COMPUTER GRAPHICS AND IMAGE PROCESSING 10, 235–245 (1979)

Ambiguity Reduction in Handwriting with Ambiguous Segmentation and Uncertain Interpretation *

SHMUEL PELEG

Computer Science Center, University of Maryland, College Park, Maryland 20742

Received February 20, 1979

A connected piece of handwriting can be segmented into letters in many possible ways, and each segment can also be interpreted as one of several possible letters. Each segmentation, and each choice of interpretations for the segments, determines a word. An iterative relaxation algorithm is presented that considerably reduces the number of possible interpretations for the handwriting, while preserving the most likely interpretations.

1. INTRODUCTION

A common approach to handwritten word recognition is to segment the writing into strokes based on features such as maximum height and maximum curvature. Strokes are then combined into letters in all possible ways, with every combination having a merit function as to its being any given letter [1]. It is desirable to combine the strokes into letters in such a way that a sequence of nonoverlapping letters will be created, and that this sequence will represent a word of the language.

A possible solution to this problem is to compute a merit function for every possible combination of strokes, and choose the combination having the maximal merit. Unfortunately the number of possible combinations is very large. In this paper a compact graph representation for all possible combinations will be presented. A relaxation process, working in parallel on all nodes of this graph, is described. This process reduces the ambiguity in every node of the graph so that the number of possible combinations is reduced, while retaining the interpretations with high merit for every possible path through the graph. Relaxation processes have often been used in image processing [2, 3]. A new approach to relaxation [6] has made it easier to apply relaxation to problems outside the image processing domain [7].

* The support of the National Science Foundation under Grant MCS-76-23763 is gratefully acknowledged, as is the help of Kathryn Riley in preparing this paper.

0146-664X/79/070235-11$02.00/0

2. HANDWRITING WITH AMBIGUOUS SEGMENTATION

Let $S = \{s_1, \ldots, s_L\}$ be a segmentation of the handwriting into strokes, where s_{i+1} immediately follows s_i, $1 \leq i \leq L$. Letters are built from sequences of strokes. Every possible subsequence of strokes is a candidate for a letter. A sequence of strokes that can be combined into a letter is a segment. In many cases a segment can represent more than one letter. It is shown in [1] how a merit function can be computed for each of these letters, based on the properties of the strokes that constitute the segment. Given these merit values, we assign probabilities to the various alternative letters.

Let $m_A, \ldots, m_Z$ be the merit values associated with the letters A–Z for the segment x. We normalize these values into probabilities:

$$P_x(\alpha) = \frac{m_\alpha}{\sum_{\lambda=A \cdots Z} m_\lambda}.$$

(Interpretation: the probability that segment x represents the letter α is $P_x(\alpha)$.) No probabilities are computed for segments that cannot be interpreted as any letter.

Determining the specific letter represented by a segment is only one problem in recognizing a word. A very important problem is to find an admissible segmentation, such that the segments chosen succeed each other, and cover the entire word. A formal treatment of the choice of segments can be found in [4]. Segments such as $x = \{s_1, s_2, s_3\}$ and $y = \{s_2, s_3\}$ can both be built from the set of strokes, but cannot occur together in an admissible segmentation into letters, since they overlap.

In the following section a compact representation for this ambiguous segmentation is presented, that enables easy handling of the choice of an admissible segmentation.

2.1. Representation

A handwritten word with ambiguous segmentation will be represented by a directed acyclic graph $G = (V, E)$. The set of nodes V consists of the following nodes:

1. A node v_i for every segment $x_i \subseteq S$ that can represent at least one English letter.
2. v_s, the initial node.
3. v_f, the final node.

The set of arcs, E, consists of the following arcs:

1. The arc $\langle v_i, v_j \rangle$ for every pair of segments x_i, x_j such that x_j immediately follows x_i.
2. The arc $\langle v_s, v_i \rangle$ for every segment x_i that includes the first stroke of S, s_1.
3. The node $\langle v_i, v_f \rangle$ for every segment x_i that includes the last stroke of S, s_L.

For every path $v_{i_0}, v_{i_1}, \ldots, v_{i_{k+1}}$ such that $v_{i_0} = v_s$ and $v_{i_{k+1}} = v_f$, it is clear that

$v_{i_1}, \ldots, v_{i_k}$ represents an admissible segmentation, since the segments $x_{i_1}, \ldots, x_{i_k}$ are nonoverlapping, and cover all the strokes. For a formal proof see [4].

