NOISY SUBSEQUENCE RECOGNITION USING CONSTRAINED STRING EDITING INVOLVING ARBITRARY OPERATIONS*

B. J. Oommen[1] and R. K. S. Loke

ABSTRACT

We consider a problem which can greatly enhance the areas of cursive script recognition and the recognition of printed character sequences. This problem involves recognizing words/strings by processing their noisy subsequences. Let X^* be any unknown word from a finite dictionary **H**. Let U be any arbitrary *subsequence* of X^*. We study the problem of estimating X^* by processing Y, a noisy version of U. Y contains substitution, insertion, deletion *and generalized transposition* errors -- the latter occurring when transposed characters are themselves subsequently substituted, as is typical in cursive and typewritten script, in molecular biology and in noisy chain-coded boundaries. We do this by defining the constrained edit distance between $X \in$ **H** and Y subject to any arbitrary edit constraint involving the number and type of edit operations to be performed. In this paper we present the first reported solution to the analytic problem of achieving constrained editing of one string to another using these four edit operations. An algorithm to compute this constrained edit distance has been presented. Using these algorithms we present a syntactic Pattern Recognition (PR) scheme which corrects noisy text containing all these types of errors. Experimental results which involve strings of lengths between 40 and 80 with an average of 30.24 deleted characters and an overall average noise of 68.69 % demonstrate the superiority of our system over existing methods.

I. INTRODUCTION

In this paper we consider the syntactic pattern recognition of noisy sunsequences which have been garbled by the introduction of Subsitution, Insertion, Deletion (SID) and Generalized Transpositions (GT) errors.

A common problem in syntactic pattern recogniton is that of correcting errors in a string. A package solving this problem is typically used in the recognition of printed character sequences and cursive script (and more recently, even closed boundaries [MV93]) *after* the individual symbols of the "alphabet" have been hypothesized using statistical methods. A generalization of this involves recognizing a string by processing one of its noisy *substrings* [KO83a]. Such a scheme would permit us to recognize strings and sequences by using only noisy partial (occluded) information.

In this paper we consider a far more general problem. To view the problem in its generality, we pose it in an arbitrary communication setting. Let us assume that a sender intends to transmit a string

*Partially supported by the Natural Sciences and Engineering Research Council of Canada.

[1]Senior Member, IEEE. Both authors can be contacted at *School of Computer Science, Carleton University,Ottawa ; CANADA : K1S 5B6.* E-mail addess of first author : oommen@scs.carleton.ca.

$X^* \in H$. However, rather than send the entire string X^* he chooses to (randomly or otherwise) delete characters from it, and merely transmit U, one of its *subsequences*. U is transmitted through a noisy channel and is further subjected to SID and *generalized transposition* errors. The receiver receives Y, the garbled form of U. We intend to recognize X^* by merely processing Y. The reader will be able to comprehend the difficulty in the PR problem if he observes that any substring consisting of the consecutive characters of U, need not be a contiguous substring of X^*, and that whereas there are $O(N^2)$ contiguous substrings for a string X of length N, there are $O(2^N)$ subsequences.

To clarify situations, we shall consider an example from one of the experiments we have performed. Let us suppose X^* is the following string:

X^* = sincetheadventofthedigitalcomputertherehasbeenaconstante

Notice that the $|X^*|$ is 55. Suppose the sender deletes portions of X^* to form the string U, where,

U = sincetdventofthtalcompuherehaconte.

Observe that in this case U is a subsequence of X^* in which 22 of the 55 characters have been deleted. The resultant string U is transmitted through a noisy channel. The received string Y contains additional substitution, deletion, insertion and *generalized transposition* errors and is received as

Y = sincwtdvetnohfhatzlcopmuheerhaocnte

In this case, between X^* and Y there is an overall number of at least 31 errors. Our aim is to recognize X^* by processing Y.

Clearly, a syntactic package which achieves this will greatly enhance the power of OCR systems which recognize printed character sequences and cursive script. Indeed, we will then be able to recognize strings and sequences, by merely having as an input a noisy versiuon of *any* of its subsequences. This could permit the recognition of strings and sequences which have been "occluded" at multiple junctures, and thus permit far more noisy environments.

To achieve this we present the first reported solution to the analytic problem of editing (or more precisely, aligning) one string to another using these four edit operations subject to any arbitrary constraint involving the number and type of these operations. Using these algorithms we present a syntactic PR scheme which corrects the noisy subsequences containing all these types of errors. Indeed, the technique which we have proposed in this paper recognizes the above string in our experimental set-up described later.

Besides having applications in file editing, the noisy subsequence matching problem has also potential applications in the area of information retrieval. Consider a data base which consists of many records each identified by distinct keywords. Suppose that the system manager permits the user to access a record by merely referring to any arbitrary subsequence of the keyword. In such a case the program that solves the noisy subsequence matching problem can be used to process a noisy (incorrectly spelled) version of any subsequence of the keyword, and locate a record characterized by the entire keyword. Apart from text processing, noisy string (substring and subsequence) correction has also applications in image processing where the boundary of the image to be recognized is syntactically coded as a string [MV93]. Inexact sequence comparison is also used extensively in the comparison of biological macromolecules where "noisy" version of proteins strings are compared with their exact forms typically stored in large protein databases. The literature on sequence correction is extensive; indeed, the reviews [HD80, Pe80] list hundreds of publications that tackle the problem from various perspectives.

Damareau [See references in Oo87a,b] was probably the first researcher to observe that most of the errors that were found in strings were either a single Substitution, Insertion and Deletion (SID) or reversal (transposition) error. Thus the question of computing the dissimilarity between strings was reduced to comparing them using these edit transformations. In most of the existing literature the transposition operation has been modelled as a sequence of a single insertion and deletion.

The first breakthrough in comparing strings using the three elementary edit transformations was the concept of the Levenshtein metric introduced in coding theory [Le66], and its computation. The Levenshtein distance between two strings is defined as the minimum number of edit operations required to transform one string into another. Many researchers among whom are Wagner and Fischer [WF74] generalized it by using edit distances which are symbol dependent. The latter distance is termed as the Generalized Levenshtein Distance (GLD). One of the advantages of the GLD is that it is a metric if the inter-symbol distances obey triangular inequality constraints [OTK76,SK83]. Wagner and Fischer [WF74] also proposed an efficient algorithm for computing this distance by utilizing the concepts of dynamic programming. This algorithm is optimal for the infinite alphabet case. Various amazingly similar versions of the algorithm are available in the literature, a review of which can be found in [SK83]. Masek and Paterson [MP80] improved the algorithm for the finite alphabet case and Ukkonen [Uk85] designed solutions for cases involving various other inter-substring edit operations. Related to these algorithms are the ones used to compute the Longest Common Subsequence (LCS) of two strings (Hirschberg [Hi77], Hunt and Szymanski [HS77] and others [SK83]). String correction using the GLD as a criterion has been done for noisy strings [Pe80,SK83], substrings [KO83c], and subsequences [Oo87b] and also for strings in which the dictionaries are treated as generalized tries [KO81] and grammars [SK83]. Besides these, various probabilistic methods have been studied in the literature [Pe80,KO84].

Although some work has been done to extend the traditional set of SID operations to include the transposition of adjacent characters [LW75,SK83] the problem for constrained editing is unsolved for "Generalized" Transposition (GT) errors. The difference between the the latter operations and those traditionally considered as "transpositions" is the following. Currently, transpositions merely imply errors caused when the order of adjacent symbols is reversed. Such an error could cause the string "develop" to be mutated into "dveelop". As opposed to this, *Generalized* Transpositions (GTs) permit these transposed symbols to be subsequently substituted. Thus, if one was working on a typewriter keyboard, this could cause the string "develop" to be mutated into "dbrelop" -- which would arise when the typist inherently "reversed" the two characters ("ev") due to the sequence in which the fingers touched the keyboard, but also accidentally shifted his/her hands to the right of the keyboard one key too far -- which happens all too often. Clearly, GT errors can be represented as a sequence of two substitutions ('e' $\rightarrow$ 'b', and 'v' $\rightarrow$ 'r'). We shall show that the recognition accuracies obtained by representing them as GTs is *much* more than by representing them as two substitutions.

This "new" GT operation is not only applicable in the recognition of typewritten and cursive script, but also has vast potential application in processing of chain-coded images [MV93] and biological macromolecules. To see the latter, consider the representation of the boundary of a handwritten cursive "2". A study of various boundaries shows that the "hook" at the top of "2" varies with the writer --some "hooks" being more curved than others. A less curved "hook" can have a "0101" chain-coded representation, which is equivalent to "1010" when the symbols are transposed. As opposed to this, a more curved "hook" can have the code "6710", which is precisely edited from "0101" by two GTs, where the symbols of one of the transpositions has been subsequently substituted. Indeed, such scenarios are numerous in boundary representations. GT errors are also encountered in the study of biological macromolecules [SK83] where the mutation (substitution) of transposed molecules occurs in the "next" generation after the protein sequences are transposed.

