# Optimal and Information Theoretic Syntactic Pattern Recognition for Traditional Errors[+]

**B. J. Oommen[1] and R. L. Kashyap[2]**

## Abstract

In this paper we present a foundational basis for optimal and information theoretic syntactic pattern recognition. We do this by developing a rigorous model, $\mathfrak{M}^*$, for channels which permit arbitrarily distributed substitution, deletion and insertion syntactic errors. More explicitly, if A is any finite alphabet and $A^*$ the set of words over A, we specify a stochastically consistent scheme by which a string $U \in A^*$ can be transformed into any $Y \in A^*$ by means of arbitrarily distributed substitution, deletion and insertion operations. The scheme is shown to be *Functionally Complete* and *stochastically consistent*. Apart from the synthesis aspects, we also deal with the analysis of such a model and derive a technique by which $\Pr[Y|U]$, the probability of receiving Y given that U was transmitted, can be computed in cubic time using dynamic programming. Experimental results which involve dictionaries with strings of lengths between 7 and 14 with an overall average noise of 39.75 % demonstrate the superiority of our system over existing methods. The model also has applications in speech and uni-dimensional signal processing.

## 1. Introduction

In the field of statistical Pattern Recognition (PR), the patterns are represented using numerical features. As opposed to this, in syntactic and structural PR the classifiers are designed to be trained and tested by representing the patterns symbolically using primitive (elementary) symbols. Essentially, the system models the noisy variations of typical samples of the patterns symbolically, and these models are utilized in the training and testing phases. In statistical PR, the noisy samples from a class are modeled (either parametrically or non-parametrically) using the class conditional probability distributions. If these distributions are known, *information theoretic, minimum probability of error* classification is possible [5,7].

In this paper we shall attempt to lay the foundation for information theoretic, minimum probability of error *syntactic* PR systems which permit arbitrarily distributed noise. In this paper we shall only deal with syntactic PR of patterns which are represented "linearly" as strings. The problem of developing similar classifiers for PR systems using two-dimensional structures such as trees and webs remains open.

[+] We would like to dedicate this paper to the memory of the late Prof. K. S. Fu who pioneered the field of *Syntactic Pattern Recognition*. Both authors remember their friend and colleague with respect. We are also very indebted to Richard Loke for helping us prepare the final manuscript and for assisting us obtain the experimental results.

[1] Senior Member IEEE. Can be contacted at : School of Computer Science, Carleton University, Ottawa ; Canada : K1S 5B6. The work of this author was supported in part by the Natural Sciences and Engineering Research Council of Canada.

[2] Fellow IEEE. Can be contacted at : School of Electrical Engineering, Purdue University, W. Lafayette ; Indiana : 47907. The work of this author was supported in part by the Office of Naval Research and BMD under Contract ONR N00014-91-J-4126.

Typically, syntactic PR systems work as follows. The system has a dictionary which is a collection of all the ideal representations of the objects in question. When a noisy sample has to be processed, the system compares it with every element in the dictionary. This comparison is done sequentially or using a grammatical parsing mechanism. The question of comparing patterns reduces to one of comparing their string representations, and this is typically achieved using three standard edit operations - substitution, insertion and deletion. To achieve this, one usually assigns a distance for the elementary *symbol* operations, and the inter-pattern distance is computed as a function of these distances.

The elementary distances can be assigned weights in a variety of ways. If $\mathbf{R}^+$ is the set of non-negative real numbers, the elementary distances are defined using three elementary functions $d_s(.,.)$, $d_i(.)$ and $d_e(.)$ :

(i) $d_s(.,.)$ is a map from $\mathbf{A} \text{ X } \mathbf{A} \rightarrow \mathbf{R}^+$ and is the Substitution Map.

(ii) $d_i(.)$ is a map from $\mathbf{A} \rightarrow \mathbf{R}^+$ and is called the Insertion Map.

(iii) $d_e(.)$ is a map from $\mathbf{A} \rightarrow \mathbf{R}^+$ and is called the Deletion or Erasure Map.

