Conversion of Finite Automata to Fuzzy Automata for String Comparison

Dr. V. Ramaswamy
Principal
Bapuji Institute of Engineering & Technology
Davangere, Karnataka, India

H. A. Girijamma
Associate Professor, CSE
RNS Institute of Technology
Channasandra, Bangalore, Karnataka, India

ABSTRACT
In this paper, a method has been presented to convert finite automaton to fuzzy automaton as fuzzy automaton is better than finite automaton for strings comparison when individual levels of similarity for particular pairs of symbols or sequences of symbols are defined. A finite automaton is useful in determining whether a given string is accepted or not whereas fuzzy automaton determines the extent to which the string is accepted.

1. INTRODUCTION
Approximate string matching is the technique of finding approximate matches to a pattern in a given text. Approximate string matching is very fundamental to text processing because a spell check program must be able to identify the closest match for a given text string which is not found in the dictionary. Approximate string matching finds applications in the areas like Computational Biology, Signal Processing, Text Retrieval or Spell Checker, Correction systems for optical character recognition and Software to assist natural language translation.

Edit distance
The edit distance between two strings of characters is nothing but the number of operations required to transform one string into another. Edit distance can be defined in various ways. Following two edit distances are the most commonly used ones.

- Hamming distance
- Levenshtein distance

Hamming distance
The Hamming distance between two strings of equal length is nothing but the number of positions in which the corresponding symbols differ. i.e., it measures the minimum number of substitutions (also called number of errors) required to change one string into the other.

Example:
The Hamming distance between "**road**" and "**ride**" is 3, between **11011** and **10011** is 1 and between **438765** and **428664** is 3.

Levenshtein distance
The levenshtein distance between two strings is defined as the minimum number of edits required to transform one string into the other with the allowable edit operations being insertion, deletion and substitution of a single character.

For example, the levenshtein distance between "abcd" and "afcde" is 2, because the following two edits change one to the other. Also there is no way this can be done with fewer than two edits:

1. a**b**cd → a**f**cd (substitution of 'f' for 'b')
2. afcd → afcd**e** (insertion of 'e')

The distance d (x, y) between two strings x and y is the minimal cost of a sequence of operations that transform x into y. The cost of each of the operations allowed by levenshtein distance namely insertion, deletion and replacement or substitution is assumed to be 1. Note that Hamming distance allows only substitution whose cost is 1 and is valid only for two strings of equal length whereas Levenshtein distance holds for two strings of different lengths. Hamming distance is denoted by R whereas levenshtein distance is denoted by DIR. In other words,

R (P, P_0) is defined as the minimum number of symbol replacement operations required for the conversion of string P into string P_0 or vice versa. DIR (P, P_0) is defined as the minimum number of operations of symbol deletion (D), insertion (I) or replacement (R) required for the conversion of string P into string P_0 or vice versa.

In this paper, a technique has been presented which can be used to search or match strings in special cases when some pairs of symbols are more similar to each other when compared to other pairs. This kind of similarity cannot be handled by usual search algorithms. A spell checker based on a dictionary of correct words and abbreviations is a common way by which basic checking of a given text document can be carried out by searching each of its words in our dictionary. A word which is not found in the dictionary is highlighted and a correction is suggested. The suggested words are those in the dictionary which are closest to the unknown word in the sense that those words can be obtained from the unknown word by means of addition, deletion and replacement of symbols.

It is not too difficult to implement this common model. But it does not take into consideration the fact that some pairs of symbols are more similar than others. This is very specific to the language. For example in Latin alphabet `a' and `e' or `i' and `y' are somewhat related and hence more similar than for example `w' and `b'. In many European languages we can find some letters of extended Latin alphabet whose similarity solely depends on the nature of the language. The primary concern here is that it can't be simply implemented using standard string search models.

A fuzzy automaton allows to define individual levels of similarity for particular pairs of symbols or sequences of symbols and hence can be used as a base for providing a better string search operation. So conversion of the given finite automaton to fuzzy automaton using the similarity function is presented in the following section.

2. CONSTRUCTION OF FUZZY AUTOMATON USING FINITE AUTOMATON AND SIMILARITY FUNCTION

Consider a finite automaton M = (Q, Σ, q_0, δ, F). g_s is a similarity function which defines similarity level between each pair of input symbols. In other words,
g_s: Σ x Σ → [0, 1] defined as follows

$$g_s(a_i, a_j) = \begin{cases} 1 \text{ if } a_i \text{ and } a_j \text{ are fully equal} \\ 0 \text{ if } a_i \text{ and } a_j \text{ are completely different} \\ \text{a value between 0 and 1 depending on the similarity between } a_i \text{ and } a_j. \end{cases}$$

Assume g_s is a symmetric function so that
$g_s(a_i, a_j) = g_s(a_j, a_i)$. With M, associate a fuzzy automaton
M' = (Q', f, I, F') as given below. Q' is same as Q, I (q_0) = 1 and I (q) = 0 ∀ q ∈ Q, q ≠ q_0, F' (q) = 1 if q ∈ F and 0 otherwise. The fuzzy transition function f is computed using the formula

$$f(q_i, a, q_j) = \bigvee_{x\in\Sigma} (g_s(a, x) \wedge \delta_x(q_i, q_j)) \;\ldots\ldots(*)$$

∀ $q_i, q_j \in Q$ and $a \in \Sigma$.
Here $\delta_a(q_i, q_j) = 1$ means $\delta(q_i, a) = q_j$. Similarly $\delta_b(q_i, q_j) = 1$ means $\delta(q_i, b) = q_j$. Here i represent row number whereas j represents column number

3. EXAMPLES

Illustration of the above construction by examples.

