Noisy Subsequence Recognition Using Constrained String Editing Involving Substitutions, Insertions, Deletions and Generalized Transpositions[1]

B. J. Oommen and R. K. S. Loke
School of Computer Science
Carleton University
Ottawa ; CANADA : K1S 5B6

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. We solve the *noisy subsequence recognition problem* by defining and using the constrained edit distance between X **H** and Y subject to any arbitrary edit constraint involving the number and type of edit operations to be performed. 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.

1 Introduction

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 [3]) after the individual symbols of the "alphabet" have been hypothesized using statistical methods. 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 pose it in its generality, let us assume that a sender intends to transmit a string X^* 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 substitution, deletion, insertion 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

[1]Partially supported by the Natural Sciences and Engineering Research Council of Canada.

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.

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.

The relationship between our present work and the work that has been done in noisy string processing is found in [8]. The paper [8] also gives references to the longest common subsequence problem and sequence correction using the GLD as a criterion for strings, substrings, dictionaries treated as generalized tries and for grammars. Also, although some work has been done to extend the traditional set of SID operations to include adjacent transpositions [5, 9, 10] our work tackles the unsolved problem for constrained editing for "Generalized" Transposition (GT) errors.

1.1 Notation

A is a finite alphabet, and A^* is the set of strings over **A**. θ is the null symbol, where θ **A**, and is distinct from μ the empty string. Let $\tilde{\mathbf{A}} = \mathbf{A} \approx \{\theta\}$. A string X A^* of the form $X = x_1...x_N$, where each x_i **A**, and is said to be of length |X| = N. Its prefix of length i will be written as X_i, for 1 i 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* C is a mapping from $\tilde{\mathbf{A}}^*$ to $\mathbf{A}^*$: $C(Z')$ is Z' with all occurrences of the symbol θ removed from Z'. Note that C preserves the order of the non-θ symbols in Z'. For example, if $Z' = f\theta o\theta r$, $C(Z') = for$.

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 follows : (i) $d_s(.,.)$ is a map from **A** X **A** ∅ $\mathbf{R}^+$ and is the Substitution Map. $d_s(a,b)$ is the distance associated with substituting b for a, a,b **A** ; (ii) $d_i(.)$ is a map from **A** ∅ $\mathbf{R}^+$ and is called the Insertion Map. The quantity $d_i(a)$ is the distance associated with inserting the symbol a **A**; (iii) $d_e(.)$ is a map from **A** ∅ $\mathbf{R}^+$ and is called the Deletion or Erasure Map. The quantity $d_e(a)$ is the distance associated with deleting (erasing) the symbol a **A**, and (iv) $d_t(.,.)$ is a map from $\mathbf{A}^2$ X $\mathbf{A}^2$ ∅ $\mathbf{R}^+$ called the

Transposition Map. The quantity $d_t(ab,cd)$ is the distance associated with transposing the string "ab" into "cd". A formal expression for D(X,Y) in terms of these elementary edit distances and the set of ways by which X can be edited to Y, $\Gamma_{X,Y}$ is given in [8].

2 Permissible and Feasible Edit Constraints

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:

$0 \quad t \quad \text{Min} \left[\in \frac{N}{2} \vee, \in \frac{M}{2} \vee \right]$; $\quad \text{Max}[0, M-N] \quad i \quad q \quad M$;

$0 \quad e \quad r \quad N$; $\quad 0 \quad s \quad \text{Min}[N, M]$.

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

$H_t = \{ j \mid 0 \quad j \quad \text{Min} \left[\in \frac{N}{2} \vee, \in \frac{M}{2} \vee \right] \}$; $\quad H_i = \{ j \mid \text{Max}[0, M-N] \quad j \quad M \}$;

$H_e = \{ j \mid 0 \quad j \quad N \}$; $\quad H_s = \{ j \mid 0 \quad j \quad \text{Min}[N, M] \}$ (1)

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 $t \quad H_t$, $i \quad H_i$, $e \quad H_e$ and $s \quad H_s$, the inequalities $\{ i + s + 2t \quad M$; $\quad e + s + 2t \quad N \}$ are also satisfied.

Theorem 1 specifies the permitted forms of the feasible quadruples 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

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

Proof. The proof is included in the unabridged paper [8].

