Deformed Fuzzy Automata for the Text Error Correction Problem

J. Echanobe
Dpto. Matemática e Informática
Univ. Pública de Navarra
Campus de Arrosadía
31006 Pamplona, Spain
javi@we.lc.ehu.es

J. R. Garitagoitia
Dpto. Matemática e Informática
Univ. Pública de Navarra
Campus de Arrosadía
31006 Pamplona, Spain
joserra@si.unavarra.es

J. R. González de Mendívil
Dpto. Automática y Computación
Univ. Pública de Navarra
Campus de Arrosadía
31006 Pamplona, Spain
mendivil@si.unavarra.es

Summary

A fuzzy method for the text error correction problem is introduced. The method is able to handle insert, delete and change errors. Moreover, it uses the measurement level output that an Isolated Character Classifier can provide. The method is based on a Deformed System, in particular, a deformed fuzzy automaton is defined to model the possible errors in the words of the texts. Experimental results show good performance in correcting the three types of errors.

Keywords: Deformed Systems, Fuzzy automata, Contextual postprocessing, Text recognition.

1 Introduction

The automatic detection and correction of errors is an important problem in the recognition of texts. Textual errors are mainly caused during the recognition process, and they are known as edition errors: insert, delete or change errors. In text recognition systems, the error correction is in part provided by a Contextual Postprocessing (CP). Let $\omega = a_1 a_2 \dots a_m$ be an observed word which is obtained from a previous stage of the system; being the characters a_i $(1 \leq i \leq m)$ belong to an alphabet Σ. The objective of the CP is to estimate a word $\hat{\omega}$ in a set of words D (a dictionary) that is the best selection for ω, e.g., it minimizes a certain distance function $d(\hat{\omega}, \omega)$ or maximizes the posteriori probability $P(\hat{\omega} \mid \omega)$. This problem is referred to as one of *text error correction*.

Different methods for handling the text error correction problem have been proposed in the literature: Statistical methods as in [4]; Dictionary based methods [6]; Hybrid methods (a combination of the previous ones) [5]. All of these methods assumed that each character a_i in the observed word ω is a hard decision output by a classifier. However, the classification of a character depends on the proximity values between a group of prototype characters and the input to be classified [12]. Consequently, it is possible to take into account that information and to provide a fuzzy character $\tilde{a}_i$ $(1 \leq i \leq m)$ for each input, that is, $\tilde{a}_i : \Sigma \rightarrow [0, 1]$.

In [2], a fuzzy method for the CP stage was introduced. The method allows to combine the information of a dictionary together with a sequence of observed fuzzy characters, $\tilde{\omega} = \tilde{a}_1 \tilde{a}_2 \dots \tilde{a}_m$. A simple and flexible implementation of the method was achieved by using Deformed Systems [8]. In [2], the method was only applied to deal with change errors produced by an Isolated Character Classifier (ICC), we denoted these errors as ICC errors. The results show that over a 97% of words with change errors was corrected in an experiment using an ICC of 32% of change error rate, a text of 6396 words, and a dictionary of 1700 words. For that experiment the total recognition rate was 99.10% which improves the results given by other methods [5].

In this paper, a fuzzy method for the text error correction problem is introduced. The method is based on a Deformed System, in particular, a fuzzy automaton is defined to model the possible delete, insert or change errors in the words of texts. The deformed fuzzy automaton is implemented in order to work with sequences of fuzzy characters. The experimental results show a good performance for the three types of errors.

The rest of the paper is organized as follows. Section 2 is devoted to introduce the method. Section 3 provides an efficient algorithm for implementing the method. Finally, the experimental results and the conclusion are shown in Section 4.

2 Deformed Fuzzy Automata

The proposed fuzzy method will begin with the definition of the finite deterministic automaton which accepts (recognizes) the set D. More precisely, we provide an automaton for each word in the dictionary. Given a dictionary $D = \{w_1, \dots, w_r\}$, $\omega_i \in \Sigma^+$; we define the automaton $M(\omega_i)$ in the follow-