As mentioned earlier, we shall first consider the analytic problem of editing one string X to another, Y using these four edit operations subject to any arbitrary constraint. This edit constraint can be arbitrarily complex, so long as it is specified in terms of the number and type of the edit operations to be included in the optimal edit transformation. For the sake of clarity, we give below some examples of constrained editing.

EXAMPLE 1. Here are some typical constrained editing problems :

(a) What is the optimal way of editing X to Y using no more than k insertions?

(b) How can we optimally transform X to Y using exactly k transpositions?

(c) Is it possible to transform X to Y using exactly k_t generalized transpositions, k_s substitutions, k_i insertions and k_e deletions? If it is possible, what is the distance between X and Y subject to this constraints?

I.1 Notation

A is a finite alphabet, and A^* is the set of strings over **A**. θ is the null symbol, where $\theta \notin \mathbf{A}$, and is distinct from μ the empty string. Let $\tilde{\mathbf{A}} = \mathbf{A} \cup \{\theta\}$. A string $X \in A^*$ of the form $X = x_1 \ldots x_N$, where each $x_i \in \mathbf{A}$, and is said to be of length $|X| = N$. Its prefix of length i will be written as X_i, for $1 \le i \le N$. Uppercase symbols represent strings, and lower case symbols, elements of the alphabet.

Let Z' be any element in $\tilde{\mathbf{A}}^*$, the set of strings over $\tilde{\mathbf{A}}$. The *Compression Operator* $\mathfrak{C}$ is a mapping from $\tilde{\mathbf{A}}^*$ to $\mathbf{A}^*$: $\mathfrak{C}(Z')$ is Z' with all occurrences of the symbol θ removed from Z'. Note that $\mathfrak{C}$ preserves the order of the non-θ symbols in Z'. For example, if $Z'= f\theta o\theta r$, $\mathfrak{C}(Z')$ = for.

We now define the costs associated with the individual edit operations. If $\mathbf{R}^+$ is the set of nonnegative real numbers, we define the elementary edit distances using four elementary functions $d_s(.,.)$, $d_i(.)$, $d_e(.)$, $d_t(.,.)$ defined as :

(i) $d_s(.,.)$ is a map from $\mathbf{A} \times \mathbf{A} \rightarrow \mathbf{R}^+$ and is the Substitution Map. $d_s(a,b)$ is the distance associated with substituting b for a, $a,b \in \mathbf{A}$. Although not mandatory, $d_s(a,a)$ is generally set to zero.

(ii) $d_i(.)$ is a map from $\mathbf{A} \rightarrow \mathbf{R}^+$ and is called the Insertion Map. The quantity $d_i(a)$ is the distance associated with inserting the symbol $a \in \mathbf{A}$.

(iii) $d_e(.)$ is a map from $\mathbf{A} \rightarrow \mathbf{R}^+$ and is called the Deletion or Erasure Map. The quantity $d_e(a)$ is the distance associated with deleting (or erasing) the symbol $a \in \mathbf{A}$.

(iv) $d_t(.,.)$ is a map from $\mathbf{A}^2 \times \mathbf{A}^2 \rightarrow \mathbf{R}^+$ called the Transposition Map. The quantity $d_t(ab,cd)$ is the distance associated with transposing the string "ab" into "cd". This can be thought of as a "serial" operation: "ab" is first transposed to "ba" and subsequently the individual characters are substituted.

I.2 The Set of Edit Possibilities : $\Gamma_{X,Y}$

For every pair (X,Y), $X, Y \in \mathbf{A}^*$, the finite set $\Gamma_{X,Y}$ is defined by means of the compression operator $\mathfrak{C}$, as a subset of $\tilde{\mathbf{A}}^* \times \tilde{\mathbf{A}}^*$ as :

$\Gamma_{X,Y} = \Big\{ (X',Y') \;\Big|\; (X',Y') \in \tilde{\mathbf{A}}^* \times \tilde{\mathbf{A}}^*$, and each (X',Y') obeys

(i) $\mathfrak{C}(X') = X$, $\mathfrak{C}(Y') = Y$,

(ii) $|X'| = |Y'|$,

(iii) For all $1 \leq i \leq |X'|$, it is not the case that $x_i' = y_i' = \theta \Big\}$. (1)

By definition, if $(X',Y') \in \Gamma_{X,Y}$ then $\text{Max}\big[\, |X|, |Y| \,\big] \leq |X'| = |Y'| \leq |X| + |Y|$.

The set $\Gamma_{X,Y}$ describes order-preserving mappings and its cardinality is :

$$|\Gamma_{X,Y}| = \sum_{(k=\max[0,|Y| - |X|])}^{|Y|} \frac{(|X| + k)!}{k!(|Y|-k)!(|X|-|Y|+k)!}$$

Each element in $\Gamma_{X,Y}$ corresponds to one way of editing X into Y, using the SID operations. The edit operations themselves are specified for all $1 \leq i \leq |X'|$ by (x_i', y_i'), which represents the transformation of x_i' to y_i'. The cases below consider the SID operations :

(i) If $x_i' \in \mathbf{A}$ and $y_i' \in \mathbf{A}$, it represents the substitution of y_i' for x_i'.

(ii) If $x_i' \in \mathbf{A}$ and $y_i' = \theta$, it represents the deletion of x_i'.

(iii) If $x_i' = \theta$ and $y_i' \in \mathbf{A}$, it represents the insertion of y_i'.

However, on examining the individual elements of $\Gamma_{X,Y}$ it becomes clear that each pair contains more information than that. Indeed, *in each pair*, there is also information about the various ways by which

X can be edited to Y even if the set of edit operations is grown so as to include GTs. Thus, when (X',Y') = (abθ, cde), apart from the operations described above, the pair also represents the GT of ‘ab’ to ‘cd’ and the insertion of ‘e’. Observe that the transformation of a symbol a $\in$ **A** to itself is also considered as an operation in the arbitrary pair (X',Y') $\in \Gamma_{X,Y}$. Also note that the same set of edit operations can be represented by multiple elements in $\Gamma_{X,Y}$. This duplication serves as a powerful tool in proofs of various analytic results [KO81, KO83a, KO83b, KO84, Oo87a, Oo87b].

EXAMPLE 2. Let X = me and Y = on. Then,

$$\Gamma_{X,Y} = \left\{(\mathrm{me},\mathrm{on}),(\theta\mathrm{me},\mathrm{o}\theta\mathrm{n}),(\theta\mathrm{me},\mathrm{on}\theta),(\mathrm{m}\theta\mathrm{e},\theta\mathrm{on}),(\mathrm{m}\theta\mathrm{e},\mathrm{on}\theta),(\mathrm{me}\theta,\theta\mathrm{on}),(\mathrm{me}\theta,\mathrm{o}\theta\mathrm{n}),\ldots\right\}.$$

In particular the pair (me,on) represents the edit operations of substituting the 'm' and ‘e’ with ‘o’ and ‘n’ respectively. It also represents the transposition from ‘me’ to ‘on’. ♦♦♦

Since the edit distance between X and Y is the minimum of the sum of the edit distances associated with operations required to change X to Y, this distance, D(X,Y), has the expression :

$$D(X,Y) = \underset{((X',Y')\in \Gamma_{X,Y})}{\mathrm{Min}} \sum_{i=1}^{|J'|} \left[\text{ Distances Associated with the Operations in } (X', Y')\right] \qquad (2)$$

where, (X',Y') represents J' possible edit operations.

II. EDIT CONSTRAINTS

II.1 Permissible and Feasible Constraints and Constraint Specification

Consider the problem of editing X to Y, where |X| = N and |Y| = M. Suppose we edit a prefix of X into a prefix of Y, using exactly *i* insertions, *e* deletions (or erasures), *s* substitution and *t* GTs. Since the numbers of edit operations are specified, this corresponds to editing $X_{e+s+2t} = x_1 \ldots x_{e+s+2t}$ the prefix of X of length *e+s+2t*, into $Y_{i+s+2t} = y_1 \ldots y_{i+s+2t}$, the prefix of Y of length *i+s+2t*.

