# An imperfect string matching experience using deformed fuzzy automata

J.J. Astrain[1], J.R. Garitagoitia[1], J. Villadangos[2],
F. Fariña[1], and J.R. González de Mendívil[1]

[1] Dpt. Matemática e Informática
[2] Dpt. Automática y Computación
Universidad Pública de Navarra
Campus de Arrosadía, 31006 Pamplona, Spain
{josej.astrain,joserra,jesusv,fitxi,mendivil}@unavarra.es

**Abstract.** This paper presents a string matching experience using deformed fuzzy automata for the recognition of imperfect strings. We propose an algorithm based on a deformed fuzzy automaton that calculates a similarity value between strings having a non-limited number of edition errors. Different selections of the fuzzy operators for computing the deformed fuzzy automaton transitions allows to obtain different string similarity definitions. The selection of the parameters determining the deformed fuzzy automaton behavior is obtained via genetic algorithms.

## 1 INTRODUCTION

The applications of pattern recognition based on structural and syntactic methods have to cope with the problem of recognizing strings of symbols that do not fit with any one of the defined patterns. The problem is usually resolved by defining a function that allows measuring the similarity (distance or dissimilarity) between pairs of strings [14].

This approach consists basically in the construction of a dictionary by using the representatives of the classes. For the recognition of an erroneous string, the string is compared with every sample in the dictionary. A similarity value (or distance) is then computed with every one of the samples in terms of the edit operations (normally insertion, deletion and substitution of a symbol) needed for the transformation of a string into another one. Finally, the string is classified in the class whose representative obtains greater similarity (less distance). In the literature different definitions for string distance, as well as algorithms that calculate them, are proposed [9, 10, 1, 12, 15].

Recognition rates of pattern classification systems are sometimes low because different steps in the recognition process are considered in isolated and sequential way. Each step makes a decision at its own level, and it could be erroneous. Later steps have to do their task taking into account erroneous data, which invariably affects the recognition rate. For instance, text recognizers usually contain steps for segmentation, isolated character recognition, word recognition,

sentence recognition, and so on, which act sequentially in such a way that for the classification of a character it is not taken into account the word in which the character is located. Thus, in the work of a particular step, the information level pertaining to a later step is not considered.

In order to improve recognition rates and, at the same time, to keep the work separate in different steps, the use of fuzzy symbols has been proposed [3, 2, 7, 16] as an adequate way to represent the ambiguity in the classification of previous steps. A fuzzy symbol is a fuzzy set defined over the total set of symbols, with its membership function representing the similarity between the observed symbol and every symbol in the set. Within the concept of string matching, the problem of classifying strings presents now a higher complexity because it is necessary to work with strings containing fuzzy symbols which, in addition to edition errors, present imprecise information in the fuzzy symbols.

We propose a fuzzy automaton for the classification of strings containing any amount of errors, that can use different fuzzy operations ($t$-conorms and $t$-norms) [7] implementing different distance (in fact, similarity) definitions. Deforming the fuzzy automaton [11], it can be performed a string matching accepting as inputs imperfect strings of fuzzy symbols. Given a particular problem we have to choose adequately the fuzzy values of the transitions as well as the values of the $t$-conorm/$t$-norm parameters. The problem of selecting such parameters is an optimization problem. In this paper, we propose tuning the parameters of the fuzzy automaton using a genetic algorithm [5].

In order to validate our method, an experimental system for the recognition of texts has been developed. The character classifiers provide fuzzy characters that permit to represent the ambiguity of the results obtained by the classifiers of isolated characters [17, 3, 2]. The contextual processing step of the system makes the comparison between the strings of fuzzy characters and the strings of a dictionary representing the lexicographical context. This step is implemented by the proposed deformed fuzzy automaton. The experimental results show that the deformed system has a great capacity for recovering errors. This fact leads to high word recognition rates even in the presence of a high amount of errors.

The rest of the paper is organized as follows. In Section 2 the algorithm of the deformed fuzzy automaton that computes similarity between strings is introduced. Section 3 is devoted to show experimental results obtained when the deformed fuzzy automaton is applied to a problem of text recognition. Section 4 presents the conclusions. Finally, references end the paper.

## 2 STRING SIMILARITY BASED ON DEFORMED FUZZY AUTOMATA

This section is devoted to the introduction of a deformed fuzzy automaton that computes string similarity between two strings, the *observed* and the *pattern* strings respectively.