Since each segment can represent several possible letters, a probability vector P_i is assigned to every node. The probability vector P_i associated with the node v_i contains the probability that segment x_i represents each possible letter, and is computed from the letter merit functions as described earlier.

Figure 1 displays three graphs representing the handwritten words "let," "even," and "book." In order to facilitate the operations that will be described in the sequel, every word is bounded by two space marks (#) on each side, and one node on each side serves as initial or final node. The probability vector assigned to each node is represented by a list of pairs near the node. This list represents the probability ($\times 100$) of each nonzero entry in the probability vector. When only one letter is possible, it is listed with no probability assigned. In Fig. 1a the word "let" is on the path 1-2-4-6-5-7-8; in 1b, the word "even" is on the path 1-2-3-6-10-13-14-15; and in 1c the word "book" is on the path 1-2-3-7-13-15-17-18.

To determine what word is represented by the handwriting, two choices must then be made:

1. Determine a path from the initial node to the final node.
2. At every node on the path, determine the letter represented by the node.

The above choices cannot be separated, and have to be made concurrently.

2.2. Maximal Interpretation

When faced with a problem of choice among many possibilities, a common practice is to assign "merits" to each possible choice, and choose those possibilities having high merit. In our case the merit function for every possible word will consist of two parts: the segmentation merit and the language merit.

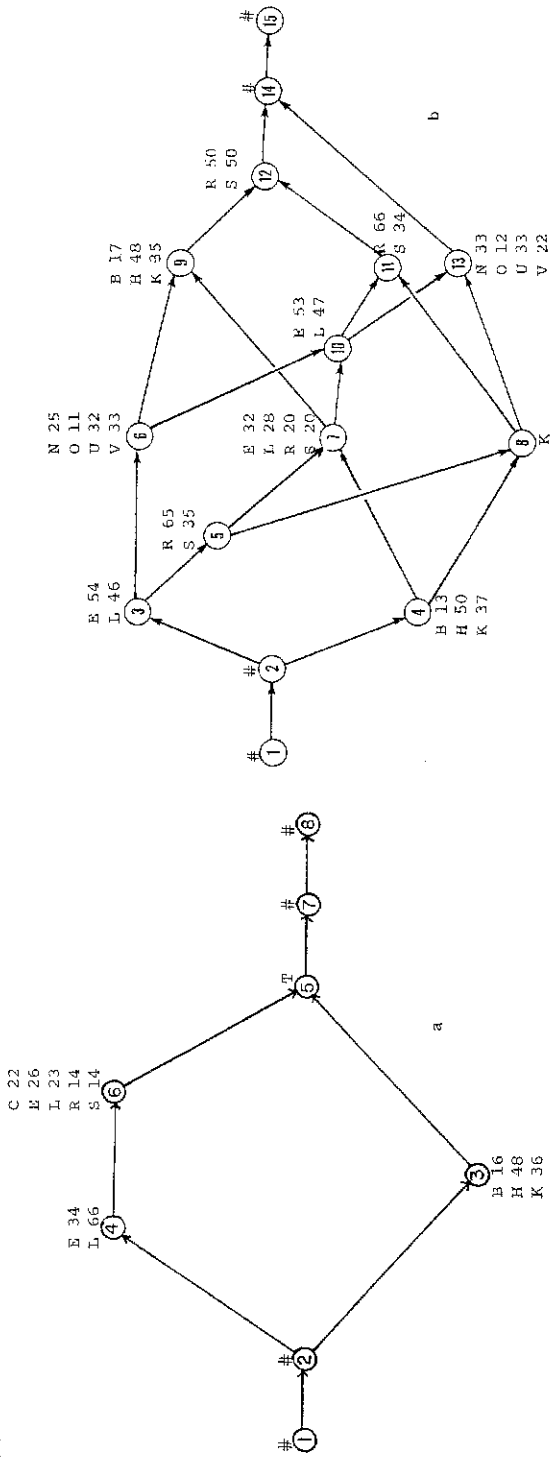
Given a word $\alpha_1 \cdots \alpha_N$ built from the path $v_{i_1}, \ldots, v_{i_N}$ (the initial and final nodes excluded), the segmentation merit is defined by

$$\mu_S = \prod_{k=1}^{N} P_{i_k}(\alpha_k).$$

This merit function will give the highest values to words using the letters that have the highest probabilities at every node. It will exclude (give zero merit to) all words using a letter that has zero probability at the corresponding node. The information used by this function is derived from the segmentation of the handwritten word into strokes.