To obtain bounds on the magnitude of the variables *i, e, s* and *t,* we observe that they are constrained by the lengths of the strings X and Y. Thus, if r = *e* + *s* + *2t*, q = *i* + *s* + *2t* and R = Min [M, N], these variables will have to obey the following obvious constraints:

$$0 \le t \le \mathrm{Min}\left[\left\lfloor \tfrac{N}{2}\right\rfloor,\left\lfloor \tfrac{M}{2}\right\rfloor\right]\ ;\quad \mathrm{Max}[0, M-N] \le i \le q \le M\ ;\ \ 0 \le e \le r \le N\ ;\ \ 0 \le s \le \mathrm{Min}[N, M].$$

Values of quadruples (*i, e, s, t*) which satisfy these constraints are termed *feasible*. Let

$$H_t = \left\{ j \mid 0 \le j \le \mathrm{Min}\left[\left\lfloor \tfrac{N}{2}\right\rfloor,\left\lfloor \tfrac{M}{2}\right\rfloor\right] \right\}\ ; \qquad H_i = \{ j \mid \mathrm{Max}[0, M-N] \le j \le M \}\ ;$$

$$H_e = \{ j \mid 0 \le j \le N \}\ ; \qquad H_s = \{ j \mid 0 \le j \le \mathrm{Min}[N, M] \}. \qquad (3)$$

H_t, H_i, H_e and H_s are called the set of *permissible* values of *i, e, s* and *t.* A quadruple (*i, e, s, t*) is feasible if apart from $t \in H_t$, $i \in H_i$, $e \in H_e$ and $s \in H_s$, the following is satisfied:

$$i + s + 2t \le M \qquad \text{and} \qquad e + s + 2t \le N. \qquad (4)$$

The following theorem specifies the permitted forms of the feasible quadruples which are encountered in editing X_r, the prefix of X of length r, to Y_q, the prefix of Y of length q.

THEOREM 1. To edit X_r, the prefix of X of length r, to Y_q, the prefix of Y of length q, the set of feasible quadruples is given by

$$\{ (i, r\text{-}q+i, q\text{-}i\text{-}2t, t) \mid \text{Max}[0, q\text{-}r] \leq i \leq q\text{-}2t \}$$

Proof : Consider the constraints imposed on the feasible values of i, e, s and t. Since we are interested in the editing of X_r to Y_q, we have to consider only those quadruples (i, e, s, t) satisfying

$$e + s + 2t = r \text{ and } i + s + 2t = q.$$

Given the number of transpositions, the number of insertions can take any value from max[0, q-r] to q-2t. So, for every value of i in this range, the feasible quadruple (i, e, s, t) must have exactly q-i-2t substitution. Similarly, since the sum of the number of substitutions, deletions and transpositions (which account for two times the number of symbols involved) is r, the quadruple (i, e, s, t) must have exactly r-q+i deletions. Hence the theorem. ♦♦♦

An edit constraint is specified in terms of the number and type of edit operations that are required in the process of transforming X to Y. It is expressed by formulating the number and type of edit operations in terms of four sets Q_t, Q_i, Q_e and Q_s, which are subsets of the sets H_t, H_i, H_e and H_s defined in (3). We clarify this using the three constraints given in Example 1.

EXAMPLE 3.

(a) To edit X to Y performing no more than k insertions, the sets Q_s, Q_e and Q_t are all equal to the null set. Further,

$$Q_i = \{ j \mid j \in H_i, j \leq k \}.$$

(b) To edit X to Y performing exactly k GTs, Q_i, Q_e and Q_s would all be null. Furthermore, $Q_t = \{ k \} \cap H_t$. Note that if k is not in H_t, the problem is infeasible.

(c) To edit X to Y performing exactly k_t transpositions, k_s substitutions, k_i insertions and k_e deletions yields

$$Q_t = \{ k_t \} \cap H_t,\ Q_i = \{ k_i \} \cap H_i,\ Q_e = \{ k_e \} \cap H_e \text{ and } Q_s = \{ k_s \} \cap H_s.$$

Note that the problem is infeasible unless $k_t \in H_t$, $k_i \in H_i$, $k_e \in H_e$ and $k_s \in H_s$.

For every value of t in the set Q_t, we define the sets Q_i^t, Q_e^t and Q_s^t as. :

$$Q_i^t = \{ i \mid i \leq M\text{-}2t \} \cap Q_i, ;\ Q_e^t = \{ e \mid e \leq N\text{-}2t \} \cap Q_e, ;\ Q_s^t = \{ s \mid s \leq \text{Min}[N\text{-}2t, M\text{-}2t] \} \cap Q_s.$$

These sets represent the number of edit operations *given that t* GTs had occurred.

THEOREM 2. Given a value of t, every edit constraint specified for the process of editing X to Y can be written as a unique subset of H_i.

Proof : Let the constraints be specified by Q_i^t, Q_e^t and Q_s^t.

Every element $j \in Q_e^t$ requires exactly j deletions to be performed. From Theorem 1, since $|X| = N$, this requires that the number of substitutions be $N - j$. Further, since $|Y| = M$, the number of insertions is going to be $M - N + j$.

Similarly, if $j \in Q_s^t$, the edit transformation must contain exactly j substitutions. Since $|X| = N$ and $|Y| = M$, Theorem 1 requires that $M - j$ insertions and $N - j$ deletions to be performed.

Let

$$Q_e^{t*} = \{ M-N+j \mid j \in Q_e^t \} \text{ and } Q_s^{t*} = \{ M-j \mid j \in Q_s^t \}.$$

Clearly, for any arbitrary constraint, the number of insertions that are permitted, for a given value of t, is given by the set of integers which is obtained by intersecting Q_i^t, Q_e^{t*} and Q_s^{t*}, which is obviously a subset of H_i. A formal definition of the set is given below.

$$Q_t^* = Q_i^t \cap Q_e^{t*} \cap Q_s^{t*} \text{ for every value of } t \text{ in } Q_t.$$ ♦♦♦

The set referred to above, which describes the constraint and which is the subset of H_i given a value of t will in future be written as T_t. The example below will help clarify issues.

EXAMPLE 4.

Let X = "for" and Y = "far". Suppose we want to transform X to Y by performing at most 2 transpositions, at least 1 insertion, at most 1 substitution and exactly 1 deletion. Then,

$Q_t = \{0,1\}$, $Q_i = \{1,2,3\}$, $Q_e = \{1\}$ and $Q_s = \{0,1\}$.

Hence,

$Q_i^0 = \{1,2,3\}$, $Q_e^0 = \{1\}$ and $Q_s^0 = \{0,1\}$;

$Q_i^1 = \{1\}$, $Q_e^1 = \{1\}$ and $Q_s^1 = \{0,1\}$;

and also,

$Q_e^{0*} = \varnothing$ and $Q_s^{0*} = \{2,3\}$;

$Q_e^{1*} = \{1\}$ and $Q_s^{1*} = \{1\}$.

Therefore,

$T_1 = \{1\}$ and $T_0 = \varnothing$.

Note that the problem is *infeasible* when there are no transpositions.

The optimal transformations must therefore contain exactly one insertion when there is one transposition. Some candidate edit transformations are given by the following subset of $\Gamma_{X,Y}$:

{ (θfor,fθar), (θfor,farθ), (fθor,θfar), (foθr,farθ), (forθ,θfar), (forθ,faθr) }

Note that every pair in the above subset corresponds to at most 2 transpositions, at least 1 insertion, at most 1 substitution and exactly 1 deletion. ♦♦♦

We refer to the edit distance subject to the constraint T_t as D_t (X,Y). By definition, if $T_t = \emptyset$, then D_t (X,Y) = ∞. The distance for the optimal edit transformations will be denoted by D_c(X,Y) which is the minimum of all D_t (X,Y). We now consider the computation of D_t (X,Y) and D_c(X,Y).

III. W: The Array Of Constrained Edit Distances

Let W(*i, e, s, t*) be the constrained edit distance associated with editing X_{e+s+2t} to Y_{i+s+2t} subject to the constraint that exactly *i* insertions, *e* deletions, *s* substitutions and *t* GTs are performed in the process of editing. As before, let $r = e+s+2t$ and $q = i+s+2t$. Let $\Gamma_{i,e,s,t}$(X,Y) be the subset of the pairs in Γ_{X_r, Y_q} in which every pair corresponds to *i* insertions, *e* deletions, *s* substitutions and *t* transpositions. Since we shall always be referring to the strings X and Y, we refer to this set as $\Gamma_{i,e,s,t}$. Thus, using the notation of (1), if (*i, e, s, t*) is feasible for the problem, W(*i, e, s, t*) has the expression:

$$W(i, e, s, t) = \underset{((X'_r, Y'_q) \in \Gamma_{i,e,s,t})}{\text{Min}} \sum_{j=1}^{|X'_r|} \Big[\text{ Distances Associated with the Operations in } (X'_r, Y'_q) \Big] \quad (5)$$

where, (X'_r, Y'_q) represents J' possible edit operations.

We shall derive the recursively computable properties of the array W(*i, e, s, t*).