The inter-string distance is called the Levenshtein distance if for all $a,b \in \mathbf{A}$ these distances are: $d_s(a, b)$ is 1 if $a \neq b$ and is 0 if $a = b$, and $d_i(a) = d_e(a) = 1$ for all a.

A more interesting and novel assignment of the distances is the parametric distances recently introduced by Bunke *et al* [2]. In this case, for all $a, b \in \mathbf{A}$ the substitution distance is r if $a \neq b$ and is 0 if $a =b$. The parametric string distance has some amazingly interesting properties derived in [2]. The assignment of 'r' and the application of the inter-string distance in PR has also been alluded to in [2].

If, however, the elementary symbol edit distances are *symbol* dependent, the distance is called the Generalized Levenshtein Distance. The question of how the elementary symbol edit distances can be assigned is relatively open; indeed, they can be parametrically assigned as in [2] or can be related to the inter-symbol confusion probabilities via their negative logarithms as recommended in [11,20]. The explicit form of the individual edit distances often takes the form :

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

$d_e(a) = -\ln [\, \Pr(a \text{ is deleted}) / \Pr(a \rightarrow a) \,]$

$d_i(a) = K_i \cdot d_e(a)$, where $K_i$ is an empirically determined constant.

The fundamental problem that arises from all the above three assignment strategies is that the final classified string obtained using such edit distances has *no probabilistic significance* except in some rather simple cases. Furthermore, if D(X, Y) is the edit distance associated with editing X to Y, the latter has no explicit relationship to $\Pr(X \rightarrow Y)$ except in a few rather trivial cases.

A little insight into the problem would reveal that the fundamental question which traditional strategies avoid is one of stochastically modeling the structural behaviour of the patterns. Viewed from a reverse engineering (black-box) perspective this question is one of specifying how the individual patterns from the various classes *could have been generated*, an understanding of which could lead to the designing of optimal classifiers. This is the central problem studied in this paper.

In this paper we shall present a new model, $\mathfrak{M}^*$ for noisy channels which transfer (or rather, carry) symbolic data, garbling it with *arbitrarily distributed* substitution, deletion and insertion errors. To our knowledge, this is the first generalized model of its type. All of the results claimed in this paper are rigorously proved in the unabridged paper [19]. They are omitted here in the interest of brevity. The unabridged paper also contains a general survey of the various alphabet and dictionary representations useful in syntactic PR.

When the dictionary is prohibitively large, problem analysts tackle the problem by modeling the dictionary using a stochastic string generation mechanism. The most elementary model is the one in which only the unigram (single character) probabilities of the dictionary are required [3,9,16]. This model is also referred to as the Bernoulli Model. A generalization of this is the Markovian (bigram) Model [1,3,9,16,21-23] where the probability of a particular symbol occurring depends on the previous symbol. As opposed to the stochastic models given for dictionaries, in this paper we shall consider the *channel* as an excited random string generator. Explicitly, we shall consider the channel as a generator whose input is a string U and whose output is the *random* string Y. The model for the channel is that Y is obtained by mutating U with an arbitrary sequence of string deforming substitution, deletion and insertion operations. Viewed from the perspective of these edit operations, this is a "distant" relative of Viterbi-type algorithms [6,16,21,23].

Our paper is a generalization of the classic paper of Bahl *et. al.* [1]. In addition to the properties of the channel described in [1], ours is functionally complete even though the distribution for the number of insertions is not necessarily a mixture of geometric distributions. Also, our model is stochastically consistent even though the parameters of the garbling operations are completely arbitrary. Although not explicitly stated, it is easy to verify that the latter is tacitly assumed to be the distribution for the number of insertions for the hidden Markov models used in text, character and texture classification. Finally, and most importantly, if the input is itself an element of a dictionary, the technique for computing the probability Pr[Y|U] can be utilized in a Bayesian way to compute the *a posteriori* probabilities. Thus we can obtain an information theoretic, minimum probability of error pattern classification rule independent of the model used for the dictionary itself.