The language merit corresponds to the probability of occurrence of the word in English. Since it is impractical to store all English words with their corresponding probabilities, the merit is estimated using a simplified stochastic model. The language is approximated as a stochastic process, with each letter depending on the two letters that precede it. Trigram and diagram statistics of English are used to find the parameters of this process. The recursive expression is

$$\text{Prob}(\alpha_1\alpha_2\alpha_3) = f(\alpha_1\alpha_2\alpha_3)$$


a
C 22
E 26
L 23
R 14
S 14
E 34
L 66
B 16
H 48
K 36
b
N 25
O 11
U 32
V 33
E 54
L 46
R 65
S 35
E 32
L 28
R 20
S 20
E 53
L 47
B 17
H 48
K 35
R 50
S 50
R 66
S 34
N 33
O 12
U 33
V 22
B 13
H 50
K 37
c
B 41
H 35
K 24
L 24
R 24
S 24
E 28
U 22
N 19
H 19
O 15
K 13
V 08
B 04
V 32
U 27
O 23
N 18
R 64
S 36
R 53
S 47
R 50
S 50
L 58
E 42
M 35
W 65
L 67
E 33
E 34
L 30
R 18
S 18
A 37
S 37
O 25
V 32
U 27
O 23
N 18
H 46
K 33
B 21

and

$$\text{Prob}(\alpha_1 \cdots \alpha_{N-1}\alpha_N\alpha_{N+1}) = \text{Prob}(\alpha_1 \cdots \alpha_{N-1}\alpha_N) \cdot \text{Prob}(\alpha_{N-1}\alpha_N\alpha_{N+1} \mid \alpha_{N-1}\alpha_N)$$

where

$$\text{Prob}(\alpha_{N-1}\alpha_N\alpha_{N+1} \mid \alpha_{N-1}\alpha_N) = \frac{\text{Prob}(\alpha_{N-1}\alpha_N \mid \alpha_{N-1}\alpha_N\alpha_{N+1}) \cdot \text{Prob}(\alpha_{N-1}\alpha_N\alpha_{N+1})}{\text{Prob}(\alpha_{N-1}\alpha_N)}$$

$$= \frac{\text{Prob}(\alpha_{N-1}\alpha_N\alpha_{N+1})}{\text{Prob}(\alpha_{N-1}\alpha_N)} = \frac{f(\alpha_{N-1}\alpha_N\alpha_{N+1})}{f(\alpha_{N-1}\alpha_N)}$$

since $\text{Prob}(\alpha_{N-1}\alpha_N \mid \alpha_{N-1}\alpha_N\alpha_{N+1}) = 1$. The f's are estimates of the digram and trigram probabilities based on statistics taken from a large sample of English text. For the actual computation of the language merit we surround each word by two space symbols (#) at each end, and set $M_L(\alpha_1 \cdots \alpha_N) = \text{Prob}(\#\#\alpha_1 \cdots \alpha_N\#\#)$. The language merit gives high probability to words that are common according to the stochastic model. Words excluded are words having illegal trigrams.

The two merit functions are multiplied together to give the combined effect of the language and the segmentation. A similar combination of two merit functions is used in [5], where the language merit is computed from a stochastic grammar. This merit function,

$$M(\alpha_1 \cdots \alpha_N) = M_S(\alpha_1 \cdots \alpha_N) \cdot M_L(\alpha_1 \cdots \alpha_N)$$

favors words having both high segmentation merit and language merit, and excludes all forbidden words from either the language or the segmentation. Table 1 shows the five words having the highest merit for each graph of Fig. 1. Note that in each case the original word has the highest merit.

As mentioned earlier, a path from the initial node to the final node and a choice of one letter at each node on this path determines a word. In order to find words with high merit, it is necessary to find all paths from the initial node to the final node and on each path make all the possible choices at every node. Even small graphs such as that in Fig. 1 yield a huge number of possible words. Finding all the words, computing the merit function for each, and choosing words with high merit, involves a considerable amount of work.

The next sections describe how the ambiguity at each node can be reduced by a relaxation algorithm. This algorithm, which works locally on every node, can significantly reduce the number of possible words, without eliminating words having high merit.