THEOREM 3. Let W(*i, e, s, t*) be the quantity defined in (5) for any two strings X and Y. Then,

$$\begin{aligned} W(i, e, s, t) = \textbf{Min} \ \Big[\ & \{ W(i\text{-}1, e, s, t) + d_i(y_{i+s+2t}) \}, \\ & \{ W(i, e\text{-}1, s, t) + d_e(x_{e+s+2t}) \}, \\ & \{ W(i, e, s\text{-}1, t) + d_s(x_{e+s+2t}, y_{i+s+2t}) \}, \\ & \{ W(i, e, s, t\text{-}1) + d_t(x_{e+s+2t-1}x_{e+s+2t}, y_{i+s+2t-1}y_{i+s+2t}) \} \Big] \end{aligned}$$

for all feasible quadruples (*i, e, s, t*).

Proof : The proof of the theorem is divided into four main cases :

(a). Any three of the four variables *i*, *e*, *s* and *t* are zero.

(b). Any two of the four variables *i*, *e*, *s* and *t* are zero.

(c). Any one of the four variables *i*, *e*, *s* and *t* are zero.

(d). None of the variables *i*, *e*, *s* and *t* are zero.

The most involved of these cases is case (d). Cases (a), (b) and (c) are merely one-, two- and three-parameter subcases respectively of the case (d). To avoid repetition, we prove only case (d). Thus for the rest of the proof, we encounter only strictly positive values for the variables *i*, *e*, *s* and *t*.

Let $r = e + s + 2t$ and $q = i + s + 2t$, $X_r = x_1 \ldots x_r$ and $Y_q = y_1 \ldots y_q$. By definition,

$$W(i, e, s, t) = \underset{((X'_r, Y'_q) \in \Gamma_{i,e,s,t})}{\text{min}} \sum_{j=1}^{|X'_r|} \Big[\text{ Distances Associated with the Operations in } (X'_{rj}, Y'_{qj}) \Big] \quad (5)$$

where, (X'_{rj}, Y'_{qj}) represents J' possible edit operations and is an arbitrary element of the set $\Gamma_{i,e,s,t}$, with x'_{rj} and y'_{qj} as the *j*th symbols of X'_r and Y'_q respectively. Let the lengths of X'_r and Y'_q in the arbitrary element be *L*. Then the last two symbols of X'_r and Y'_q are x'_{rL-1}, y'_{qL-1} and x'_{rL}, y'_{qL} respectively.

We now partition the set $\Gamma_{i,e,s,t}$ into nine *mutually exclusive and exhaustive* subsets:

$$\Gamma^1_{i,e,s,t} = \left\{ (X'_r, Y'_q) \,\middle|\, (X'_r, Y'_q) \in \Gamma_{i,e,s,t}, \text{ with } x'_{rL-1} = \theta,\ x'_{rL} = \theta,\ y'_{qL-1} = y_{q-1},\ y'_{qL} = y_q \right\},$$
$$\Gamma^2_{i,e,s,t} = \left\{ (X'_r, Y'_q) \,\middle|\, (X'_r, Y'_q) \in \Gamma_{i,e,s,t}, \text{ with } x'_{rL-1} = \theta,\ x'_{rL} = x_r,\ y'_{qL-1} = y_q,\ y'_{qL} = \theta \right\},$$
$$\Gamma^3_{i,e,s,t} = \left\{ (X'_r, Y'_q) \,\middle|\, (X'_r, Y'_q) \in \Gamma_{i,e,s,t}, \text{ with } x'_{rL-1} = \theta,\ x'_{rL} = x_r,\ y'_{qL-1} = y_{q-1},\ y'_{qL} = y_q \right\},$$
$$\Gamma^4_{i,e,s,t} = \left\{ (X'_r, Y'_q) \,\middle|\, (X'_r, Y'_q) \in \Gamma_{i,e,s,t}, \text{ with } x'_{rL-1} = x_r,\ x'_{rL} = \theta,\ y'_{qL-1} = \theta,\ y'_{qL} = y_q \right\},$$
$$\Gamma^5_{i,e,s,t} = \left\{ (X'_r, Y'_q) \,\middle|\, (X'_r, Y'_q) \in \Gamma_{i,e,s,t}, \text{ with } x'_{rL-1} = x_r,\ x'_{rL} = \theta,\ y'_{qL-1} = y_{q-1},\ y'_{qL} = y_q \right\},$$
$$\Gamma^6_{i,e,s,t} = \left\{ (X'_r, Y'_q) \,\middle|\, (X'_r, Y'_q) \in \Gamma_{i,e,s,t}, \text{ with } x'_{rL-1} = x_{r-1},\ x'_{rL} = x_r,\ y'_{qL-1} = \theta,\ y'_{qL} = \theta \right\},$$
$$\Gamma^7_{i,e,s,t} = \left\{ (X'_r, Y'_q) \,\middle|\, (X'_r, Y'_q) \in \Gamma_{i,e,s,t}, \text{ with } x'_{rL-1} = x_{r-1},\ x'_{rL} = x_r,\ y'_{qL-1} = \theta,\ y'_{qL} = y_q \right\},$$
$$\Gamma^8_{i,e,s,t} = \left\{ (X'_r, Y'_q) \,\middle|\, (X'_r, Y'_q) \in \Gamma_{i,e,s,t}, \text{ with } x'_{rL-1} = x_{r-1},\ x'_{rL} = x_r,\ y'_{qL-1} = y_q,\ y'_{qL} = \theta \right\},$$
$$\Gamma^9_{i,e,s,t} = \left\{ (X'_r, Y'_q) \,\middle|\, (X'_r, Y'_q) \in \Gamma_{i,e,s,t}, \text{ with } x'_{rL-1} = x_{r-1},\ x'_{rL} = x_r,\ y'_{qL-1} = y_{q-1},\ y'_{qL} = y_q \right\}.$$

Since the corresponding elements of (X'_r, Y'_q) cannot be θ simultaneously, it is clear that every pair in $\Gamma_{i,e,s,t}$ must be in one of the above sets. Hence these sets partition $\Gamma_{i,e,s,t}$. Rewriting (5) we obtain:

$$W(i, e, s, t) = \min_{1 \le k \le 9} \min_{((X'_r, Y'_q) \in \Gamma^k_{i,e,s,t})} \sum_{j=1}^{|X'_r|} \Big[\text{Distances Associated with the Operations in } (X'_{rj}, Y'_{qj})\Big] \quad (6)$$

where, (X'_{rj}, Y'_{qj}) represents J' possible edit operations

We shall now consider each of the nine terms in (6) individually.

Case 1.

Consider the first term in (6). In every pair in $\Gamma^1_{i,e,s,t}$ we know that the last two elements of each string in the pair are :

$$x'_{rL-1} = \theta,\ x'_{rL} = \theta,\ y'_{qL-1} = y_{q-1},\ y'_{qL} = y_q.$$

Hence,

$$\min_{((X'_r, Y'_q) \in \Gamma^1_{i,e,s,t})} \sum_{j=1}^{|X'_r|} \Big[\text{Distances Associated with Operations in } (X'_r, Y'_q)\Big]$$

$$= \min_{((X'_r, Y'_q) \in \Gamma^1_{i,e,s,t})} \sum_{j=1}^{|X'_r|} \Big[\text{Distances Associated with Operations in } (X'_{rL-1}, Y'_{qL-1})\Big] + d_i(y'_{qL}). \quad (7)$$

For every element in $\Gamma^1_{i,e,s,t}$ there is a unique element in $\Gamma_{i\text{-}1,e,s,t}$ and vice versa. By the inductive hypothesis the first term in (7) is exactly W(*i*-1, *e*, *s*, *t*). Since $r = e + s + 2t$ and $q = i + s + 2t$,

$$\min_{((X'_r, Y'_q) \in \Gamma^1_{i,e,s,t})} \sum_{j=1}^{|X'_r|} \left[\text{Distances Associated with Operations in } (X'_r, Y'_q)\right]$$

$$= W(i\text{-}1, e, s, t) + d_i(y_{i+s+2t}). \qquad (8)$$

In a similar way we consider the minimization for the rest of the cases separately. In each case we minimize the corresponding quantity inductively by utilizing the inductive hypothesis obtained by excluding the last elements of the pairs. In the interest of brevity (and to avoid repititive arguments) we omit the algebraic manipulations. This leads us to the following results :

Case 2. $W(i, e\text{-}1, s, t) + d_e(x_{e+s+2t})$

Case 3. $W(i, e, s\text{-}1, t) + d_s(x_{e+s+2t}, y_{i+s+2t})$.

Case 4. $W(i\text{-}1, e, s, t) + d_i(y_{i+s+2t})$.

Case 5. $W(i\text{-}1, e, s, t) + d_i(y_{i+s+2t})$.

Case 6. $W(i, e\text{-}1, s, t) + d_e(x_{e+s+2t})$.

Case 7. $W(i, e, s\text{-}1, t) + d_s(x_{e+s+2t}, y_{i+s+2t})$.

Case 8. $W(i, e\text{-}1, s, t) + d_e(x_{e+s+2t})$.