Initially, a finite deterministic automaton $M(\omega)$ which recognizes the pattern string $\omega$ is defined. This automaton will not accept the observed string $\alpha$ unless

$\alpha = \omega$. Second, this automaton is modified as it was introduced in [4], obtaining a fuzzy automaton $MF(\omega)$ in order to accept every observed string $\alpha$ providing a value for the similarity between $\alpha$ and $\omega$. This modification includes the management of all the possible edit operations (insertion, deletion, and change of a symbol).

When the symbols of an observed string are processed by a isolated character classifier (ICC), a fuzzy symbol representing the ambiguity of each symbol classification is obtained. The value of this fuzzy symbol represents the degree of proximity of the pattern symbol to the observed symbol. Then, it is necessary to work with imperfect strings of fuzzy symbols, that is, strings which may contain insertion, deletion, and substitution of fuzzy symbols (edition errors).

Third, in order to work with imperfect fuzzy symbols, the fuzzy automaton is deformed as it was explained in [4]. Then, an algorithm for the computation of the deformed fuzzy automaton is introduced (see figure 1).

**Input**: $M(\omega) = (Q, \Sigma, \delta, q_0, \{q_n\})$, $Q = \{q_0, \ldots, q_n\}$, $n$ is the length of the string $\omega$
the lists $\mu_{d_x^{q_j}}$, $\mu_{c_{xa}^{q_{i-1}q_i}}$ and $\mu_{i_a^{q_{i-1}q_i}}$, $\forall a, x \in \Sigma$, $\forall i : 1 \ldots n$
$\forall j : 0 \ldots n$
$\tilde{\alpha}$ observed string, length $m$ ($\tilde{y}_k$ k-th symbol of $\tilde{\alpha}$)
**Output**: $MD(\omega, \tilde{\alpha})$
where $MD(\omega) = (Q, \Sigma, \mu, \sigma, \eta, \tilde{\mu})$ is the deformed fuzzy automaton for $M(\omega)$

**algorithm** computation
  initialstate;
  $\forall k : 1..m$:
    transition($k$);
    $\varepsilon$-closure;
  decision
**endalgorithm**
**procedure** initialstate
  $\forall i : 1..n$:
    $Q(q_i)$:= 0;
  $Q(q_0)$:= 1;
  $\varepsilon$-closure
**endprocedure**
**procedure** decision
  $MD(\omega, \tilde{\alpha})$:= $Q(q_n)$
**endprocedure**

**procedure** transition($k$)
  $\forall i : 0..n$:
    $C_1$:=$Q(q_i) \otimes (\oplus_{x \in \Sigma}(\mu_{d_x^{q_i}} \otimes \mu_{\tilde{y}_k}(x)))$;
    $C_2$:=$Q(q_{i-1}) \otimes (\oplus_{x \in \Sigma}(\mu_{c_{xa}^{q_{i-1}q_i}} \otimes \mu_{\tilde{y}_k}(x)))$ [a];
      where $a$ is that $\delta(q_{i-1}, q_i, a) = 1$.
    $Q'(q_i)$:= $C_1 \oplus C_2$;
  $\forall i : 0..n$: $Q(q_i)$ := $Q'(q_i)$
**endprocedure**
**procedure** $\varepsilon$-closure
  $\forall i : 1..n$:
    $Q(q_i)$:=max$(Q(q_i), Q(q_{i-1}) \otimes \mu_{i_a^{q_{i-1}q_i}})$;
      where $a$ is that $\delta(q_{i-1}, q_i, a) = 1$.
**endprocedure**

[a] $Q(q_{-1}) = 0$.

**Fig. 1.** Algorithm for the deformed fuzzy automaton.

The main program is given by the algorithm computation. The algorithm starts by building the initial fuzzy state which is given by $\sigma$ (from the defini-

tion of $MD(\omega)$). The body of the algorithm is a loop in which the procedures `transition` and $\varepsilon$-`closure` are executed for every fuzzy symbol $\tilde{y}_k$ of the observed string $\tilde{\alpha}$. These two procedures compute the state transition by a fuzzy symbol and the empty string, respectively. Finally, the procedure `decision`, computes the final fuzzy state given by $\eta$. There are only two possible transitions to reach state $q_i$: One from state $q_{i-1}$, representing a substitute operation ($C_2$ in figure 1), the other one from the same state $q_i$, representing a delete operation ($C_1$ in figure 1). The procedure $\varepsilon$-`closure` represents insert operations.

