letters. The recognition of cast font material has not even been attempted yet because of the difficulty of separating adjacent characters even on high-quality print.

Present work is directed toward application of the clustering procedure to the video space itself.[2] Because of the systematic correlations between certain video bits due to misregistration, the simple distance measures discussed here are inadequate to separate the most difficult classes. Alternatives here are to include sophisticated and time-consuming registration routines, or to transfer operations to a registration invariant domain, by means, for example, of the autocorrelation transform.

Further decreases in the sample size required for faultless performance in the decoding phase could be obtained by resorting to the devices advocated by textbooks on cryptography.[12],[21] Although consideration of common words, suffixes and prefixes, etc., would reduce the generality of the program, for specific applications these more sophisticated methods of cryptoanalysis would certainly be resorted to.

While the accuracy of the categorizer is not sufficient for many purposes, one may already envision applications to the customary goals of character recognition devices such as language translation and automatic indexing and abstracting. One particularly promising area is the "natural language" search program developed by Magnino,[17] which is relatively insensitive to spelling errors. Error-correcting procedures based on dictionary look-up[7] may further enhance the usefulness of the categorizer.

The experience gained in developing a recognition method independent of the particular geometrical configuration of the characters in the text has also been applied to other tasks. Portions of the clustering algorithm were used to group Chinese characters,[8] to design ternary references for a commercial multifont reader, to derive Boolean recognition logic, and to classify the entries, on the basis of a philological ques-

[2] *Note added in proof:* this endeavor has proved to be successful and will be reported.

tionnaire, in a dictionary designed for automatic translation. The figure-of-merit from the cryptogram program was applied to detecting errors in the output of a conventional categorizer.

Acknowledgment

G. Shelton, Jr. first proposed the notion of linking a clustering program to a cryptoanalysis program in an attempt to make up a self-sufficient reading system. His attention and advice since the inception of the project have been a periodic source of encouragement to us. L. Loh, who programmed the majority of the clustering algorithms, also contributed many excellent suggestions for their improvement. Mary Ellen Barrett is responsible for the coding of the "Crypt" algorithm. We are grateful to C. Marr and A. Sebastiano for their careful supervision of the data scanning process.

We also acknowledge, as a possible source of inspiration, sundry peripatetic conversations on the subject of clustering with our colleagues in the Systems Science Group, with Rubin, Friedman, Bonner, and Abraham of other IBM departments, and with Ball, Hall, and Singleton of Stanford Research Institute.

References

[1] C. Abraham, "Evaluation of clusters on the basis of random graph theory," IBM Research Memo., November 1962.

[2] G. H. Ball, "Data analysis in the social sciences: What about the details?," *1965 Fall Joint Computer Conf., AFIPS Proc.*, vol. 27, pt. 1. Washington, D.C.: Spartan, 1965, pp. 533–559.

[3] G. H. Ball and D. J. Hall, "ISODATA, a novel method of data analysis and pattern classification," Stanford Research Inst., Menlo Park, Calif., Tech. Rept., April 1965.

[4] H. G. Baskin, R. Bakis, R. J. Potter, J. Reines, and G. Shelton, "System description of a multifont page reading machine," IBM, Yorktown Heights, N. Y., Rept. RC 1052, October 1963.

[5] R. E. Bonner, "A 'logical pattern' recognition program," *IBM J. Research and Develop.*, vol. 6, pp 353–359 July 1962.

[6] ——, "On some clustering techniques," *IBM J. Research and Develop.*, vol. 8, pp. 22–32, January 1964.

[7] G. Carlson, "Techniques for replacing characters that are garbled on input," *1966 Spring Joint Computer Conf., AFIPS Proc.*, vol. 28. Washington, D.C.: Spartan, 1966, pp. 189–192.

[8] R. Casey and G. Nagy, "Recognition of printed Chinese characters," *IEEE Trans. Electronic Computers*, vol. EC-15, pp. 91–101, February 1966.