3. PROBABILITY UPDATING

The problem of probabilistic graph labeling is as follows:

Let $G = (V, E)$ be a graph with $V = \{v_1, \ldots, v_n\}$ the set of nodes and $E \subset V \times V$ the set of arcs, and let $\Lambda = (\lambda_1, \ldots, \lambda_L)$ be a set of labels. With

FIG. 1. The original probabilistically labeled graphs for three handwritten words. (a) "LET" (13 possible words). (b) "EVEN" (848 possible words. (c) "BOOK" (3960 possible words).

every node $v_i \in V$ a random variable l_i is associated, specifying the probabilities of the possible labels for that node. Initially, based on some measurements, a probability distribution $P_i^{(0)}: \Lambda \to [0, 1]$ is estimated for every random variable l_i. In this section the updating of these distributions is discussed, based on the distributions at neighboring nodes, and on statistical relations among the random variables l_i.

Given a graph $G = (V, E)$, a set of labels Λ, and an estimated discrete probability distribution $P_i: \Lambda \to [0, 1]$ for each random variable l_i, new estimated probability distributions for l_i are to be computed.

We can regard the probability vectors P_i as events, i.e., we can think of them as being chosen from a space of possible vectors. We shall now consider various prior and conditional probabilities involving these P_i events and the outcomes of the random variables l_i. In particular, we shall consider probabilities of the form

(1) $\text{Prob}(l_i = \alpha | P_i)$; this is just the probability that node v_i has label α, given that its probability distribution of labels is P_i. We denote this probability by $P_i(l_i = \alpha)$.

(2) $\text{Prob}(P_i | l_i = \alpha)$; this is the probability that the distribution for node v_i is P_i, given that the observed label of v_i is α.

Evidently we have $P_i(l_i = \alpha) = \text{Prob}(P_i | l_i = \alpha) \cdot \text{Prob}(l_i = \alpha)/\text{Prob}(P_i)$. (The problem of actually calculating $\text{Prob}(P_i)$ will not be considered yet.)

Let v_i, v_j, and v_k be three nodes such that v_j and v_k are neighbors of v_i. We can consider the joint events $(l_i = \alpha, l_j = \beta, l_k = \gamma)$ and (P_i, P_j, P_k), and write

$$\text{Prob}(l_i = \alpha, l_j = \beta, l_k = \gamma | P_i, P_j, P_k) \equiv P_{ijk}(l_i = \alpha, l_j = \beta, l_k = \gamma)$$

$$= \frac{\text{Prob}(P_i, P_j, P_k | l_i = \alpha, l_j = \beta, l_k = \gamma) \cdot \text{Prob}(l_i = \alpha, l_j = \beta, l_k = \gamma)}{\text{Prob}(P_i, P_j, P_k)}. \quad (1)$$

We now assume that

$$\text{Prob}(P_i, P_j, P_k | l_i = \alpha, l_j = \beta, l_k = \gamma)$$

$$= \text{Prob}(P_i | l_i = \alpha)\ \text{Prob}(P_j | l_j = \beta)\ \text{Prob}(P_k | l_k = \gamma)$$

$$= \frac{\text{Prob}(l_i = \alpha | P_i)\ \text{Prob}(P_i)}{\text{Prob}(l_i = \alpha)} \cdot \frac{\text{Prob}(l_j = \beta | P_j)\ \text{Prob}(P_j)}{\text{Prob}(l_j = \beta)} \cdot \frac{\text{Prob}(l_k = \gamma | P_k)\ \text{Prob}(P_k)}{\text{Prob}(l_k = \gamma)}$$

$$= \frac{P_i(l_i = \alpha) P_j(l_j = \beta) P_k(l_k = \gamma)\ \text{Prob}(P_i)\ \text{Prob}(P_j)\ \text{Prob}(P_k)}{\text{Prob}(l_i = \alpha)\ \text{Prob}(l_j = \beta)\ \text{Prob}(l_k = \gamma)}.$$

The meaning of this assumption is that the probability vector P_i is directly dependent only on l_i, and once l_i is given, the probability of the vector being P_i

TABLE 1

The Words Having Maximal Merit (See Text) for the Three Graphs in Fig. 1, with Their Relative Merits