Operators $\otimes$ and $\oplus$ represent the $t$-norm and the $t$-conorm respectively. One can note that for each replacement of a $t$-conorm and a $t$-norm, a different fuzzy automaton is obtained, and thus, a different similarity operator. In [4] we show that when the operators $\oplus$ and $\otimes$ are replaced by the *maximum* and the *algebraic product* operators respectively, a similarity operator that calculates (transforming the output value as $-log(MF(\omega, \alpha))$ the Generalized Levenshtein Distance [9] is obtained. The use of different $t$-conorms and $t$-norms may be of interest for practical problems due to different aggregations they provide.

One can note that the string similarity provided by our deformed fuzzy automaton has a more flexible behavior because it makes use of a parametric $t$-norm/ $t$-conorm for computing the automata transitions. In the following, we use the Hamacher $t$-norm/ $t$-conorm and we denote $\gamma$ to the parameter that Hamacher uses to cover the space ($\gamma \geq 0$) of possible $t$-conorms and $t$-norms [7].

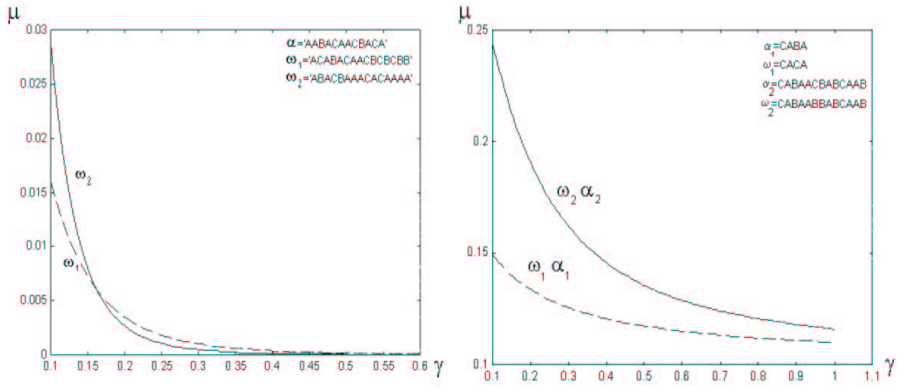


**Fig. 2.** *(Left)* Effects of changing the $\gamma$ parameter of the Hamacher $t$-norm/$t$-conorm when comparing the observed string $\alpha$ with two pattern strings $\omega_1$ and $\omega_2$. *(Right)* Avoiding the effects of the string length.

In the figure 2 *(left)*, we show the effects of changing the $\gamma$ parameter of the Hamacher $t$-norm/$t$-conorm when comparing the observed string $\alpha$ with two pattern strings $\omega_1$ and $\omega_2$ varying the value of the Hamacher's parameter. The figure shows that the final decision depends not only in the number of edit operations to transform $\alpha$ into $\omega_1$ or $\omega_2$ but in the similarity measure obtained

varying the $t$-norm/$t$-conorm. In fact, we could select the $t$-norm/$t$-conorm that gives the desired result.

The Levenshtein Edit Distance is dependant of the length of the strings. This fact has motivated to researches to introduce Normalized Edit Distances [10], that consider the relation between the number of errors and the length of the strings when their similarity is analyzed. A similar effect can be obtained with our deformed fuzzy automaton by using different values of the $t$-norm/$t$-conorm parameter for each target string length. Consider the example depicted by figure 2 *(right)*. It shows the results obtained for different values of Hamacher parameters when recognizing a short input string, $\alpha_1$, with one substitution error and a long input string $\alpha_2$ also with one substitution error. If we need that the errors in short strings be more significant than in long strings, we could assign to the short string a value of the Hamacher's parameter higher than the one to the long string. For example, selecting $\gamma = 1$ for computing the fuzzy automaton of $\omega_1$ while $\gamma = 0,2$ when computing the fuzzy automaton of $\omega_2$, we will obtain the desired result. The similarity between $\alpha_2$ and $\omega_2$ is higher than the similarity between $\alpha_1$ and $\omega_1$.