$\rightarrow\rightarrow\rightarrow$

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 (1). 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 :

$Q_i^t = \{ i \mid i \quad M-2 \quad t \} \leftrightarrow Q_i$; $\quad Q_e^t = \{ e \mid e \quad N-2 \quad t \} \leftrightarrow Q_e$;

and $Q_s^t = \{ s \mid s \quad \text{Min}[N-2t, M-2t]\} \leftrightarrow Q_s$.
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. The proof is given in the unabridged paper [8] and involves computing subsets of H_i for the various subsets Q_i^t, Q_e^t and Q_s^t.

$\rightarrow\rightarrow\rightarrow$

The set (the subset of H_i) referred to above, which describes the constraint given a value of t will be written as T_t. A detailed example of how these sets are created is found in the main paper. Also, we shall refer to the edit distance subject to the constraint T_t as D_t (X,Y). By definition, if $T_t =$, then D_t (X,Y) = . The distance for the optimal edit transformations is denoted by D_c(X,Y) which is the minimum of all D_t (X,Y).

3 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 $q = i+s+2t$. Using the notation in [8], 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 $\Gamma_{i,e,s,t}$. Assuming (i,e,s,t) is feasible for the problem, W(i,e,s,t) has the expression

$$W(i, e, s, t) = \min_{(X_r', Y_q') \in \Gamma_{i,e,s,t}} \sum_{j=1}^{|X_r'|} d(X_{rj}', Y_{qj}') \qquad (3)$$

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

Theorem 3 Let W(i, e, s, t) be as defined in (3) for strings X and Y. Then,

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

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

Proof. The proof is involved and can be found in the unabridged paper [8]. $\rightarrow\rightarrow\rightarrow$

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 theorems whence we derive a computational scheme for D_c(X,Y).

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

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

Proof. The theorem follows from Theorem 1 by setting r = N and q = M. $\rightarrow\rightarrow\rightarrow$

Theorem 5 The distance D_c(X,Y), is obtained as follows:

$$D_c(X,Y) = \underset{k}{\text{Min}}\ Q_t\ D_k(X,Y)$$

Sketch of 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.
$\rightarrow\rightarrow\rightarrow$

4 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 with the vertices, axes, planes and cubes are evaluated sequentially in an intelligent manner. 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; e = N - M + i; s = M - i - 2t; 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, 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 as *Algorithm Gen_Constrained_Distance* below, 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. It is clear that given X and Y, $D_c(X,Y)$ 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 $0 \quad R \quad \text{Min}\left[\in\frac{|X|}{2}\vee,\in\frac{|Y|}{2}\vee\right]$.

ALGORITHM GEN_CONSTRAINED_DISTANCE

Input: The strings $X = x_1x_2...x_N$, $Y = y_1y_2...y_M$, the 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* ♦ 0 to *R* **Do**
For *i* ♦0 to M-2*t* **Do**
For *e* ♦0 to N-2*t* **Do**
For *s* ♦0 to **Min** [M-*i*-2*t*,N-*e*-2*t*] **Do**

$$Wc(i,e,s) \blacklozenge \textbf{Min}\Big[\ Wc(i\text{-}1,e,s) + d_i(y_{i+s+2t}),\ Wc(i,e\text{-}1,s) + d_e(x_{e+s+2t}),\ Wc(i,e,s\text{-}1) + d_s(x_{e+s+2t}, y_{i+s+2t}),\ Wp(i,e,s) + \text{cost}\Big],$$

$$\text{where cost} = d_t(x_{e+s+2t-1}x_{e+s+2t}, y_{i+s+2t-1}y_{i+s+2t})$$

If *i*,*e*,*s*,*t* are all equal to zero then Wc(i,e,s) = 0
EndFor
EndFor
If $i \quad T_t$, then $D_c(X,Y) \blacklozenge \textbf{Min}[D_c(X,Y), Wc(i, N\text{-}M+i, M\text{-}i\text{-}2t)]$
EndFor
For *i* ♦ 0 to M-2*t* **Do**
For *e* ♦ 0 to N-2*t* **Do**
For *s* ♦ 0 to **Min**[M-*i*-2*t*,N-*e*-2*t*] **Do**
Wp(*i*, *e*, *s*) ♦ Wc(*i*, *e*, *s*)
EndFor
EndFor