## 2. Notation

Let A be a finite alphabet, and $A^*$ be the set of strings over A. $\lambda \notin$ A is the null symbol. A string $X \in A^*$ of the form $X=x_1x_2...x_N$ is said to be of length $|X| = N$. Its prefix of length i will be written as $X_i$, $i < N$. Upper case symbols represent strings, and lower case symbols, elements of the alphabet under consideration.

Let Y' be any string in $(A \cup \{\lambda\} )^*$, the set of strings over $(A \cup \{\lambda\})$. The string Y' is called an output edit sequence. The operation of transforming a symbol a $\in$ A to $\lambda$ will be used to represent the deletion of the symbol a. To differentiate between the deletion and insertion operation, the symbol $\xi$ is introduced. Let X' be any string in $(A \cup \{\xi\})^*$, the set of strings over $(A \cup \{\xi\})$. The string X' is called an input edit sequence. Observe that $\xi$ is distinct from $\lambda$, the null symbol. Transforming $\xi$ to b $\in$ A will represent the insertion of b.

The Output Compression Operator, $C_O$, is a function from $(A \cup \{\lambda\})^*$ to $A^*$. $C_O(Y')$ is Y' with all the occurrences of $\lambda$ removed. Note that $C_O$ preserves the order of the non-$\lambda$ symbols in Y'. Thus, if Y'=f$\lambda$o$\lambda$r, $C_O$(Y')=for. Analogously, the Input Compression Operator, $C_I$ is a function from $(A \cup \{\xi\})^*$ to $A^*$. $C_I(X')$ is X' removes all the occurrences of $\xi$, and preserves the order of the non-$\xi$ symbols in X'.

For every pair (U,Y), U,Y$\in$ $A^*$, the finite set $\Gamma(U,Y)$ is defined by means of the compression operators $C_I$ and $C_O$, as a subset of $(A \cup \{\xi\})^*$ x $(A \cup \{\lambda\})^*$ as :
$\Gamma(U,Y) = \{(U', Y') \mid (U', Y') \in (A \cup \{\xi\})^* \times (A \cup \{\lambda\})^*$, and each (U',Y') obeys}

(i) $C_I(U') = U$ ; $C_O(Y') = Y$

(ii) $|U'| = |Y'|$

(iii) For all $1 \leq i \leq |U'|$, it is not the case that $u'_i = \xi$ and $y'_i = \lambda$. (1)

By definition, if (U', Y') $\in \Gamma(U,Y)$, then, Max[ $|U|, |Y|$ ] $\leq |U'| = |Y'| \leq |U| + |Y|$.

The meaning of the pair (U', Y') $\in \Gamma(U,Y)$ is that it corresponds to one way of editing U into Y, using the edit operations of substitution, deletion and insertion. The edit operations themselves are specified for $1 \leq i \leq |Y'|$, as $(u'_i, y'_i)$, which represents the transformation of $u'_i$, to $y'_i$. $\Gamma(U,Y)$ is an exhaustive enumeration of the set of all the ways by which U can be edited to Y using the edit operations of substitution, insertion and deletion without destroying the order of the occurrence of the symbols in U and Y. We do not permit the channel to delete an inserted or substituted symbol.

**Lemma O.**

The number of elements in the set $\Gamma(U,Y)$ is given by :

$$\left|\Gamma(U,Y)\right| = \sum_{k=\mathrm{Max}(0,|Y|-|U|)}^{|Y|} \frac{(|U|+k)!}{(k!\,(|Y|-k)!\,(|U|-|Y|+k)\,!)} \tag{2}$$

**Proof :** The theorem is proved in the unabridged paper. [19] ♦♦♦

## 3. Modeling/Synthesis -- The String Generation Process

We now describe $\mathfrak{M}^*$, the model by which Y is generated from U $\in A^*$.

First of all we assume that $\mathfrak{M}^*$ utilizes a probability distribution G over the set of positive integers. The random variable in this case is referred to as Z and is the number of insertions that are performed in the mutating process. G is called the *Quantified* Insertion Distribution, and in the most general case, can be conditioned on the input string U. The quantity G(z|U) is the probability that Z =z given that U is the input word. Thus, the sum of G(z|U) over all feasible values of z is unity.