(a)		(b)		(c)	
LET	506.19	EVEN	60.99	BOOK	6.56
HT	72.62	BEEN	22.86	HOOK	5.33
ELT	53.67	HEEN	14.53	BUSH	1.48
ECT	36.64	EVERS	14.47	HUSH	1.08
EST	34.28	BLEN	5.66	EMAK	0.97

is independent of $P_j, j \neq i$, and of $l_j, j \neq i$. Under this assumption, (1) becomes

$$P_{ijk}(l_i = \alpha, l_j = \beta, l_k = \gamma)$$

$$= P_i(l_i = \alpha)P_j(l_j = \beta)P_k(l_k = \gamma) \cdot \frac{\text{Prob}(l_i = \alpha, l_j = \beta, l_k = \gamma)}{\text{Prob}(l_i = \alpha)\ \text{Prob}(l_j = \beta)\ \text{Prob}(l_k = \gamma)} \cdot \frac{\text{Prob}(P_i)\ \text{Prob}(P_j)\ \text{Prob}(P_k)}{\text{Prob}(P_i, P_j, P_k)}. \quad (2)$$

From now on, we will denote

$$\frac{\text{Prob}(l_i = \alpha, l_j = \beta, l_k = \gamma)}{\text{Prob}(l_i = \alpha)\ \text{Prob}(l_j = \beta)\ \text{Prob}(l_k = \gamma)}$$

by $r_{ijk}(\alpha, \beta, \gamma)$. The quantities $r_{ijk}(\alpha, \beta, \gamma)$ are independent of the estimated distributions P_i. These quantities are computed in advance by using some model to find the required probabilities. See [6] for an approach to computing these coefficients in the absence of such models. Now

$$P_{ijk}(l_i = \alpha) = \sum_{\beta \in \Delta} \sum_{\gamma \in \Delta} P_{ijk}(l_i = \alpha, l_j = \beta, l_k = \gamma),$$

and

$$\sum_{\alpha \in \Delta} \sum_{\beta \in \Delta} \sum_{\gamma \in \Delta} P_{ijk}(l_i = \alpha, l_j = \beta, l_k = \gamma) = 1.$$

We thus have

$$P_{ijk}(l_i = \alpha) = \frac{\sum_{\beta \in \Delta} \sum_{\gamma \in \Delta} P_{ijk}(l_i = \alpha, l_j = \beta, l_k = \gamma)}{\sum_{\lambda \in \Delta} \sum_{\beta \in \Delta} \sum_{\gamma \in \Delta} P_{ijk}(l_i = \lambda, l_j = \beta, l_k = \gamma)}$$

$$= \frac{P_i(l_i = \alpha) \cdot \sum_{\beta \in \Delta} \sum_{\gamma \in \Delta} P_j(l_j = \beta)P_k(l_k = \gamma)r_{ijk}(\alpha, \beta, \gamma)}{\sum_{\lambda \in \Delta} P_i(l_i = \lambda) \cdot \sum_{\beta \in \Delta} \sum_{\gamma \in \Delta} P_j(l_j = \beta)P_k(l_k = \gamma)r_{ijk}(\lambda, \beta, \gamma)}, \quad (3)$$

since the factor

$$\frac{\text{Prob}(P_i)\ \text{Prob}(P_j)\ \text{Prob}(P_k)}{\text{Prob}(P_i,\ P_j,\ P_k)}$$

cancels out.

Note that $P_{ijk}(l_i = \alpha)$ is dependent on the nodes v_i, v_j, and v_k. In a similar approach, rules for any number of nodes can be derived. A rule using two nodes, rather than three as is done here, can be found in [6].

The Coefficients

This paper handles problems from the domain of English text, and a node in the graph will represent a segment of the handwriting. The relations that will be used are the probabilities of certain combinations of letters appearing in English text. Only trigrams will be used; the event $(l_i = \alpha, l_j = \beta, l_k = \gamma)$ means that nodes v_i, v_j, and v_k represent the sequence $\beta\alpha\gamma$. The a priori probability $\text{Prob}(l_i = \alpha, l_j = \beta, l_k = \gamma)$ is equal to the probability that a randomly chosen trigram from an English text is $\beta\alpha\gamma$, and $\text{Prob}(l_i = \alpha)$ is equal to the probability that a randomly chosen letter from English text will be α. The above probabilities can be estimated from a long sample of English text. They were computed from the novel *Wuthering Heights* by Emily Bronte, which contains approximately one million letters. The coefficients $r_{ijk}(\beta, \alpha, \gamma)$ used in the updating rule are then

$$r_{ijk}(\beta, \alpha, \gamma) = \frac{\text{Prob}(\beta\alpha\gamma)}{\text{Prob}(\beta)\ \text{Prob}(\alpha)\ \text{Prob}(\gamma)}$$

where v_j represents a letter preceding v_i, v_k represents a letter following v_i, $\text{Prob}(\beta\alpha\gamma)$ is the a priori probability of a randomly chosen trigram being $\beta\alpha\gamma$, and $\text{Prob}(\alpha)$ is the probability of a randomly chosen letter being α.