As conclusion, the membership values of the elementary edit operations and the value of the parameter of the selected parametric $t$-norm/$t$-conorm for each pattern string as the parameters of the system must to be considered. Those parameters must to be tuned in order to adjust the system to each concrete problem.

## 3 EXPERIMENTAL RESULTS

In order to evaluate the proposed method, an experimental system for hand-printed text recognition has been considered. In the experimental system, the contextual post processing [6] is implemented by the related deformed fuzzy automaton. The objective is to analyze the capability of the deformed system to correct the edition errors that are introduced during text creation (typographical and lexicographical errors) or by the previous stages of the system (segmentation and individual classification of the characters). In the experiment, the characters are classified by using a neural network, a perceptron with three layers trained by using the Back Propagation Training Algorithm [13]. This neural network has been trained by using 25 alphabets from 25 different authors. When an input character is obtained from the input text, the neural network produces a fuzzy symbol, $\tilde{y} = \{(x, \mu_{\tilde{y}}(x)) \mid x \in \Sigma\}$, where $\mu_{\tilde{y}}(x)$ is the neural output unit value associated with the character $x$ of the alphabet $\Sigma$ (26 letters).

The input texts are obtained from a large corpus of words known as *Brown Corpus* [8]. This corpus is formed by 15 different scope texts extracted from real life documents. We have taken a portion (about 6000 words) of each text. These texts are referred as *Brown-A*, *Brown-B*, ..., *Brown-R*. Taking the words of those texts, we have built-up the dictionaries *Dic-A*, *Dic-B*, ..., *Dic-R* (with an extension that ranges from 1268 to 2213 words).

### 3.1 Parameters of the system

The contextual post processing based on deformed system has been implemented by using the algorithm introduced in figure 1 modified to work with a set of words (dictionary), $D \subset \Sigma^*$. Several parameters of the algorithm must be fixed for the experiments like the fuzzy operations for computing transitions and the membership function values associated to transitions. The transitions have been computed using the Hamacher's $t$-conorm and $t$-norm [7].

We have selected the membership function values associated to transitions to be independent of symbols and states, so only three values $\mu_d$, $\mu_i$, $\mu_c$ are associated to delete, insert and substitute operation respectively.

In order to determine the values for the three edit operations and the $\gamma$ parameter, we use a genetic algorithm [5]. The genetic algorithm can be formalized as a function, $GA(O, D, MD, \{\mu\}, \gamma)$, where $O$ is the set of observed strings of fuzzy symbols; $D$ the set of patterns (dictionary); $MD$ the deformed fuzzy automaton; $\{\mu_d, \mu_i, \mu_c\}$ the membership values of the transitions and $\gamma$ the parameter of the parametric $t$-conorm/$t$-norm. The goal of the genetic algorithm is to determinate the parameter values that minimize the mean quadratic error of the system: $J(O) = (1/2) \sum_{\tilde{\alpha} \in O} ||target(\tilde{\alpha}) - out(\tilde{\alpha})||$. In that equation $out(\tilde{\alpha}) = [MD(\omega_1, \tilde{\alpha}), \ldots, MD(\omega_n, \tilde{\alpha})]$. Each component represents the similarity of $\alpha$ with one of the patterns. As $\tilde{\alpha}$ is associated to a unique pattern in the training set $O$, the desired output is a vector $target(\tilde{\alpha}) = [a_1, \ldots, a_i, \ldots, a_n]$, such that $target(\tilde{\alpha})[i] = 1$ if and only if $\tilde{\alpha}$ is an element of the class $\omega_i$; in other case, $target(\tilde{\alpha})[i] = 0$.

In order to evaluate the error of the fuzzy automaton when classifying a given observed string $\tilde{\alpha}$, we are going to consider as metric the euclidean distance: $||target(\tilde{\alpha}) - out(\tilde{\alpha})||$. The selected optimization function will allow to select those parameters that maximize the similarity between the observed string of fuzzy symbols, and its associated pattern also maximize the difference between the observed string and the other patterns.

We have used a genetic algorithm with 10 populations, evolving during 300 evolution steps. At each step, the best four populations remain in the system for the next step of evolution. The other six are generated by selecting pairs of populations. The crossing and mutation probabilities are 0.25 and 0.1 respectively.