The second distribution that $\mathfrak{M}^*$ utilizes is the distribution Q over the alphabet under consideration. Q is called the *Qualified* Insertion Distribution. The quantity Q(a) is the probability that a $\in$ A will be the inserted symbol conditioned on the fact that an insertion operation is to be performed. The sum of Q(a) over all a $\in$ A is unity.

Apart from G and Q, the final distribution which $\mathfrak{M}^*$ utilizes is a probability distribution S over A x (A $\cup$ $\{\lambda\}$). S is called the Substitution and Deletion Distribution. For b $\in$ (A $\cup$ $\{\lambda\}$), the quantity S(b|a) is the conditional probability that the given symbol a $\in$ A in the input string is mutated by a stochastic substitution or deletion. S(c|a) is the conditional probability of a$\in$A being substituted for by c $\in$ A, and analogously, S($\lambda$|a) is the conditional probability of a $\in$ A being deleted. Observe that S has to satisfy the following constraint for all a $\in$ A :

$$\sum_{b\in(A\cup\{\lambda\})} S(b|a) = 1. \tag{3}$$

Using the above distributions we now informally describe the model for the garbling mechanism (or equivalently, the string generation process). Let |U| = N. Using the distribution G, the generator randomly determines the number of symbols to be inserted. Let Z be random variable denoting the number of insertions that are to be inserted in the mutation. Based on the random choice let us assume that Z takes the value z. The algorithm then determines the position of the insertions among the individual symbols of U. This is done by randomly generating an input edit sequence $U' \in (A \cup \{\xi\})^*$ with each of the $((N+k)!/(k!\ N!))$ possible strings are equally likely.

Note that $C_I(U')$ is U and that the positions of the symbol $\xi$ in U' represents the positions where symbols will be inserted into U. The occurrences of $\xi$ are now transformed independently into the individual symbols of the alphabet using Q. Finally, the non-inserted symbols in U' are substituted for or deleted using S.

This defines the model $\mathfrak{M}^*$ completely. The above process is formalized below.

```
Algorithm $\mathfrak{M}^*$_GenerateString
Input :  The word U and the distributions G, Q and S.
Output : A random string Y which garbles U with traditional mutations.
Method:
    1. Using G randomly determine z, the number of symbols to be inserted in U.
    2. Randomly generate an input edit sequence $U' \in (A \cup \{\xi\})^*$ by randomly
       determining the positions of the insertions among the symbols of U.
    3. Randomly independently transform the $\xi$'s into symbols of A using Q.
    4. Randomly independently substitute or delete the non-inserted symbols in U'
       using S.
END Algorithm $\mathfrak{M}^*$_GenerateString
```

A graphical display of the channel modeling and a detailed example of the garbling process is included in [19]. We shall now derive its analytic properties.

Let $|U| = N$ and $|Y| = M$. Then, the following results are true :

**Theorem I**

If the edit operations occur independently $\Pr[Y|U]$, the probability of receiving Y from $\mathfrak{M}^*$ given that U is transmitted has the form :

$$\Pr[Y|U] = \sum_{z=\mathrm{Max}(0,M-N)}^{M} \frac{G(z)\ (N!\ z!)}{((N+z)!)} \sum_{U'} \sum_{Y'} \prod_{i=1}^{N+z} p(y'_i|u'_i), \qquad (4)$$

in which (a) $y'_i$ and $u'_i$ are the symbols of Y' and U' respectively, (b) $p(y'_i|u'_i)$ is interpreted as $Q(y'_i)$ if $u'_i = \xi$, and, (c) $p(y'_i|u'_i)$ is interpreted as $S(y'_i|u'_i)$ if $u'_i \neq \xi$. Furthermore, the framework is both functionally complete and consistent.

**Proof :** The proof is quite intricate and is found in the unabridged paper [19]. ♦ ♦ ♦

We shall now demonstrate the efficient computation of $\Pr[Y|U]$.