Case 9. The are two distinct possibilities for the minimization here.

Case 9.1 $W(i, e, s\text{-}1, t) + d_s(x_{e+s+2t}, y_{i+s+2t})$.

Case 9.2 $W(i, e, s, t\text{-}1) + d_t(x_{r-1}x_r, y_{q-1}y_q)$.

Combining the above nine cases proves the theorem. ♦♦♦

The computation of the distance D_t (X,Y) from the array W(*i*, *e*, *s*, *t*) only involves combining the appropriate elements of the array using T_t. This is proved in the following theorem.

THEOREM 4. The quantity D_t (X,Y) is related to the elements of the array W(*i*, *e*, *s*, *t*) as follows:

$$D_t\,(X,Y) = \min_{i \in T_t} W(i, N\text{-}M\text{+}i, M\text{-}i\text{-}2t, t).$$

Proof : The theorem follows directly from Theorem 1 with the substitution *r* = N and *q* = M. ♦♦♦

THEOREM 5. The distance for the optimal edit transformations D_c(X,Y), is obtained as follows :

$$D_c(X,Y) = \min_{k \in Q_t} D_k\,(X,Y)$$

Proof. Consider the individual D_t (X,Y) quantities. Each is the minimum edit distance associated with transforming X to Y with a feasible set of operations, given that there are *t* transpositions. The minimum of these would be the minimum edit distance for the optimal edit transformations. ♦♦♦

Using the above results we now propose a computational scheme for D_c(X,Y).

IV. THE COMPUTATION OF THE W-ARRAY AND $D_C(X,Y)$

To compute $D_c(X,Y)$, we make use of the fact that although this index does not itself seem to have any recursive properties, the index W(. , . , . , .), which is closely related to it, has the interesting properties proved in Theorem 3. The evaluation of the array W(. , . , . , .) has to be done in a systematic manner, so that any quantity W(i, e, s, t) must be evaluated before its value is required in any further evaluation. This is easily done by considering a four-dimensional coordinate system whose axes are i, e, s and t respectively. Initially, the value associated with the origin, W(0, 0, 0, 0) is assigned the value zero, and the contributions associated with the vertices on the axes are evaluated. Thus, W(i, 0, 0, 0), W(0, e, 0, 0), W(0, 0, s, 0) W(0, 0, 0, t) are evaluated for all permissible values of i, e, s and t. Subsequently, the i-e, i-s, i-t, e-s, e-t, s-t planes are traversed, and the contributions associated with the vertices on these planes are evaluated using the previously evaluated values. Then, the three-dimensional cubes i-e-s, i-e-t, i-s-t and e-s-t are traversed. Finally, the contributions corresponding to strictly positive values of the variables are evaluated. At each stage, the variables must be tested for permissibility. Finally, $D_c(X,Y)$ is evaluated by minimizing over the relevant contributions of W(.,.,.,.) associated with the points that lie on the four-dimensional line given by the parametric equation :

$$i = i; \quad e = N - M + i; \quad s = M - i - 2t; \quad t = t$$

Rather than use this traditional method for traversing the W-array we shall develop a compact version of it using a pair of three dimensional arrays instead of a four-dimensional array. To do this, we shall take advantage of the following fact. For a particular value of t, in order to compute W(i, e, s, t) for all permissible values of i, e and s, it is sufficient to store only the values of W (i, e, s, t-1) for all the corresponding permissible values of i, e and s. Consider the four-dimensional trellis described above. We shall successively evaluate the array Wc (for current W-array) in cubes *hyper-parallel* to the cube t = 0. Two arrays are maintained, namely,

(i) Wp: the cube *hyper-parallel* to t = 0, maintained for the previous value of t, and,

(ii) Wc: the cube *hyper-parallel* to t = 1 maintained for the current value of t.

The algorithm, given formally below, merely evaluates these two arrays in a systematic manner. Initially, the quantities associated with the individual axes are evaluated. The lines, planes and cubes of the Wc array are initialized and traversed as described above. Also, prior to updating Wp, its pertinent component required in the computation of $D_c(X,Y)$, is used to update the latter.

ALGORITHM GEN_CONSTRAINED_DISTANCE

Input: The strings $X = x_1x_2...x_N$, $Y = y_1y_2...y_M$, the set of elementary edit distances and the constraint sets T_t . Let R be the largest integer in the set Q_t.

Output: The constrained distance $D_c(X,Y)$.

Notation : Values at negative indices of Wc and Wp are set to infinity.

Method

```
For $t \leftarrow 0$ to $R$ Do
    For $i \leftarrow 0$ to $M-2t$ Do
        For $e \leftarrow 0$ to $N-2t$ Do
            For $s \leftarrow 0$ to Min$[M-i-2t, N-e-2t]$ Do
                $Wc(i, e, s) \leftarrow$ Min [ $Wc(i-1, e, s) + d_i(y_{i+s+2t})$,
                                  $Wc(i, e-1, s) + d_e(x_{e+s+2t})$,
                                  $Wc(i, e, s-1) + d_s(x_{e+s+2t}, y_{i+s+2t})$,
                                  $Wp(i,e,s)+d_t(x_{e+s+2t-1}x_{e+s+2t}, y_{i+s+2t-1}y_{i+s+2t})$ ]
                If $i,e,s,t$ are all equal to zero then $Wc(i,e,s) = 0$
            EndFor
        EndFor
        If $i \in T_t$, then $D_c(X,Y) \leftarrow$ Min$[D_c(X,Y), Wc(i, N-M+i, M-i-2t)]$
    EndFor
    For $i \leftarrow 0$ to $M-2t$ Do
        For $e \leftarrow 0$ to $N-2t$ Do
            For $s \leftarrow 0$ to Min$[M-i-2t, N-e-2t]$ Do
                $Wp(i, e, s) \leftarrow Wc(i, e, s)$
            EndFor
        EndFor
    EndFor
EndFor
END ALGORITHM GEN_CONSTRAINED_DISTANCE
```

Given the strings X and Y, it can be shown that $D_c(X,Y)$ is computed with $O(|X|\cdot|Y|\cdot\text{Min}(|X|,|Y|))$ space and in $O(R\cdot|X|\cdot|Y|\cdot\text{Min}(|X|,|Y|))$ time, where $0 \le R \le \text{Min}\left[\left\lfloor \frac{|X|}{2} \right\rfloor, \left\lfloor \frac{|Y|}{2} \right\rfloor\right]$.

V. NOISY SUBSEQUENCE RECOGNITION AND EXPERIMENTAL RESULTS

Let us assume the characteristic of the noisy channel are known, and further, let L_i be the expected number of insertions introduced in the process of transmitting U. This figure can be estimated although the actual number of symbols inserted in any particular transmission is unknown. Since U can be any arbitrary *subsequence* of X*, and since the words of the dictionary can be of completely different lengths, it is obviously meaningless to compare Y with every X ∊ **H** using the GLD. Thus, before we compare Y with the individual words of the dictionary, we have to use the additional information obtainable from the noisy channel.

Since the number of insertions introduced in any transmission is unknown, it is reasonable to compare X ∊ **H** and Y subject to the constraint that the number of insertions that actually took place is its *best estimate*. In the absence of any other information, this estimate is its expected value, L_i. However, if L_i is not a feasible value for the number of insertions, then the closest feasible value is used to compare X ∊ **H** and Y. An identical argument can be given for the reason why L_t, the expected number of GTs that take place per transmission, is used as the best estimate for the number of GTs that took place in yielding Y. This is the rationale for the algorithm presented below.

```
Algorithm Recognize_Noisy_Subsequences
Input: (i)  The dictionary of H and a noisy subsequence, Y. Also known is $L_t$, the expected number
            of transpositions that can take place per transmission and $L_i$, the expected number of
            insertions that can take place per transmission.
      (ii)  Y a noisy version of a subsequence of an unknown $X^* \in$ H.
Output :    The estimate $X^+$ of $X^*$.
Method:
            For every string $X \in$ H Do
                If $L_t$ and $L_i$ are feasible values Then
                    Set the constraint set $T_t$ with respect to $L_t$ and $L_i$
                Else
                    Set the constraint set $T_t$ with respect to the feasible integers closest to $L_t$ and $L_i$
                EndIf
                Invoke Gen_Constrained_Distance to compute $D_c(X,Y)$ Algorithm
            EndFor
            $X^+$ is the string minimizing $D_c(X,Y)$
END Algorithm Recognize_Noisy_Subsequences
```

To investigate the power of our new measure (and its computation) and to demonstrate the accuracy of our new scheme in the original PR problem various experiments were conducted. The results obtained were remarkable. The algorithm was compared with PR results obtained if

(i) only SID errors were assumed and corrected using Wagner & Fischer [WF74] algorithm,

(ii) SID and traditional transposition errors were assumed and corrected using Lowrance and Wagner [LW75] algorithm.