Input: $M(\omega_i) = (Q, \Sigma, \delta, \{q_0\}, \{q_n\})$, $Q = \{q_0, \ldots, q_n\}$, n is the length of the chain ω_i
$\tilde{\omega}$ observed string, lenght m ($\tilde{x}_k$ k-th symbol of $\tilde{\omega}$)
Output: C has the value $\mu_{\tilde{L}(MD(\omega_i))}(\tilde{\omega}) = \max_{q',q'' \in Q} (\sigma(q') \star \tilde{\mu}^*(q', q'', \tilde{\omega}) \star \eta(q''))$
where $MD(\omega_i) = (Q, \Sigma, \mu, \sigma, \eta, \tilde{\mu})$ is the deformed fuzzy automaton for $M(\omega_i)$

algorithm initiation
  $\forall i : 1..n$:
    $C(q_i) := 0$;
  $C(q_0) := 1$;
  phase_2
endalgorithm
algorithm computation
  $\forall k : 1..m$:
    phase_1(k);
    phase_2
endalgorithm
algorithm decission
  $C := C(q_n)$
endalgorithm
Note 1: $C(q_{-1}) = 0$.

algorithm phase_1(k)
  $\forall i : 0..n$:
    $C'(q_i) := \max(C(q_i) \star \max_{a \in \Sigma}(\mu_{d_a}), C_1(q_{i-1}, q_i))$;
      where $C_1(q_{i-1}, q_i) = C(q_{i-1}) \star \max_{a \in \Sigma}(\mu_{c_{ax}} \star \mu_{\tilde{x}_k}(a))$,[1]
      and x is that $\delta(q_{i-1}, q_i, x) = 1$
  $\forall i : 0..n$: $C(q_i) := C'(q_i)$
endalgorithm
algorithm phase_2
  $\forall i : 1..n$:
    $C(q_i) := \max(C(q_i), C(q_{i-1}) \star \mu_{i_x})$;
      where x is that $\delta(q_{i-1}, q_i, x) = 1$
endalgorithm

Figure 1: Algorithm for the deformed automaton

ing way. If the word ω_i has length n_i, then $M(\omega_i) = (\{q_0^i, q_1^i, \ldots, q_{n_i}^i\}, \Sigma, \delta^i, q_0^i, \{q_{n_i}^i\})$. The states are ordered from 0 to n_i; where q_0^i and $q_{n_i}^i$ are the start and final state respectively. The transition function is $\delta^i(q_{k-1}^i, q_k^i, a) = 1$, being $a \in \Sigma$ the k-th character in the word ω_i. The rest of the transition function values are null.

The proposed method must handle with erroneous observed words. For that purpose, we define for each $M(\omega_i)$ a fuzzy automaton [7, 9] $MF(\omega_i) = (Q^i, \Sigma, \mu^i, \sigma^i, \eta^i)$ which models the possible insert, delete and change errors for the word ω_i. In the previous definition $Q^i = \{q_0^i, q_1^i, \ldots, q_{n_i}^i\}$ is the set of states; $\sigma^i(q_0^i) = 1$ (0 for the remainder states) is the initial fuzzy configuration, $\eta^i(q_{n_i}^i) = 1$ (0 for the remainder states) is the final fuzzy configuration; and the transition function is defined by the following procedure (the symbol ε denotes the empty string in the expressions):

(1) if $\delta^i(q_{k-1}^i, q_k^i, a) = 1$ then
  $\mu^i(q_{k-1}^i, q_k^i, a) = \mu_{c_{aa}} = 1$ (no error)
  $\mu^i(q_{k-1}^i, q_k^i, \varepsilon) = \mu_{i_a}$ (insert operator)
  $\forall x \in \Sigma$, $x \neq a$: $\mu^i(q_{k-1}^i, q_k^i, x) = \mu_{c_{xa}}$ (change op.)
(2) $\forall q^i \in Q^i$, $\forall x \in \Sigma$: $\mu^i(q^i, q^i, x) = \mu_{d_x}$ (delete op.)

One can note that the different error operators are introduced in order to simulate the inverse effect over the observed word, e.g., a fuzzy transition generated via a delete operator is supplied to model the possible insert error in the observed word. The values $\mu_{i_a}, \mu_{c_{xa}}, \mu_{d_x} \in [0, 1]$ are the membership values associated to the fuzzy transitions, and they can be interpreted as a 'cost' for each transition.

It is possible to expand the transition function μ^i to work with strings from Σ^*, that is, $\mu^{i^*} : Q^i \times Q^i \times \Sigma^* \to [0, 1]$ is defined as (i) $\mu^{i^*}(q', q'', \varepsilon) = \mu^i(q', q'', \varepsilon)$; and (ii) $\mu^{i^*}(q', q'', \omega a) = \max_{q \in Q^i}(\mu^{i^*}(q', q, \omega) \star \mu^i(q, q'', a))$, with $q', q'' \in Q^i$, $a \in \Sigma \cup \{\varepsilon\}$, and $\omega \in \Sigma^*$.