## 4. Analysis : Computing P[Y|U] Efficiently

Consider the problem of $\mathfrak{M}^*$ editing U to Y, where $|U|=N$ and $|Y|=M$. Suppose we edit a prefix of U into a prefix of Y, using exactly i insertions, e deletions and s substitutions. Since the number of edit operations are specified, this corresponds to editing $U_{e+s} = u_1 \dots u_{e+s}$ into $Y_{i+s} = y_1 \dots y_{i+s}$. Let $\Pr[Y_{i+s}|U_{e+s}\ ; Z=i]$ be the probability of obtaining $Y_{i+s}$ given that $U_{e+s}$ was the original string, and that exactly i insertions took place in garbling. Then, by definition,

$$\Pr[Y_{i+s}|U_{e+s}\ ; Z=i] = 1 \qquad \text{if } i=e=s=0 \qquad (5)$$

To obtain an explicit expression for the above quantity for values of i, e and s which are non-zero, we have to consider all the possible ways by which $Y_{i+s}$ could have been obtained from $U_{e+s}$ using exactly i insertions. Let r=e+s and q=i+s. Let

$\Gamma_{i,e,s}(U,Y)$ be the subset of the pairs in $\Gamma(U_r,Y_q)$ in which every pair corresponds to i insertions, e deletions and s substitutions. Since we shall consistently be using the strings U and Y, $\Gamma_{i,e,s}(U,Y)$ will be referred to as $\Gamma_{i,e,s}$. Using (4),

$$\Pr[Y_{i+s}|U_{e+s}\,;Z=i] = \frac{(s+e)!\ i!}{(s+e+i)!} \sum_{(U'_r,Y'_q)} \prod_{j=1}^{|U'_r|} p(y'_{qj}|u'_{rj}), \quad (6)$$

if i, e or s > 0, and, $(U'_r,Y'_q)$ is an arbitrary element of $\Gamma_{i,e,s}$, with $u'_{rj}$ and $y'_{qj}$ as the jth symbols of $U'_r$ and $Y'_q$ respectively.

Let W(.,.,.) be the array whose general element W(i,e,s) is the sum of the product of the probabilities associated with the general element of $\Gamma_{i,e,s}$ defined as :

$$\begin{aligned} W(i,e,s) &= 0, && \text{if i,e or s} <0 \\ &= \frac{(s+e+i)!}{i!\ (s+e)!} \Pr[Y_{i+s}\,|U_{e+s}\,;Z=i] && \text{otherwise} \qquad (7)\end{aligned}$$

Using the expression for $\Pr[Y_{i+s}|U_{e+s}\,;Z=i]$ we obtain the explicit form of W(i,e,s) for all i, e, s ≥ 0 below.

$$\begin{aligned} W(i,e,s) &= 1, && \text{if } i=e=s=0 \\ &= \sum_{(U'_r,Y'_q)} \prod_{j=1}^{|U'_r|} p(y'_{qj}|u'_{rj}), && \text{if i, e or s} > 0 \qquad (8)\end{aligned}$$

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

$$\text{Max}[0,M-N] \le i \le q \le M\,;\,0 \le e \le r \le N\,;\,0 \le s \le \text{Min}[M,N] \quad (9)$$

Triples (i,e,s) which satisfy these constraints are termed as "feasible". Let, $H_i=\{\ j \mid \text{Max}[0,M-N] \le j \le M\}$, $H_e=\{j \mid 0\le j\le N\}$,and $H_s=\{j|0\le j\le \text{Min}\ [M, N]\}$. $H_i$, $H_e$ and $H_s$ are called the set of permissible values of i, e and s. A triple (i,e,s) is feasible if apart from $i\in H_i$, $e\in H_e$, and $s\in H_s$, $i + s \le M$, and $e + s \le N$.

The next result specifies the permitted forms of the triples for editing $U_r$ to $Y_q$.

**Theorem II**

To edit $U_r$, the prefix of U of length r, to $Y_q$, the prefix of Y of length q, the set of feasible triples is given by $\{\ (i, r-q+i,\ q-i\ ) \mid \text{Max}\ [0, q-r] \le i \le q\ \}$.