4. AMBIGUITY REDUCTION

We use the relaxation updating rule of the preceding section to reduce the ambiguity at every node of the graph $G = (V, E)$ described in Section 2.

A node $v_i \in V$, having m predecessors and n successors, can participate locally in $n \times m$ paths from the initial node to the final node. These paths correspond to the fact that if v_i is on a path from the initial node to the final node, only one of its predecessors and one of its successors will also be on that path. When updating the probability at each node, the assumption is that the node is on such a path. In cases where the node cannot be on such a path, e.g., a node different from the initial node but with no predecessors, the node is deleted from the graph.

Let $P_{ijk}^{(n+1)}(l_i = \alpha)$ be the updated probability computed using (3) from the probabilities $P_i^{(n)}$ of v_i, $P_j^{(n)}$ of v_j (a predecessor of v_i), and $P_k^{(n)}$ of v_k (a successor of v_i). Since the graph actually represents only one word, only one path is required. Thus, we give every letter a probability proportional to the highest estimate it

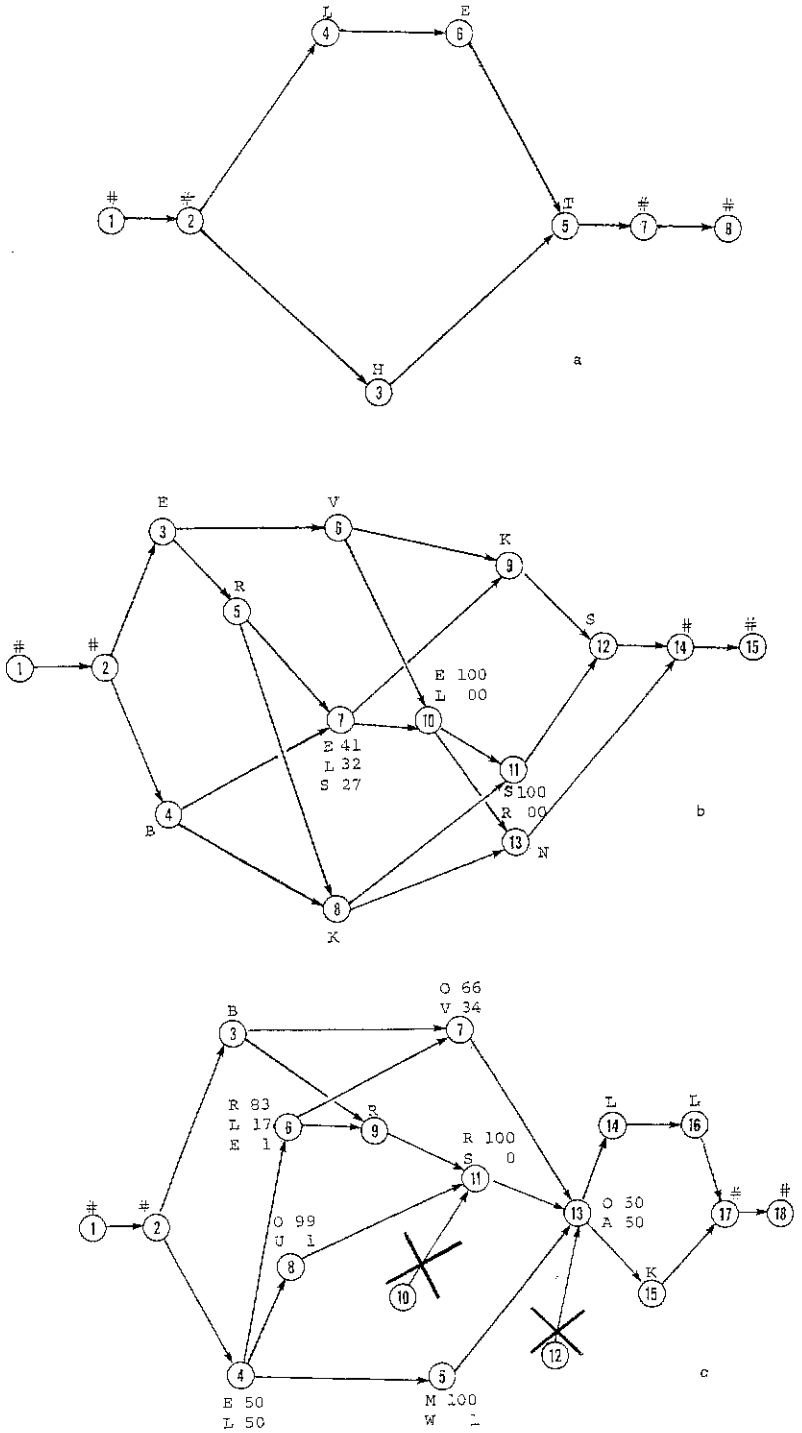


FIG. 2. The graphs of Fig. 1 after several iterations of relaxation. (a) "LET" after five iterations; two words remain possible. (b) "EVEN" after 24 iterations; 55 words remain possible. (c) "BOOK" after 18 iterations; 160 words remain possible.

TABLE 2

Same as Table 1, But for the Graphs in Fig. 2.

(a)		(b)		(c)	
LET	2931.50	EVEN	1492.20	BOOK	279.37
HT	152.74	BEEN	1283.40	EMAK	9.66
		EVESS	297.16	BOAK	8.97
		BLEN	272.79	BOOLL	7.04
		BLESS	149.52	BOALL	5.99

gets from all $n \times m$ combinations:

$$P_i^{(n+1)}(l_i = \alpha) \cong \max_{j,k} \{P_{ijk}^{(n+1)}(l_i = \alpha)\}.$$

The values for $P_i^{(n+1)}$ are normalized to give a legal probability vector.

A node (different from the initial or final node) will be deleted from the graph when it has no predecessors, no successors, or when for all α,

$$\max_{j,k} \{P_{ijk}^{(n+1)}(l_i = \alpha)\} = 0.$$

The last case occurs when no possible letter in v_i makes a legal trigram, in combination with any possible letters from one of its predecessors and one of its successors.