In the above expressions, $\star$ represents the composition functions [13]: $\mu'(x) \star \mu''(x) = \mu'(x) \cdot \mu''(x) / [\lambda + (1 - \lambda) \cdot (\mu'(x) + \mu''(x) - \mu'(x) \cdot \mu''(x))]$.

In our discussion, if $\tilde{L}(MF(\omega_i))$ is the fuzzy language which is accepted by the fuzzy automaton $MF(\omega_i)$, then for a string ω the automaton calculates the following membership value $\mu_{\tilde{L}(MF(\omega_i))}(\omega) = \max_{q', q'' \in Q^i}(\sigma^i(q') \star \mu^{i^*}(q', q'', \omega) \star \eta^i(q''))$.

One of the main motivations to use fuzzy characters as inputs to the CP is due to the fact that an ICC (or a multiple classificator) can supply outputs in the measurement level [12], e.g., each character of the alphabet has a proximity value to the character to be classified. In this work, we will study the same combination as in [2, 3] for handling with the complete type of errors. In the following, we will formulate the deformed fuzzy automaton $MD(\omega_i)$ for the fuzzy automaton $MF(\omega_i)$.

Let $\tilde{x} = \{(a, \mu_{\tilde{x}}(a) \mid a \in \Sigma\}$ be a *fuzzy character*. We assume that it is supplied for the ICC being $\mu_{\tilde{x}}(a)$ the normalized value of proximity of the input x to each character of the alphabet. We denotes $\tilde{\Sigma}$ as the set of the possible fuzzy characters over the universe Σ. Therefore, it is necessary to handle strings of observed fuzzy characters, denoted $\tilde{\omega}$. In the following, we illustrate how to transform the fuzzy automata $MF(\omega_i)$ to accept such strings of fuzzy characters. The deformed fuzzy automaton $MD(\omega_i)$ is obtained from $MF(\omega_i) \equiv (Q^i, \Sigma, \mu^i, \sigma^i, \eta^i)$ via modifying the

Table 1: Word Recognition rates and error correction rates for different composition functions and different word error rates. The experiments have been carried out for the three sets of cost values A, B and C

Dictionary I (1720 words)	Algebraic Product			Einstein Product			Minimum		
(%) Word Error	31.2	44.7	71.7	31.2	44.7	71.7	31.2	44.7	71.7
(%) Recognized Words	A) 95.5	93.2	85.6	A) 95.6	93.2	85.7	A) 78.2	68.3	44.2
	B) 95.5	93.2	85.8	B) 95.5	93.3	85.9	B) 77.8	67.7	43.5
	C) 95.5	93.1	85.7	C) 95.6	93.3	85.9	C) 77.7	67.5	43.4
(%) Corrected Words	A) 85.6	84.8	79.9	A) 85.9	84.8	80.1	A) 30.3	29.3	22.2
	B) 85.8	84.8	80.2	B) 85.7	84.9	80.4	B) 28.9	27.9	21.1
	C) 85.6	84.7	80.0	C) 85.9	85.0	80.4	C) 28.8	27.5	20.9
(%) Corrected ICC Errors	A) 94.4	94.1	91.8	A) 94.8	94.3	92.0	A) 67.6	64.3	46.5
	B) 93.2	92.8	90.5	B) 93.4	92.8	90.7	B) 60.4	56.5	40.8
	C) 93.7	93.2	91.1	C) 93.9	93.4	91.4	C) 58.5	54.2	39.0
(%) Corrected Insert Errors	A) 89.4	89.0	85.9	A) 89.9	89.4	86.2	A) 00.1	00.2	00.5
	B) 90.5	89.8	87.0	B) 90.7	90.3	87.6	B) 00.1	00.2	00.5
	C) 88.6	87.8	84.9	C) 89.4	88.5	85.5	C) 00.1	00.1	00.4
(%) Corrected Delete Errors	A) 75.2	74.7	71.3	A) 75.6	74.6	71.5	A) 41.2	37.9	24.7
	B) 77.0	76.2	73.7	B) 76.5	76.2	73.8	B) 48.0	44.8	29.7
	C) 77.5	77.0	74.6	C) 77.6	77.3	74.8	C) 50.2	46.3	31.5
(%) Corrected Change Errors	A) 84.5	84.0	81.1	A) 84.5	83.7	81.0	A) 13.3	12.6	09.1
	B) 83.7	83.2	80.4	B) 83.3	83.1	80.1	B) 10.5	09.8	06.9
	C) 84.1	83.5	80.8	C) 84.0	83.6	80.8	C) 10.1	09.3	06.6

transition function. The new transition function, denoted $\tilde{\mu}^i$, $\tilde{\mu}^i : Q^i \times Q^i \times (\tilde{\Sigma} \cup \{\varepsilon\}) \rightarrow [0,1]$ is defined as follows:

(1) $\tilde{\mu}^i(q', q'', \tilde{a}) = \max_{x \in \Sigma}(\mu^i(q', q'', x) \star \mu_{\tilde{a}}(x))$, with $q', q'' \in Q^i$, and $\tilde{a} \in \tilde{\Sigma}$;
(2) $\tilde{\mu}^i(q', q'', \varepsilon) = \mu^i(q', q'', \varepsilon)$, with $q', q'' \in Q^i$.

It is possible the extension of $\tilde{\mu}^i$ to $\tilde{\mu}^{i^*} : Q^i \times Q^i \times \tilde{\Sigma}^* \rightarrow [0,1]$, in the same way as in the fuzzy automaton.

If $\tilde{L}(MD(\omega_i))$ is the fuzzy language which is accepted by the deformed fuzzy automaton $MD(\omega_i)$, then for an observed fuzzy string $\tilde{\omega}$ the machine calculates the membership value $\mu_{\tilde{L}(MD(\omega_i))}(\tilde{\omega}) = \max_{q', q'' \in Q^i}(\sigma^i(q') \star \tilde{\mu}^{i^*}(q', q'', \tilde{\omega}) \star \eta^i(q''))$.

Therefore, given an observed string of fuzzy characters $\tilde{\omega} \in \tilde{\Sigma}^*$, each $MD(\omega_i)$ calculates $MD(\omega_i, \tilde{\omega}) = \mu_{\tilde{L}(MD(\omega_i))}(\tilde{\omega})$, which can be interpreted as the 'cost' to transform $\tilde{\omega}$ in ω_i. Finally, $\tilde{\omega}$ is classified as a dictionary word $\hat{\omega} \in D$, such that $MD(\hat{\omega}, \tilde{\omega}) \geq MD(\omega_i, \tilde{\omega})$, $\forall \omega_i \in D$.

3 The Algorithm

In practice, it is possible to development an algorithm that simulates the process to be carried on by the deformed fuzzy automaton. We does not explicitly make the $MD(\omega_i)$ automaton for each ω_i in the dictionary; the algorithm operates on the original automata $M(\omega_i)$.

The algorithm for performing the process has to compute the final membership value associated to every one of the strings that can be obtain from the observed fuzzy word $\tilde{\omega}$. These strings are obtained by applying repeatedly the insert, delete and change operations. Finally, we choose as the estimated string, the string with the minimum cost (maximum membership). The execution time needed for this process is very high. Thus, for example, for the change operation it would be necessary to process t^m strings, where t is the size of Σ and m is the length of the observed fuzzy word. However, in order to decrease the execution time, we have developed a dynamic programming algorithm (similar to that proposed by Viterbi [10, 11] and also by Bouloutas [1]). This algorithm computes for every input fuzzy character, the fuzzy transition needed for reaching every state of the automaton. Figure 1 shows this algorithm. Given an automaton $M(\omega_i)$, the complexity of the *phase_1* is $\mathcal{O}(t^2 \cdot n)$, where t is the size of Σ and n is the length of the string ω_i. In the other hand, the *phase_2* has a complexity $\mathcal{O}(n)$. The complete execution of the algorithm *computation* for an observed string of length m is $\mathcal{O}(m \cdot t^2 \cdot n)$.

4 Experimental Results

Basically the experiments consist in processing handprinted texts by the text recognition system and comparing the results before and after the deformed fuzzy automaton stage; thus, we can analyze how efficient is this automaton to reduce the error rate.

First, we have tested the recognition system with a large collection of values for the deformed automaton transitions. From these previous experiments we con-

Table 2: Word Recognition rates and error correction rates for different dictionary sizes