**Proof :** The proof is included in the unabridged paper [19]. ♦♦♦

The following theorem (proved in [19]) states the recursive property of W(.,.,.).

**Theorem III**

Let W(i,e,s) be defined as in (8) for any two strings U and Y. Then, for all non-negative i,e and s,

$$W(i,e,s) = W(i-1,e,s).p(y_{i+s}|\xi) + W(i,e-1,s).p(\lambda|u_{e+s}) + W(i,e,s-1).p(y_{i+s}|u_{e+s})$$

where p(b|a) is interpreted as in (4). ♦♦♦

We compute $\Pr[Y|U]$ as a weighted combination of elements of W(.,.,.) as below.

**Theorem IV**

If $h(i)= G(i) \dfrac{N!\, i!}{(N+i)!}$, $Pr[Y|U]$ can be evaluated from the array W(i,e,s) as :

$$Pr[Y|U] = \sum_{i=Max(0,M-N)}^{M} h(i).W(i, N-M+i, M-i).$$

**Proof :** The proof is included in [19]. ♦♦♦

To evaluate $Pr[Y|U]$ we make use of the fact that although it has no known recursive properties, W(.,.,.), which is closely related to it obeys Theorem III. The Algorithm EvaluateProbabilities which we now present, evaluates the array W(.,.,.) for all permissible values of the variables i, e and s subject to the constraints of Theorem II. Using the array W(i,e,s) it evaluates $Pr[Y|U]$ by adding up the weighted contributions of the pertinent elements in W(.,.,.). This is formalized below.

**Algorithm EvaluateProbabilities**
**Input:** The strings $U=u_1u_2...u_N$, $Y=y_1y_2...y_M$, and distributions G, Q and S.
**Output:** W(i,e,s) for all permissible i, e and s and the probability Pr[Y|U].
**Method :**

```
R=Min [ M, N] ; W(0,0,0)=1
Pr[Y|U] = 0
For i=1 to M Do
    $W(i,0,0) = W(i-1,0,0). Q(y_i)$
For e=1 to N Do
    $W(0,e,0) = W(0,e-1,0).S(\lambda|u_e)$
For s=1 to R Do
    $W(0,0,s) = W(0,0,s-1).S(y_s|u_s)$
For i=1 to M Do
    For e=1 to N Do
        $W(i,e,0) = W(i-1,e,0).Q(y_i) + W(i,e-1,0).S(\lambda|u_e)$
For i=1 to M Do
    For s=1 to M-i Do
        $W(i,0,s) = W(i-1,0,s).Q(y_{i+s}) + W(i,0,s-1).S(y_{i+s}|u_s)$
For e=1 to N Do
    For s=1 to N-e Do
        $W(0,e,s) = W(0,e-1,s).S(\lambda|u_{s+e}) + W(0,e,s-1).S(y_s|u_{s+e})$
For i=1 to M Do
    For e=1 to N Do
        For s=1 to Min[(M-i) , (N-e)] Do
            $W(i,e,s)=W(i-1,e,s).Q(y_{i+s})+W(i,e-1,s).S(\lambda|u_{e+s})+W(i,e,s-1).S(y_{i+s}|u_{e+s})$
For i=Min[0 , M-N] to M Do
    $Pr[Y|U] = Pr[Y|U] + G(i) .((i! N!)/(N+i)!). W(i,N-M+i,M-i)$
END Algorithm EvaluateProbabilities
```

Obviously, the above process requires cubic time and space respectively. A *more efficient* but intricate algorithm to compute it is included in [19].

## 4.1 An Information Theoretic Bound

Using the model $\mathfrak{M}^*$ it is easy to see how optimal syntactic pattern recognition can be obtained. Indeed, if the distributions G, Q and S are known (the inference (estimation) problem of these distributions remains open) PR can be achieved by evaluating the string $U^*$ which maximizes the probability $\Pr[Y|U]$ over all U in the dictionary. Seen from a Bayesian perspective this would be equivalent to computing the *a posteriori* probabilities if all the strings are equally likely *a priori*, and thus yield optimal, minimum probability of error pattern classification. In a non-Bayesian approach this represents a maximum likelihood pattern classification scheme.