(iii) SID and generalized transposition errors were assumed and corrected using a recent unconstrained editing algorithm for all the four operations [OL94].

The dictionary used consisted of a hundred strings taken from the classical book on pattern recognition by Duda and Hart [DH73]. Each string was the first line of a section or sub-section of the book, starting from Section 1.1 and ending with Section 6.4.3. Further, to mimic a UNIX nroff (or troff) file, all the Greek symbols were typed in as English strings. Subsequently, to make the problem more difficult, the spaces betwees words were eliminated, thus discarding the contextual information obtainable by using the blanks as delimiters. Finally, these strings were randomly truncated so that the length of the words in the dictionary was uniformly distributed in the interval [40, 80]. Thus, the first line of [DH73, Sec. 3.4.1], which reads

"In this section we calculate the *a posteriori* density $p(\theta/X)$ and the desired probability"

yielded the following string in the dictionary H:

"inthissectionwecalculatetheaposterioridensitypthetaxandthedesiredpro".

A subset of the hundred words used in the dictionary is given in Table I.

* * * * *	**Insert Table I Here** **(A subset of words in H)**	* * * * *

With regard to error generation, the conditional probability of inserting any character a $\in$ A given that an insertion occurred was assigned the value 1/26; and the probability of deletion was set to be 1/20. The table of probabilities for substitution (typically called the confusion matrix) was based on the proximity of the character keys on a standard QWERTY keyboard and is given in Table II.

* * * * *	**Insert Table II Here** **(QWERTY Keyboard Confusion Matrix)**	* * * * *

Using the noise generation technique based on the principles described in [Oo87b] a set of 500 noisy subsequences, S, were generated. In this generation strategy a subsequence U of X^* was first obtained by randomly deleting the symbols from X^*. The resultant subsequence U had an average of 30.24 characters deleted from the original strings. This subsequence U was further subjected to insertion, deletion, substitution and transposition errors using a technique similar to the one described in [Oo93]. For the sake of completeness, a subset of the subsequences U, their noisy versions Y and the number of errors associated with them is presented in Table III. The error statistics associated with the noisy subsequences used is given in Table IV. Notice that the percentage error was intentionally made to be large relative to the average length of the words so as to test the various algorithms for such error conditions. Observe also that even though the number of errors associated with each of the noisy subsequences is large, it would be even larger if we view it from the perspective of the set of *standard* edit operations where a GT is a combination of two substitution errors.

* * * * *	**Insert Table III Here** **(A subset of the noisy subsequences)**	* * * * *

	Avg. Errors
Number of insertions	2.142
Number of deletions	30.442
Number of substitutions	3.220
Number of transpositions	5.410
Total number of errors	41.214
Percentage error	68.69%

Table IV: Table showing the average error statistics in the individual noisy subsequences used in the experiments.

As is typical in PR problems using edit distances [SK83, Oo87b], the individual edit distances were calculated for all a,b,c,d $\in$ A as follows[2] :

$$d_S(a,b) = -\ln [\Pr(a \rightarrow b) / \Pr(a \rightarrow a)]$$

[2]The substitution, insertion and deletion distance functions were computed once and subsequently maintained as arrays. But since the maintenance of the generalized transposition distances would require a 4-dimensional array, they were computed on invocation.

$$d_e(a,\theta) = -\ln [\Pr(a\rightarrow\theta) / \Pr(a\rightarrow a)]$$

$$d_i(\theta,a) = -K_i \cdot \ln [\Pr(a \text{ is inserted} \mid \text{insertion occurred}) / \Pr(a\rightarrow a)]$$

$$d_t(ab,cd) = 1 + K_t\cdot(d_s(a,d) + d_s(b,c)).$$

In the above, the value of K_i was determined as the most conservative value (among all substitutions) which satisfied the triangular inequality $d_s(a,b) < d_e(a) + d_i(b)$. In our experiments K_i had the value 1.3 because, for the confusion matrix used, that was the most conservative distance for insertion which satisfied this inequality law. As opposed to this, since K_t was not required to satisfy any rigid inequality constraints, it was varied as a parameter in the PR system and the recognition accuracy was studied as a function of K_t. The variation of the recognition accuracy as a function of K_t is tabulated in Table V and drawn graphically in Figure I. Notice that the accuracy has a minimum value of 85.4 % when K_t is unity and increases to a maximum value of 94 %. It thereafter dips marginally. Thus we can reasonably diminish the "importance" of GT operations and increase the recognition accuracies.

Value of K_t	**Accuracy**
1.0	85.40
1.3	88.40
1.5	90.00
1.7	91.60
2.0	92.40
2.3	92.80
2.5	93.00
2.7	93.20
3.0	93.40
3.3	93.40
3.6	93.80
4.0	94.00
4.3	93.60
4.5	93.40
5.0	93.40

Table V: The variation of the recognition accuracy as a function of K_t

The four algorithms, Wagner & Fischer (WF), Lowrance & Wagner (LW), the unconstrained algorithm for generalized transpositions (SID_GT) [OL94] and our constrained algorithm (Const_SID_GT), were tested with the 500 noisy subsequences. In the case of our algorithm, rather than have the constraint set use only a single feasible integer for the number of insertions and transpositions, the algorithm was marginally modified to include a small range of integers. Also, Note that the weights of insertion, deletion and substitution for Lowrance and Wagner's algorithm [LW75] are the respective average distances obtained from the above distances. To ensure the correctness of the latter algorithm, the weight of transposition was obtained by selecting the minimum weight that could satisfy $2W_t \geq W_i + W_e$.The results obtained in terms of accuracy are tabulated in Table VI. Note that our scheme far outperforms the traditional string correction algorithm (94.0 % instead of 64.2 %). It also outperforms the Lowrance and Wagner algorithm (which had an accuracy of 75.6 %). Our recent unconstrained distance criterion for all the four errors [OL94] yielded an accuracy of 74.6 %. The power of the strategy in PR is obvious !!

Algorithm	Recognition Accuracy
Wagner & Fischer (WF)	64.20%
Lowrance & Wagner (LW)	75.60%
Unconstrained Editing (SID_GT)	74.60%
Constrained Editing (Const_SID_GT) with K_t=4	94.00%

Table VI: The recognition accuracy of the various algorithms implemented.

VI. CONCLUSIONS

In this paper we have considered the problem of recognizing strings by processing their noisy subsequences. The solution which we propose is the only known solution in the literature when the noisy subsequences contain substitution, insertion, deletion and generalized transposition errors. Given a noisy subsequence Y of an unknown string $X^* \in$ **H**, the technique we propose estimates X^* by computing the constrained edit distance between every $X \in$ **H** and Y. The procedure to compute the latter edit distance has been described in detail and the time and space required for the procedure have been given. Experimental results using strings of length between 40 and 80 and with a high percentage of noise, demonstrate the power of the strategy in pattern recognition.

The disadvantage of the scheme, in that it individually compares Y with every element of **H** is currently being studied by using tries to represent the dictionary. We are also currently investigating the use of the results obtained in this paper in the recognition of chain-coded boundaries [MV93].

REFERENCES

[AS82] K. Abe and N. Sugita, Distances between strings of symbols -- review and remarks, *Proceedings of the Sixth Int. Conf. on Pat. Recog*, ICPR-6 : 172-174 (1982).

[GM90] J. Golic and M. Mihaljevic, A noisy clock-controlled shift register cryptanalysis concept based on sequence comparison approach, *Proceedings of EUROCRYPT 90,* Aarhus, Denmark, 487-491 (1990).

[HD80] P. A. V. Hall and G. R. Dowling, Approximate string matching, *Comput. Surveys*, 12:381-402 (1980).

[Hi77] D. S. Hirschberg, Algorithms for longest common subsequence problem, *J. Assoc. Comput. Mach.*, 24:664-675 (1977).

[HS77] J. W. Hunt and T. G. Szymanski, A fast algorithm for computing longest common subsequences, *Comm. Assoc. Comput. Mach.*, 20:350-353 (1977).

[KO81] R. L. Kashyap and B. J. Oommen, An effective algorithm for string correction using generalized edit distances -I. Description of the algorithm and its optimality, *Inform. Sci.*, 23(2):123-142 (1981).

[KO83a] R. L. Kashyap and B. J. Oommen, A common basis for similarity and dissimilarity measures involving two strings, *Internat. J. Comput. Math.*, 13:17-40 (1983).

[KO83b] R. L. Kashyap and B. J. Oommen, Similarity measures for sets of strings, *Internat. J. Comput. Math.*, 13:95-104 (1983).

[KO83c] R. L. Kashyap and B. J. Oommen, The noisy substring matching problem, *IEEE Trans. Software Engg.*, SE-9:365-370 (1983).