In order to assure that the *deformed system* does not increase the number of erroneous words, we have made experiments without introducing errors. In those tests, we obtained a 100% of recognition rate. We conclude that the *deformed system* does not create errors by itself.

### 3.2 Experimental results

In a first experiment, we introduce texts with different error rates in order to examine the capability of the *deformed system* to correct those errors. Figure 3 presents the results obtained in this experiment (it shows the average values obtained after 50 experiments). Besides the error reduction rate (with its standard deviation), the table also shows the percentages of recognition rate (correct

| | % Error Rate | | | |
|---|---|---|---|---|
| | 31.3 | 44.7 | 71.6 | 87.7 |
| % Recognition Rate | 95.61 | 93.26 | 84.81 | 66.93 |
| **% Error Reduction Rate** | **86.0** | **84.9** | **78.8** | **62.4** |
| **Deviation (50 iterations)** | **0.723** | **0.702** | **0.575** | **0.550** |
| % Class. Error Reduction Rate | 95.4 | 94.6 | 90.7 | 74.7 |
| % Insertion Error Reduction Rate | 91.1 | 90.0 | 82.2 | 57.4 |
| % Deletion Error Reduction Rate | 74.1 | 73.5 | 67.9 | 51.4 |
| % Substitution Error Reduction Rate | 84.5 | 83.7 | 78.2 | 58.7 |
| Poss (Recognition Rate 99%) | 4 | 5 | 15 | 15 |

**Fig. 3.** Results obtained for different error rate. Input text: *Brown-A*. Dictionary: *Dic-A* (1725 words).

words), corrected classification errors, corrected insertion errors, corrected deletion errors and corrected substitution errors. The row labelled 'Poss' indicates the interval (when the output of the *deformed system* is given in the rank level [17]) where, more than 99% of the corrected words fall into. For example, if Poss= 4 we assure that more than the 99% of corrected words are between the first and the fourth position when the output of the automaton is given in the rank level. Such a rank output becomes useful when a higher level of context (i.e., syntactic or semantic analysis) is wanted to be used.

In a second experiment we have evaluated the influence of the text domain. Figure 4 shows the results obtained with the texts *Brown-B* (*Dic-B*) to *Brown-R* (*Dic-R*). Again, we introduce an error every 15 characters (*segment length*). This corresponds to an error rate from 25.4% to 34.3% depending on the text. We can see how the best correction rate is obtained with the *Brown-E* text. This is so because such a text has the lowest percentage (58.56%) of short words (less than six characters). Note that short words are difficult to correct because they have less context. On the other hand, the worst correction rate is obtained with the *Brown-N* text. This text has the highest percentage of short words (79.68%).

Finally, we made an experiment intended to evaluate the influence of fuzzy symbols and the fuzzy operations chosen for the transitions of the automata (see figure 5). In the first line the results correspond to the use of Levensthein Distance [9] in order to compare the observed strings with the words in the dictionary. In this case the observed strings are composed of individual characters, being those that present greater value in the membership function in the corresponding fuzzy symbols. The edit operation costs used by the algorithm [15] implementing the Levensthein Distance have unitary values. Therefore, an observed string will be classified as the word in the dictionary having the smallest number of edit operations.

As it was proved in [4], when the fuzzy automaton uses as $t$-conorm the *maximum* and as $t$-norm the *algebraic product*, it permits to calculate the Levensthein Distance giving appropriate values to the transitions modelling the edit