We now show that the PR obtained by utilizing $\mathfrak{M}^*$ is not only optimal - it also attains the information theoretic upper bound. This is done by using arguments analogous to those used in developing bounds for sorting and other computer science operations. Observe that this presupposes that we compare $\mathfrak{M}^*$ with all other channel models which have the same common underlying garbling philosophy.

**Theorem V**

If transmitted symbols can only be substituted for or deleted and received symbols are obtained as either a result of transmitted symbols being substituted for or as inserted symbols, then, for specific distributions G, Q and S, the garbling model $\mathfrak{M}^*$ attains the information theoretic bound for recognition accuracies.

**Proof :** The theorem is proved in the unabridged paper. [19] ♦♦♦

## 5. Experimental Results

To investigate the power of our new model 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 :

(i) Algorithm_LD : A PR scheme which used any traditional editing [10,11,14,17,20,22,24] algorithm and unit inter-symbol costs.

(ii) Algorithm_GLD : A PR scheme which used any traditional editing [10,11,14,17,20,22,24] algorithm using symbol-dependent costs.

The dictionary consisted of 342 words obtained as a subset of the 1023 most common English words [4] augmented with words used in computer literature. The length of the words was greater than or equal to 7 and the average length of a word was approximately 8.3 characters. From these, two sets (**SA** and **SB** respectively) of 1026 noisy strings were generated using the method described in Section 2. 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 [19]. The statistics associated with the sets **SA** and **SB** are given below in Table I. A subset of some of the words in SA is given Table II.

| | SA | SB |
|---|---|---|
| Number of insertions | 1872 (1.825) | 2142 (2.088) |
| Number of deletions | 418 (0.407) | 414 (0.404) |
| Number of substitutions | 769 (0.750) | 822 (0.801) |
| Total number of errors | 3059 (2.981) | 3378 (3.292) |
| Percentage error | 36.00% | 39.75% |

**Table I:** Noise statistics of the sets SA and SB. The figures in brackets are the average number of errors per word.

| Original word (dictionary) | Noisy word | Total number of errors |
|---|---|---|
| administration | sdmlnistratib | 5 |
| advance | ewafawdvxsance | 7 |
| advantage | taodbivawafxe | 8 |
| affairs | kafruvkfnixsrs | 9 |
| artillery | vuaegrdtuilllordiery | 11 |
| beginning | ssbehsimgjninmmg | 10 |
| cooperation | coeopewryaueipxn | 8 |
| executive | yslxvkrcutivoe | 7 |
| followed | zdslfdxllwkedekuid | 14 |
| sitting | psxruttoing | 6 |
| strength | mzeckieeotrenxsbth | 13 |
| striking | yvysatqrickinwet | 10 |
| victory | vbtctlavrdy | 7 |
| without | xvwigobuhnout | 8 |

**Table II:** A subset of the dictionary, noisy strings and error characteristics.

The three algorithms, Levenshtein Distance (Algorithm_LD), the Generalized Levenshtein Distance (Algorithm_GLD) and our algorithm (Algorithm_OPT_PR), were tested with the sets of 1026 noisy words, **SA** and **SB**. The results obtained in terms of the recognition accuracy for the two sets are tabulated below in Tables III. Note that our scheme far outperforms the traditional string correction and GLD algorithms. The reader should observe that, as in all PR problems, it is much harder to increase the recognition accuracy at the higher end of the spectrum.

| **Algorithm** | **Accuracy (SA)** | **Accuracy (SB)** |
|---|---|---|
| Algorithm_LD | 94.93% | 93.76% |
| Algorithm_GLD | 96.00% | 94.35% |
| Algorithm_OPT_PR | 97.66% | 96.49% |

**Table III:** The recognition results obtained from the noisy data sets **SA** and **SB**.

## 6. Conclusions

In this paper we have presented a formal foundation for designing optimal and information theoretic, minimum probability of error syntactic pattern recognizers. We have done this by presenting a new model for noisy channels which permit arbitrarily distributed substitution, deletion and insertion errors. The scheme has been shown to be functionally complete and stochastically consistent. Apart from presenting the model we have specified how Pr[Y|U] can be efficiently computed to yield minimum probability of error syntactic PR. Experimental results which involve dictionaries with strings of lengths between 7 and 14 with an overall average noise of 39.75 % demonstrate the superiority of our system over existing methods.