[9] D. B. Cooper and P. W. Cooper, "Nonsupervised adaptive signal detection and pattern recognition," *Information and Control*, vol. 7, pp. 416–444, September 1964.

[10] W. D. Fisher, "On grouping for maximum homogeneity," *J. Am. Statistical Assoc.*, vol. 53, pp. 789–798, December 1958.

[11] H. P. Friedman and J. Rubin, "On some invariant criteria for grouping data," IBM, New York Scientific Center, N. Y., Tech. Rept. 39 001, April 1966.

[12] H. F. Gaines, *Cryptanalysis*. New York: Dover, 1956.

[13] I. J. Good, *The Estimation of Probabilities, Research Monograph 30*. Cambridge, Mass.: M.I.T. Press, 1965.

[14] L. A. Kamentsky and C. N. Liu, "Computer-automated design of multifont print recognition logic," *IBM J. Research and Develop.*, vol. 7, pp. 2–14, January 1963.

[15] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proc. 5th Berkeley Symp. on Probability and Statistics*, pp. 281–297, 1967.

[16] C. N. Liu and G. L. Shelton, Jr., "An experimental investigation of a mixed-font print recognition system," *IEEE Trans. Electronic Computers*, vol. EC-15, pp. 916–925, December 1966.

[17] J. J. Magnino, "CIS—a computerized normal text current awareness technique," *Industrial Electronics* (London), vol. 4, pp. 269–273, June 1966.

[18] L. Ornstein, "Computer learning and the scientific method: a proposed solution to the information theoretical problem of meaning," *J. Mount Sinai Hospital*, vol. 32, pp. 437–494, July–August 1965.

[19] E. A. Poe, "The Gold Bug" (1843), in *Poe's Best Tales*. New York: Random House, 1924.

[20] R. J. Potter, "An optical character scanner," *J. Soc. Photographic Instrumentation Engineers*, vol. 2, pp. 75–78, February–March 1964.

[21] F. Pratt, *Secret and Urgent*. New York: Bobbs-Merrill, 1939.

[22] J. Rabinow, "Diverse reading machine," U. S. Patent 3 237 161, February 1966.

[23] D. J. Rogers and T. T. Tanimoto, "A computer program for classifying plants," *Science*, vol. 132, pp. 1115–1118, October 21, 1960.

[24] J. H. Ward, Jr., "Hierarchical grouping to optimize an objective function," *J. Am. Statistical Assoc.*, vol. 58, pp. 236–244, March 1963.

Short Notes

More Efficient Use of the *F* Matrix in Practical Circuit Analysis Programs

S. R. SEDORE

***Abstract*—This note defines a general *F* matrix that arises from a tree that is formed according to a specific element priority. An alternative based on this matrix is presented to the usual method of repeated matrix manipulation for arriving at the solutions of various network quantities. The conditions required for this alternative and its limitations are also presented.**

Manuscript received October 18, 1967; revised February 13, 1968. This work was supported in part by the Air Force Weapons Laboratory under Contract AF29 (601)-6852.

The author is with the IBM Federal Systems Division, Electronics Systems Center, Owego, N. Y. 13827

***Index Terms*—Automatic circuit analysis program, economy in machine time, *F* matrix derivation, fundamental circuit matrix, fundamental cut set matrix, row and column processing in lieu of matrix manipulation.**

I. Introduction

A number of automatic circuit analysis programs have been developed that operate along the general lines given by Bashkow [1] and expanded by Bryant [2]. The A matrix given in that paper permits the matrix equation

$$\dot{Y} = AY + BU + N\dot{U} \tag{1}$$

which summarizes the general transient problem in terms of the state variable vector Y. Another matrix, however, comes into prominence when the practical solution of (1) is to be implemented. This matrix, called the F matrix [3] here, can be derived from two basic matrices of incidence [4]. It may then be used together with basic