[KO84] R. L. Kashyap, and B. J. Oommen, String Correction Using Probabilistic Methods, *Pattern Recognition Letters*, 147-154 (1984).

[Le66] A. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals, *Soviet Phys. Dokl.*, 10:707-710 (1966).

[LW75] R. Lowrance and R. A. Wagner, An extension of the string to string correction problem, *J. Assoc. Comput. Mach.*, 22:177-183 (1975).

[Ma78] D. Maier, The complexity of some problems on subsequences and supersequences, *J. Assoc. Comput. Mach.*, 25:322-336 (1978).

[MV93] A. Marzal and E. Vidal, Computation of normalized edit distance and applications, *IEEE Trans. on Pat. Anal. and Mach. Intel.*, PAMI-15:926-932 (1993).

[MP80] W. J. Masek and M. S. Paterson, A faster algorithm computing string edit distances, *J. Comput. System Sci.*, 20:18-31 (1980).

[NW70] S. B. Needleman and C. D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two protiens, *J. Mol. Biol.*, 443-453 (1970).

[OF91] B. J. Oommen and E. T. Floyd, An Improved Algorithm for the recognition of noisy subsequences, *Proceedings of the 1991 IASTED International Symposium on Artificial Intelligence Applications and Neural Networks*, Zurich, 145-147 (1991).

[OL94] B. J. Oommen and R. K. S. Loke, Pattern recognition of strings with substitutions, insertions, deletions and generalized transpositions. Submitted for Publication. Available as a Carleton University Technical Report.

[Oo87a] B. J. Oommen, Constrained string editing, *Information Sciences*, 40: 267-284 (1987).

[Oo87b] B. J. Oommen, Recognition of Noisy Subsequences Using Constrained Edit Distances, *IEEE Trans. on Pattern Anal. and Mach. Intel.*, PAMI-9:676-685 (1987).

[Oo93] Oommen, B.J. and Kashyap, R.L., "Symbolic Channel Modelling for Noisy Channels which Permit Arbitrary Noise Distributions", *Proceedings of the 1993 International Symposium on Computer and Information Sciences*, Turkey, pp. 492-499 (1993).

[OTK76] T. Okuda, E. Tanaka, and T. Kasai, A method of correction of garbled words based on the Levenshtein metric, *IEEE Trans. Comput.*, C-25:172-177 (1976).

[Pe80] J. L. Peterson, Computer programs for detecting and correcting spelling errors, *Comm. Assoc. Comput. Mach.*, 23:676-687 91980).

[Sa72] D. Sankoff, Matching sequences under deletion/insertion constraints, *Proc. Nat. Acad. Sci. U.S.A.*, 69:4-6 (1972).

[SK83] D. Sankoff and J. B. Kruskal, *Time Warps,String Edits and Macromolecules: The Theory and practice of Sequence Comparison*, Addison-Wesley (1983).

[Uk85] E. Ukkonen, Algorithms for approximate string matching, Information and Control, 64: 100-118 (1985).

[WF74] R. A. Wagner and M. J. Fisher, The string to string correction problem, *J. Assoc. Comput. Mach.*, 21:168-173 (1974).

[WC76] C. K. Wong and A. K. Chandra, Bounds for the string editing problem, *J. Assoc. Comput. Mach.*, 23:13-16 (1976).

Index of X^*	X^*
1	sincetheadventofthedigitalcomputertherehasbeenaconstante
2	toillustratesomeofthethetypesofproblemsweshalladd
3	theprecedingexamplecontainsmanyoftheelementsofthemostco
4	therearemanyproblemsinpatternrecognitionandmachineperceptionforwhichtheclassif
5	theoriginationofpartiofthisbookisprimarilystatisticalchaptertwostatesthecla
6	thepublishedliteratureonpatternclassificationandsceneanalysis
7	bayesdecisiontheoryisafundamentalstatisticalspproachtotheproblemofpatte
8	weshallnowformalizetheideasjustconsideredandshallgeneralizetheminfourwaysonewes
9	letusspecializetheseresultsbyconsideringthetwocategoryclassificatio
10	inclassificationproblemseachstateofnatureisusu
11	therearemanydifferentwaystorepresentpatternclassifierso
12	whilethetwocatagorycaseishustaspecialinstan
13	bythinkingofaclassifierasadeviceforpartitionin
14	thestructureofabayesclassifierisdeterminedprim
15	webeginwiththeunivariatenormaldensitypxequaloneoverroes
16	thegeneralmultivariatenormaldensityiswrittenaspxequaloneovertwop
17	insectiontwofivewesawtheminimumerrorrateclassificationcanbeachievedbyuseofthe
18	thesimplestcaseoccureswhenthefeaturesarestatisticallyindependentandwheneac
18	anothersimplecasearisesw henthecovariancematricesforalloftheclassesareidenticalg
20	inthegeneralmultivariatenormalcasethecovarianc
21	untilnowwehaveassumedthatthefeaturevectorxcouldbeanypointxineuclideandspacehowe
22	asanexampleofaspecificcalssificationproblemin
23	letusreconsiderourintroductoryexampleofdesigningaclassifiertosorttwoty
24	wehavenowcompletedanexpositionofbayesdecisiontheorywithspeciale
25	decisiontheoryisassociatedwiththenamesofmanywellhnownstatisti
26	inchaptertwowesawhowwecoulddesignanoptimalcla
27	supposethatweseparateasetofsamplesaccordingtocalsssothatwehavecsetsof
28	toseehowtheseresultsapplytoaspecificcasesupposethatthesam
29	inthegeneralandmoretypicalmultivariatenormalcaseneithe
30	readersfamiliarwithstaticticsknowthatthemaximumlikelihoodes
31	thecomputationoftheaposterioriprobabilitiespwixliesatthehea
32	althoughthedesiredprobabilitydensitypxisunknownweass
33	inthissectionwecalculatetheaposterioridenditypthetaxandthedesiredpro
34	havingobtainedtheaposterioridensitypmuexalltha
35	thetreatmentofthemultivariatecaseisadirectgeneralizationoftheuniv
36	wehavejustseenhowthebayesianapproachcanbeusedtoobtainthedesiredde
37	fromapracticalviewpointtheformalsolutionprovidedbyeqsfourteenthirtyfourandth
38	toseehowthefactorizationtheoremcanbeusedtoobtai
39	inpractivalmulticategoryapplicationsitisnotatallunusualtoenco
40	webeginbyconsideringtheproblemofestimatingacovariancematrixthisr

Table I : A subset of 40 out of the 100 words used in the dictionary, **H**.