## Abridged List of References

1. R. L. Bahl and F. Jelinek, Decoding with channels with insertions, deletions and substitutions with applications to speech recognition, *IEEE T. Inf. Th.*, IT-21:404-411 (1975).
2. Bunke, H. and Csirik, J, Parametric string edit distance and its application to pattern Recognition, *IEEE T. Syst., Man and Cybern.*, SMC-25:202-206 (1993).

3. L. Devroye, W. Szpankowski and B. Rais, A note on the height of suffix trees, *SIAM J. of Computing,* 21:48-54, (1992).
4. G. Dewey, *Relative Frequency of English Speech Sounds*, Harvard Univ. Press, (1923).
5. R. O. Duda, P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley & Sons, 1973.
6. G.D. Forney, The Viterbi Algorithm, *Proceedings of the IEEE,* Vol. 61. (1973).
7. K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1972.
8. P. A. V. Hall and G.R. Dowling, Approximate string matching, *Comp. Sur.*, 12:381-402 (1980).
9. P. Jacquet and W. Szpankowski, Analysis of digital tries with markovian dependencies, *IEEE T. Inf. Th.*, IT-37:1470-1475 (1991).
10. R. L. Kashyap and B. J. Oommen, A common basis for similarity and dissimilarity measures involving two strings, *Int. J. Comp. Math.*,17-40 (1983).
11. 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, *Inf. Sci.*, 23(2):123-142 (1981).
12. R. L. Kashyap, and B. J. Oommen, String correction using probabilistic methods, *Pattern Recognition Letters*, 147-154 (1984).
13. R. Lowrance and R. A. Wagner, An extension of the string to string correction problem, *J. Assoc. Comput. Mach.*, 22:177-183 (1975).
14. A. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals, *Soviet Phys. Dokl.*, 10:707-710 (1966).
15. W. J. Masek and M. S. Paterson, A faster algorithm computing string edit distances, *J. Comput. System Sci.*, 20:18-31 (1980).
16. D. L. Neuhoff, The Viterbi algorithm as an aid in text recognition, *IEEE T. Inf. Th.*, 222-226 (1975).
17. T. Okuda, E. Tanaka, and T. Kasai, A method of correction of garbled words based on the Levenshtein metric, *IEEE T. Comput.*, C-25:172-177 (1976).
18. B. J. Oommen, Recognition of noisy subsequences using constrained edit distances, *IEEE T. on Pattern Anal. and Mach. Intel.*, PAMI-9:676-685 (1987).
19. B. J. Oommen and R. L. Kashyap, A Formal Theory for Optimal and Information Theoretic Syntactic Pattern Recognition. Unabridged Version of the present paper. (Submitted for Publication).
20. D. Sankoff and J. B. Kruskal, *Time Warps,String Edits and Macromolecules: The Theory and practice of Sequence Comparison*, Addison-Wesley (1983).
21. R. Shinghal, and G. T. Toussaint, Experiments in text recognition with modified Viterbi algorithm, *IEEE T. on Pat. Anal. and Mach. Intel.*, 184-192 (1979).
22. S. Srihari, *Computer Text Recognition and Error Correction*, IEEE Computer Press, (1984).
23. A. J. Viterbi, Error bounds for convolutional codes and an asymptotically optimal decoding algorithm, *IEEE T. on Inf. Th.*, 260-26 (1967).
24. R. A. Wagner and M. J. Fisher, The string to string correction problem, *J. Assoc. Comput. Mach.*, 21:168-173 (1974).