EndFor
EndFor
END ALGORITHM GEN_CONSTRAINED_DISTANCE

5 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*. Of course, in the absence of any other information, the best estimate of the number of insertions that could have taken place is indeed its expected value, which we have referred to as 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. Thus, the procedure to estimate X^* is as follows : If L_t and L_i are feasible values, the constraint set T_t is set with respect to L_t and L_i. Otherwise, the constraint set T_t is set with as the feasible integers closest to L_t and L_i. The distance $D_c(X,Y)$ is computed using Algorithm Gen_Constrained_Distance, for every X **H**. X^+, the estimate of X^*, is obtained as the string which minimizes $D_c(X,Y)$. The formal algorithm is in [8].

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 [11] algorithm, (ii) SID and traditional transposition errors were assumed and corrected using Lowrance and Wagner [5,10] algorithm and (iii) SID and generalized transposition errors were assumed and corrected using a recent unconstrained editing algorithm for all the four operations [7].

The dictionary, **H**, used consisted of a hundred strings taken from the classical book on pattern recognition by Duda and Hart and were randomly truncated so that the length of the words in **H** was uniformly distributed in the interval [40, 80]. Using random deletions as in [6] a set of 500 subsequences were generated. The resultant subsequence U had an average of 30.24 characters deleted from the original strings. Each subsequence U was further subjected to insertion,

deletion, substitution and transposition errors using a technique similar to the one described in [8]. A typical example of a noisy subsequence corrected is given below :

X^*: theoriginationofpartiofthisbookisprimarilystatisticalchaptertwostatesthecla

U : theorigiofpartisbookistachartwost

Y : theoriyiopratribdooksitacahrtowst

In this case, the number of errors is 51. The error statistics associated with the set of noisy subsequences used is given in Table 1. Notice that even though the number of errors associated with each Y is large, it would be even larger from the perspective of the set of *standard* operations where a GT is a combination of two substitution errors.

The four algorithms 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. The details of the individual inter-symbol edit distance and the distance assignments used are given in [8]. The results obtained in terms of accuracy are tabulated in Table 2. 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 [7] yielded an accuracy of 74.6 %. The power of the strategy in PR is obvious !!

6 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^* **H**, the technique we propose estimates X^* by computing the constrained edit distance between every X **H** and Y. 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.

References

1. P. A. V. Hall and G. R. Dowling, Approximate string matching, Comput. Surveys, 12:381-402 (1980).
2. 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).
3. 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).
4. A. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals, Soviet Phys. Dokl., 10:707-710 (1966).

5. R. Lowrance and R. A. Wagner, An extension of the string to string correction problem, J. Assoc. Comput. Mach., 22:177-183 (1975).
6. B. J. Oommen, Recognition of noisy subsequences using constrained edit distances, IEEE Trans. on Pat. Anal. and Mach. Intel., PAMI-9:676-685 (1987).
7. B. J. Oommen and R. K. S. Loke, Pattern recognition of strings with substitutions, insertions, deletions and generalized transpositions. Unabridged Paper. Available as a Carleton University technical report (1994).
8. B. J. Oommen and R. K. S. Loke, Noisy subsequence recognition using constrained string editing involving substitutions, insertions, deletions and generalized transpositions. Unabridged Paper. Available as a Carleton University technical report (1994).
9. J. L. Peterson, Computer programs for detecting and correcting spelling errors, Comm. Assoc. Comput. Mach., 23:676-687 (1980).
10. D. Sankoff and J. B. Kruskal, Time Warps,String Edits and Macromolecules: The Theory and practice of Sequence Comparison, Addison-Wesley (1983).
11. R. A. Wagner and M. J. Fischer, The string to string correction problem, J. Assoc. Comput. Mach., 21:168-173 (1974).

	Average Errors in Y
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 1. A table showing the average error statistics in the noisy strings.

Algorithm	**Accuracy**
WF	64.20%
LW	75.60%
SID_GT	74.60%
Const_SID_GT	94.00%

Table 2. The recognition accuracies of the various algorithms tested. The details of the algorithms used are found in the text.