| | ER | RR | DRR | **ERR** | CLerr | Ierr | Derr | Cerr | Poss |
|---|---|---|---|---|---|---|---|---|---|
| Dic-B (2087 p.) | 30.223 | 95.100 | 0.249 | **83.787** | 95.237 | 88.435 | 70.806 | 81.987 | 5 |
| Dic-C (2213 p.) | 30.503 | 95.181 | 0.241 | **84.203** | 94.623 | 89.455 | 72.071 | 81.996 | 4 |
| Dic-D (1658 p.) | 30.345 | 95.338 | 0.288 | **84.636** | 95.255 | 89.033 | 73.199 | 82.591 | 4 |
| Dic-E (2063 p.) | 34.285 | 96.149 | 0.245 | **88.768** | 96.295 | 91.601 | 80.532 | 88.114 | 4 |
| Dic-F (1918 p.) | 30.347 | 95.361 | 0.227 | **84.175** | 94.686 | 89.168 | 73.147 | 83.101 | 5 |
| Dic-G (1899 p.) | 31.195 | 95.726 | 0.261 | **86.300** | 96.015 | 90.564 | 74.835 | 84.940 | 4 |
| Dic-H (1571 p.) | 32.139 | 96.031 | 0.212 | **87.649** | 96.000 | 91.890 | 77.714 | 86.120 | 4 |
| Dic-J (1480 p.) | 31.706 | 95.969 | 0.237 | **87.285** | 96.448 | 90.039 | 77.622 | 86.152 | 4 |
| Dic-K (1268 p.) | 30.790 | 96.132 | 0.253 | **87.439** | 96.124 | 91.055 | 78.311 | 85.961 | 4 |
| Dic-L (1449 p.) | 25.376 | 94.627 | 0.281 | **78.828** | 92.865 | 84.465 | 63.290 | 75.383 | 5 |
| Dic-M (1725 p.) | 28.181 | 94.908 | 0.243 | **81.933** | 93.957 | 87.968 | 67.204 | 79.543 | 5 |
| Dic-N (1447 p.) | 25.880 | 94.454 | 0.246 | **78.568** | 92.942 | 85.939 | 61.122 | 74.938 | 5 |
| Dic-P (1720 p.) | 26.943 | 94.478 | 0.235 | **79.504** | 93.219 | 85.723 | 63.619 | 76.255 | 5 |
| Dic-R (2078 p.) | 28.717 | 95.089 | 0.265 | **82.900** | 94.173 | 87.971 | 69.918 | 80.386 | 5 |

**Fig. 4.** Results (in percentage) obtained for different text domains. ER: Error Rate; RR: Recognition Rate; DRR: Standard Deviation of RR for 50 experiences; ERR: Error Reduction Rate; CLerr: Classification Error Reduction Rate; Ierr: Insertion Error Reduction Rate; Derr: Deletion Error Reduction Rate; Cerr: Substitution Error Reduction Rate; Poss: number of words needed for 99% recognition rate.

operations. In this way, giving values $\mu_i = 0.1$, $\mu_d = 0.1$, $\mu_c = 0.1$ one would obtain the same results as obtained by using the Levensthein Distance. However, we can provide a deformed system for this fuzzy automaton to work with strings of fuzzy characters. Therefore, it is possible to evaluate the influence by taking into account all the information contained in the fuzzy symbols. These results are shown in the second line of figure 5. The third line shows the results obtained with the deformed system using Hamacher's $t$-conorm and $t$-norm together with the used values for the transitions in previous experiments. This enables us to compare the influence of the election of fuzzy operations.

In figure 5 the average values obtained through a simulation with one hundred texts are shown. The best results are obtained with the deformed system method. Differences between Levensthein, max-prod and Hamacher are minor if the number of errors per word is low (error rate 31.3%), and major if the number of errors per word is high (error rate 71.6%). Differences between max-prod and Hamacher are relevant in terms of error reduction rate (near eight points). When a deformed system is used with a high number of errors per word, RR and ERR become higher.

## 4 CONCLUSIONS

In this work we have introduced a fuzzy method that allows recognizing imperfect strings of fuzzy symbols. The basic idea is to use a deformed system built

| | ER = 31.3% | | ER = 44.7% | | ER = 71.6% | |
|---|---|---|---|---|---|---|
| | RR | ERR | RR | ERR | RR | ERR |
| Levenshtein distance | 89.811 | 67.451 | 84.309 | 65.742 | 65.296 | 51.540 |
| Max-prod (deformed) | 93.014 | 77.516 | 90.049 | 77.650 | 80.597 | 72.833 |
| Hamacher (deformed) | 95.677 | 86.154 | 93.449 | 85.340 | 86.011 | 80.463 |

**Fig. 5.** Influence of fuzzy symbols and the $t$-conorm/$t$-norm in recognition rate. Input text: *Brown-A*. Dictionary: *Dic-A* (1725 words). ER: Error Rate; RR: Recognition Rate; ERR: Error Reduction Rate.

from a fuzzy automaton. The fuzzy automaton is defined in such a way that its transitions model every possible edit operation. Thus, it allows to compute the similarity between an observed string and a pattern string. When this fuzzy automaton is extended by means of a deformed system, the same computation can be performed over strings of fuzzy symbols.