Figure 2 shows the graphs of Figure 1 after several iterations of relaxation. The number of possible words was reduced from 3960 to 160 in 18 iterations for the handwritten version of "book," from 838 to 55 in 24 iterations for "even," and from 13 to 2 in 6 iterations for "let." Additional iterations had almost no effect on the labeling. The reduction in ambiguity is greater for more complicated graphs, where individual reductions of ambiguity at individual nodes have stronger effects on the total number of possible words.

Table 2 shows the five words having highest merit value from the graphs of Fig. 2. In all cases the word with highest merit in Table 1 remained the highest in Table 2.

5. CONCLUDING REMARKS

It has been shown in this paper that the probabilistic relaxation updating can be used to significantly reduce the ambiguity at each node of the graph representing the ambiguous segmentation. Methods are available in some cases to find the maximal-merit word in linear time [5], but these methods find only one word. Using relaxation, all the possible segmentations are preserved, and an alternative segmentation can be used if the maximal word is found to be incorrect in the given context.

REFERENCES

1. K. Hayes, *Reading Handwritten Words Using Hierarchical Relaxation*, Ph.D. thesis, Computer Science Center, University of Maryland, 1979.

2. A. Rosenfeld, R. A. Hummel, and S. W. Zucker, Scene labeling by relaxation operations, *IEEE Trans. Systems, Man and Cybernetics* **SMC-6**, 1976, 420–433.
3. A. Rosenfeld, Iterative methods in image analysis, *Pattern Recognition* **10**, 1978, 181–187.
4. F. R. D. Velasco and A. Rosenfeld, The application of relaxation to waveforms with ambiguous segmentation, TR-683, Computer Science Center, University of Maryland, August 1978.
5. S. Peleg, Maximal derivations for probabilistic strings in stochastic languages, TR-684, Computer Science Center, University of Maryland, July 1978.
6. S. Peleg, A new probabilistic relaxation scheme, TR-711, Computer Science Center, University of Maryland, November 1978. Proc. IEEE Conf. on Pattern Recognition and Image Processing, August 1979, to appear.
7. S. Peleg and A. Rosenfeld, Breaking substitution ciphers using a relaxation algorithm, TR-721, Computer Science Center, University of Maryland, January 1979.