	Dictionary II (3158 words)			Dictionary III (6838 words)		
(%) Word Error	31.2			31.2		
(%) Corrected Words	A) 82.3	B) 82.3	C) 82.4	A) 78.4	B) 78.6	C) 78.6
(%) Corrected ICC Errors	A) 93.4	B) 91.4	C) 92.3	A) 91.2	B) 89.5	C) 90.2
(%) Corrected Insert Errors	A) 87.1	B) 87.8	C) 86.0	A) 82.4	B) 83.6	C) 81.4
(%) Corrected Delete Errors	A) 69.0	B) 71.1	C) 71.8	A) 65.2	B) 67.3	C) 68.5
(%) Corrected Change Errors	A) 81.3	B) 80.6	C) 81.1	A) 76.8	B) 75.9	C) 76.1

clude that these values may verify certain relations among them to obtain good performances. We have chosen three set of values that verify such relations for the experiments: $\forall x, y \in \Sigma$ A) μ_{i_x}= 0.001 $\mu_{c_{xy}}$= 0.0005 μ_{d_x}=0.0001; B) μ_{i_x}= 0.005 $\mu_{c_{xy}}$= 0.001 μ_{d_x}= 0.0005; C) μ_{i_x}= 0.01 $\mu_{c_{xy}}$= 0.001 μ_{d_x}= 0.0001;

Table 1 shows the results obtained in the experiments for different composition functions (Algebraic product $\lambda = 1$, Einstein product $\lambda = 2$, and minimum) and for different edition error rates introduced by both the segmentation process and the ICC (% word error in the table). In these experiments we have used a 1000-words text written by many different authors and a 1720-words dictionary. In the table 1, we can see the rate of recognized words and also the rates of corrected errors for every error type. The results are very similar for the *algebraic product* and for the *Einstein product*. These values are very high; thus, even for a word error of 71.5%, the deformed fuzzy automaton is able to correct about an 80% of the erroneous words. However, the results obtained with the *minimum* function are very poor. Anyway, this result was expected because the minimum is a very drastic operator. In the other hand we can see how different correction rates are obtained depending on the kind of errors. Thus, best results are obtained in the case of ICC errors. This can be explained because the automaton, when it corrects a ICC error, can uses the measurement level that the ICC provides for an input letter. Finally, we can also see how the results are very similar for the three sets of cost values selected.

Table 2 shows the results obtained with the same 1000-words text but with different dictionary sizes: a 3158-words dictionary and a 6838-words dictionary. We can see in the table how the results are now lower than those in table 1. However, they are still high. Thus, with the largest dictionary the automaton corrects more than the three quarter parts of the erroneous words. In conclusion, the experimental results show good performance in correcting the three types of errors.

References

[1] A. Bouloutas, G.W. Hart, M. Schwartz; Two Extensions of the Viterbi Algorithm; *IEEE Tran. on Information Theory*, 37(2) pp. 430-436; 1991.

[2] J. Echanobe, J.R. González de Mendívil, J.R. Garitagoitia, C. Alastruey; Deformed Systems for Contextual Postproccesing; *Fuzzy Sets and Systems* 96, pp. 335-341; 1998.

[3] J. Echanobe. Tesis Doctoral. Universidad Pública de Navarra. 1998.

[4] A. Goshtasby, R.W. Ehrich; Contextual Word Recognition Using Probabilistic Relaxation Labeling; *Pattern Recognition* 21(5), pp. 455-462; 1988.

[5] J.J. Hull, S.N. Srihari, R. Choudhari; An Integrated Algorithm for Text Recognition: Comparison with a Cascaded Algorithm; *IEEE Trans. Pattern Analysis Mach. Intell.* 5(4), pp. 384-395; 1983.

[6] R.L. Kashyap, B. J. Oommen; Spelling Correction Using Probabilistic Methods; *Pattern Recognition Letters* 2(4), pp. 147-154; 1984.

[7] M. Mizumoto, J. Toyoda, K. Tanaka; Fuzzy Languages; *Systems Computers Control*; pp. 36-43; 1970.

[8] C.V. Negoita, D.A. Ralescu; *Application of Fuzzy Sets to System Analysis*; Birkaeuser, Basilea; 1975.

[9] M.G. Thomason; Finite fuzzy automata, regular fuzzy languages and pattern recognition; *Pattern Recognition* 5; pp. 383-390; 1973.

[10] A.J. Viterbi; Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm; *IEEE Tran. on Information Theory* 13(2); pp. 260-269; 1967.

[11] A.J. Viterbi; Convolutional Codes and Their Performance in Communication Systems; *IEEE Tran. on Communications Technology* 19(5); pp. 751-772; 1971.

[12] L. Xu, A. Krzyzak, C.Y. Suen; Methods for Combining Multiple Classifiers and Their Applications to Handwritting Recognition; *IEEE Trans. Sys. Man and Cyber.* 22(3); pp. 418-435; 1992.

[13] H.J. Zimmermann; *Fuzzy Set Theory and its Applications*; Kluwer Academic Pub., Boston, MA; 1990.