Our method is quite general. It does not assume a limit in the number of errors to be handled. Moreover, the deformed system allows to adjust the values of transition fuzziness and to select different fuzzy operations for computing the transitions. We wish to emphasize the influence that the fuzzy operations used when computing the transitions has on the obtained results. We have used the parametric $t$-norm/$t$-conorm proposed by Hamacher [7]. Variations of the $t$-norm/$t$-conorm parameter lead to very different results.

Due to the difficulty to adjust the values of transition fuzziness once selected the fuzzy operators, we have used a genetic algorithm that obtains good results.

The deformed system has been used as the layer of contextual post processing in a system of text recognition. Section 3 shows that the deformed system gives high ratios of error recovery even when the number of edition errors considered is high.

It has been clearly established that, although usually the string matching methods are based on discrete mathematics and probabilistic techniques, fuzzy techniques can also be useful within this area.

## Acknowledgments

The authors wish to thank to the enterprise Investigación y Programas, S.A. (IPSA) for the contribution to the experimental part and for the financial support through research grant OTRI-2001,4059 (Universidad Pública de Navarra).

## References

1. H. Bunke, H., Csirik, J,: Parametric String Edit Distance and its Application to Pattern Recognition In: IEEE Transactions on Systems, Man and Cybernetics, Vol. 25, No. 1, (1995) 202–206

2. Echanobe, J., González de Mendívil, J.R., Garitagoitia, J.R., Alastruey, C.F.: Deformed systems for contextual postprocessing In: Fuzzy Sets and Systems, Vol. 96, (1998) 335–341
3. Gader, P., Mohamed, M., Chiang, J.H.: Comparison of Crisp and Fuzzy Character Neural Networks in Handwritten Word Recognition In: IEEE Transactions on Fuzzy Systems, Vol. 3, No. 3, (1995) 357–510
4. González de Mendívil, J.R., Garitagoitia, J.R, Astrain, J.J.: Fuzzy automata for imperfect string matching In: Proccedings of ESTYLF 2000, Sevilla, Spain, (2000) 527–532
5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Reading, Massachusetts, USA (1989)
6. Hull, J.J., Srihari, S.N., Choudhari, R.: An integrated Algorithm for Text Recognition: Comparison with a Cascated Algorithm In: IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), Vol. 5, No. 4, (1983) 384–395
7. Klir, G.J., Yuan, B.: Fuzzy Sets and Fuzzy Logic: Theory and Applications. Prentice Hall, New Jersey, USA (1995)
8. Kucera, H., Francis, W.N.: Computational Analysis of Present-Day American English. RI Brown Univ. Press, Providence, USA (1967)
9. Levenshtein, V.I.: Binary codes capable of correcting deletions,insertions, and reversals In: Soviet Physics Docklady, Vol. 10, No. 8, (1966) 707–710
10. Marzal, A., Vidal, E.: Computation of Normalized Edit Distance and Applications In: IEEE PAMI, Vol. 15, No. 9, (1993) 926–932
11. Negoita, C.V., Ralescu, D.A.: Application of Fuzzy Sets to System Analysis. Birkaeuser, Basilea, Switzerland (1975)
12. Oommen, B.J., Kashyap, R.L.: A formal theory for optimal and information theoretic syntactic pattern recognition In: Pattern Recognition, Vol. 31, No. 8, (1998) 1157–1177
13. Rumelhart, D.E., Hinton, G.E., Willians, R.J.: Learning Internal Representations by Error Propagation In: Rumelhart & McClelland Eds (eds): Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations. MIT Press (1986)
14. Sankoff, D., Kruskal, J.B.: Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison. Addison-Wesley Publishing Company, Reading, Massachusetts, USA (1983)
15. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem In: Journal of ACM, Vol. 21, No. 1, (1974) 168–173
16. Wen-Tsong Chen, Gader, P., Hongchi Shi: Lexicon-Driven Handwritten Word Recognition Using Optimal Linear Combinations of Order Statistics In: IEEE PAMI, Vol. 21, No. 1, (1999) 77–82
17. Xu, L., Krzyzak, A., Suen, C.Y.: Methods for Combining Multiple Classifiers and Their Applications to Handwritting Recognition In: IEEE Trans. Sys. Man. and Cyber. (SMC), Vol. 22, No. 3, (1992) 418–435