Index of string	Strings subsequent to deletions (subsequences of the original strings)	Subsequences with substitution, insertion, deletion, and generalized transposition errors	Number of errors
1	sincetdventofthtalcompuherehaconte	sincwtdvetnohfhatzlcopmuheerhaocnte	31
1	dventofttalcomrehenaco	dvntenhyofttwwylacxkorfekenaoc	49
1	sincentoftalcompuherehasbnstant	sinwcertofstlocmyipherfeasbsntatn	37
1	adventhedigitmputerthebeenacons	advwenyhedriimtpyhuertehbeeancosn	36
1	stheadventofhedigicoehasbnstant	sthweaxvenotfehdyfiicofeasbsntatn	36
2	toillomeetypesowesha	oillomeeytpseoewshz	34
2	toillesomeoftheofproblalladd	toihwllezomprfhtewfkprolballdad	33
2	tratesethetypemswed	trghactesetyteyqepmdwsend	40
2	tesoftheetyoblems	tepsoftheeytolbeynsm	38
2	ustrateoftheetypesofswalladd	usthwratwofyteteyweosofwcalldad	33
3	ecedingexecoftheelemeemos	ecedingexceotfyeelemeemos	35
3	ecedintainanelemenfthemo	ecntedhinyaiznnwleeweknftefmo	44
3	thepreceampleinsmalementsstco	theprecezpmliennsaleemntstdco	34
3	thcedngexamplesmanyoeelemenemostco	thcexngexmapelsamznyoeelemneemwiyqo	33
3	edingelecomanyoftnttho	edntinhyelecwwymoaxinoffknttoh	48
4	thereaanyprobpattrecogndmachitionfthec	yhereaantoyrpopbattrecgojnxmaahitongfgec	56
4	earmanyproatternrecoandmaercepochtheclas	ezremanyporattenrrecaonenearcpacthhecksa	51
4	earemoblemsiternrmachiptionrwhichth	eareroblesmietrenkmacihptinorwodxnht	57
4	theresnrecnitndmachinepeforwhichteclas	tyeresnreocgintsbmacihnepfeorawctheclas	50
5	nationofpfthissprimartatistiaptertwhec	rationoftoptfhwisrpimatrjayoasitperetgec	55
5	theorigiofpartisbookistachartwost	theoriyiopratribdooksitacahrtowst	51
5	rigifpartbookisprstatiaptertwosta	rigifpzrtpnoiksrostaifaptretwsota	53
5	theorionfpaisbookaristicaertwosla	theorirnfzpibsokkaritcicaretwsola	52
5	theorigiiofthimaristicalchatate	theworugiirotihmyfaistcialcahtaet	55
6	thepublierapatternsificationaeneanaly	therubliearpfatrensiifcatqoinzneraany	37
6	thublishetureoernclasationandsclysis	thubrisheutrkeebrzclaastioannevapyiss	40
6	isureonnclassificonannalysi	isthureorncapsoswfcionahnalyis	46
6	thepedliterureonpclassificatdscenaly	thepwdlitreuerolnzclassifiactevawnlay	39
6	thepudliteratterassifiandscenis	thewpuxlitreatteyarssiifandcsensi	39
7	bayeiontryismentalstcaroachtotheoblem	baywiontriysemnatlstacroaqtctphtrolbm	47
7	bayeyisafundaisticahtoblemofpa	bayyeyusafnudiasxifcahotbleomfpa	50
7	bayeeorndamenticalspproaroblemof	bayyeerrndmaethixaclsprpoarboleonf	50
7	bayesdeciheoryiundamensticalsppotheproble	bzyesdeciehoyrinudambestuaclspatehpfobek	45
7	bayesdecryisandamentasticpproaceprobpatt	bzyesdecriysnadnaentsatixpproaaporboaft	45
8	owethsjustedandseneralizeinfourwnewes	oweyhsjusetdbadesnerlaizeqnifpruwnwes	54
8	nowfzethonsiandalizethewaysonewe	nowyfzwthosninadxlaizehtewasyonwse	57
8	weshalizetasjustcondshaeraliminrwaysowes	wwshalizeatsujsvfondhsaetlaomaiwyaskwes	54
8	weshallnowformheidestceredandsalizetheone	wwshallnofwomrhisdescterwadndaaieztgeiec	54
9	sszethesesultsbsideriategorycfic	sszyetyeseuslshbxicderaitegroycirc	47
9	letusspelizethesultserincategorycficat	retusspetolzietfeusltsrejinqaaeroycifcaf	46
9	letulizethseresultyconsidengthetwocoryclica	letulizetsheersautyxxnosicegteyhwcoirycailca	41
9	peciaesersbyconsiingtheegoryific	pecyiawserbsyocnxicinghteegroyiirc	46
9	sspecialeseressideritwocategasstio	sspexialeesrdesdizeriwtocaetgawayoo	46
10	claicationmseachstatesusu	claicatiomnsaexshtatseusu	26
10	ficatlemstateoeisusu	jcatlemsattoeesiusy	32
10	ficatseachstatture	giycatseacshttatrunce	35
10	lassiftionproblchstatnature	lathssifyiopnrnowlhcstahtatuer	30
10	inclassnproblehstatetureisu	inthclasznpotbrlwhtsateitreihs	32

Table III: A subset of the subsequences, their respective noisy versions and the number of errors associated with each noisy subsequence.

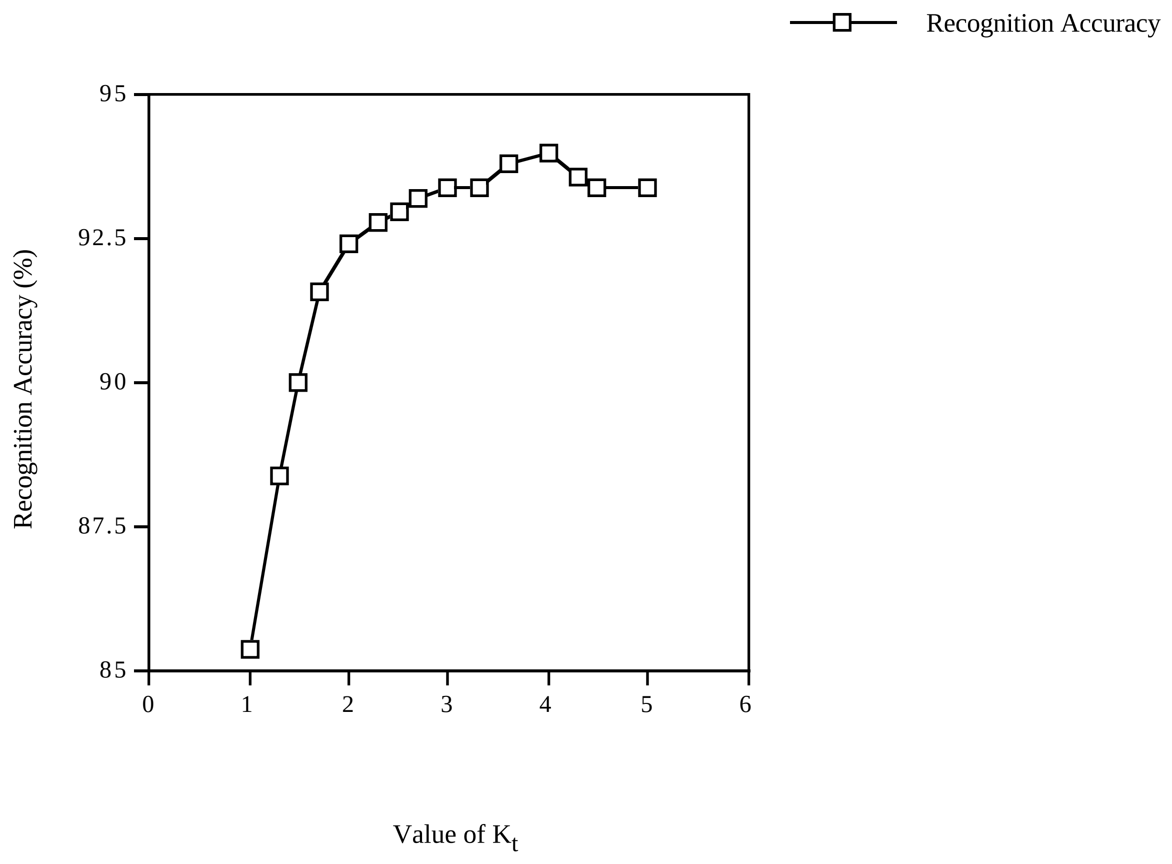

Figure I : The variation of the recognition accuracy plotted as a function of K_t.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
a	867	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	14	1	20	1
b	1	861	1	1	1	1	14	14	1	1	1	1	1	20	1	1	1	1	1	1
c	1	1	861	14	1	14	1	1	1	1	1	1	1	1	1	1	1	1	1	1
d	1	1	14	835	14	20	1	1	1	1	1	1	1	1	1	1	1	10	20	1
e	1	1	1	14	857	5	1	1	1	1	1	1	1	1	1	1	1	20	14	1
f	1	1	14	20	10	824	20	1	1	1	1	1	1	1	1	1	1	20	1	10
g	1	14	1	1	1	20	835	20	1	1	1	1	1	1	1	1	1	5	1	14
h	1	14	1	1	1	1	20	835	1	20	1	1	1	14	1	1	1	1	1	5
i	1	1	1	1	1	1	1	1	861	10	14	5	1	1	20	1	1	1	1	1
j	1	1	1	1	1	1	1	20	10	835	20	1	14	14	1	1	1	1	1	1
k	1	1	1	1	1	1	1	1	14	20	848	20	14	1	10	1	1	1	1	1
l	1	1	1	1	1	1	1	1	5	1	20	876	1	1	14	14	1	1	1	1

m	1	1	1	1	1	1	1	5	1	14	14	1	876	20	1	1	1	1	1	1
n	1	20	1	1	1	1	5	14	1	14	1	1	20	857	1	1	1	1	1	1
o	1	1	1	1	1	1	1	1	20	1	14	14	1	1	861	20	1	1	1	1
p	1	1	1	1	1	1	1	1	1	1	1	14	1	1	20	893	1	1	1	1
q	14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	892	1	5	1
r	1	1	1	14	17	14	5	1	1	1	1	1	1	1	1	1	1	863	1	17
s	17	1	1	17	10	1	1	1	1	1	1	1	1	1	1	1	5	1	841	1
t	1	1	1	1	1	14	14	10	1	1	1	1	1	1	1	1	1	17	1	858
u	1	1	1	1	1	1	1	14	17	14	10	1	1	1	1	1	1	1	1	1
v	1	17	17	5	1	14	14	1	1	1	1	1	1	1	1	1	1	1	1	1
w	10	1	1	5	17	1	1	1	1	1	1	1	1	1	1	1	17	1	14	1
x	5	1	17	14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	14	1
y	1	1	1	1	1	1	14	14	1	5	1	1	1	1	1	1	1	1	1	17
z	14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	14	1

Table I: The "confusion matrix" with the probabilities of substituting a character with another character. The figures in the table are to be multiplied by a factor of 10^{-3}.