Example 1

Consider the finite automaton M = (Q, Σ, q_1, δ, F) where
Q = {q_1, q_2, q_3, q_4, q_5), Σ = {a, b, c}, Start state = q_1 ,
F = {q_3}and δ is given by

$$\delta_a = \begin{pmatrix} 1&1&0&0&0 \\ 0&0&0&0&1 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{pmatrix} \quad \delta_b = \begin{pmatrix} 1&0&0&1&0 \\ 0&0&1&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&1 \\ 0&0&0&0&0 \end{pmatrix} \quad \delta_c = \begin{pmatrix} 1&0&0&1&0 \\ 0&0&0&0&1 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{pmatrix}$$

In the above δ_a matrix, $\delta_a(q_1, q_1) = \delta_a(q_1, q_2) = \delta_a(q_2, q_5) = 1$ i.e. $\delta(q_1, a) = q_1$, $\delta(q_1, a) = q_2$ and $\delta(q_2, a) = q_5$ respectively. Similarly δ_b and δ_c are defined. Let w_0 = {"ab"}, w_1 = {"bb"} and w_2 = {"cb"}. First compare w_1 and w_2 to w_0 using Hamming distance R. R (w_0, w_1) = 1 and R (w_0, w_2) = 1. In this case, w_1 and w_2 have got the same level of similarity to w_0. Next compare w_1 and w_2 to w_0 using levenshtein distance DIR. DIR (w_0, w_1) = 1 and DIR (w_0, w_2) = 1. In this case also w_1 and w_2 have got the same level of similarity to w_0.

Now construct the corresponding fuzzy automaton

M' = (Q', f, I, F') where Q' = {q_1, q_2, q_3, q_4, q_5), I (q_1) = 1 and I (q_2) = I (q_3) = I (q_4) = I (q_5) = 0, F' (q_3) = 1 and F' (q_1) = F' (q_2) = F' (q_4) = F' (q_5) = 0. Suppose assume that the symbols 'a' and 'c' are bit similar whereas the others are pairwise different. Assume g_s to be the matrix given below.

$$g_s = \begin{pmatrix} 1.0 & 0.0 & 0.3 \\ 0.0 & 1.0 & 0.0 \\ 0.3 & 0.0 & 1.0 \end{pmatrix}$$

Using (*), obtain the values for f_a, f_b and f_c matrices as given below.

$f(q_1, a, q_1) = f_a(q_1, q_1)$
$f_a(q_1, q_1) = \vee [(g_s(a, a) \wedge \delta_a(q_1, q_1)) , (g_s(a, b) \wedge \delta_b(q_1, q_1)), (g_s(a, c) \wedge \delta_c(q_1, q_1))]$
$= \vee [(1 \wedge 1), (0 \wedge 1), (0.3 \wedge 1))$
$= \vee [1, 0, 0.3]$
$= 1$
$f_a(q_1, q_2) = \vee [(g_s(a, a) \wedge \delta_a(q_1, q_2)) , (g_s(a, b) \wedge \delta_b(q_1, q_2)), (g_s(a, c) \wedge \delta_c(q_1, q_2))]$
$= \vee [(1 \wedge 1), (0 \wedge 0), (0.3 \wedge 0))$
$= \vee [1, 0, 0]$
$= 1$
$f_a(q_1, q_3) = \vee [(g_s(a, a) \wedge \delta_a(q_1, q_3)) , (g_s(a, b) \wedge \delta_b(q_1, q_3)), (g_s(a, c) \wedge \delta_c(q_1, q_3))]$
$= \vee [(1 \wedge 0), (0 \wedge 0), (0.3 \wedge 0))$
$= \vee [0, 0, 0]$
$= 0$
$f_a(q_1, q_4) = \vee [(g_s(a, a) \wedge \delta_a(q_1, q_4)) , (g_s(a, b) \wedge \delta_b(q_1, q_4)), (g_s(a, c) \wedge \delta_c(q_1, q_4))]$
$= \vee [(1 \wedge 0), (0 \wedge 1), (0.3 \wedge 1))$
$= \vee [0, 0, 0.3]$
$= 0.3$
$f_a(q_1, q_5) = \vee [(g_s(a, a) \wedge \delta_a(q_1, q_5)) , (g_s(a, b) \wedge \delta_b(q_1, q_5)), (g_s(a, c) \wedge \delta_c(q_1, q_5))]$
$= \vee [(1 \wedge 0), (0 \wedge 0), (0.3 \wedge 0))$
$= \vee [0, 0, 0]$
$= 0$

Similarly all other values of f_a, f_b and f_c are calculated.

$$f_a = \begin{pmatrix} 1.0 & 1.0 & 0.0 & 0.3 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

$$f_b = \begin{pmatrix} 1.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

$$f_c = \begin{pmatrix} 1.0 & 0.3 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

Now $L(M')(w_0) = I \text{ o } f_{ab}^* \text{ o } F$
$= \vee [(f_{ab}^* \text{ o } F)(q_1) \wedge I(q_1)]$
$= (f_{ab}^* \text{ o } F)(q_1) \wedge I(q_1)$
$= (f_{ab}^* \text{ o } F)(q_1)$
$= \vee [f_{ab}^*(q_1, q_3) \wedge F(q_3)]$
$= (1 \wedge 1)$
$= 1$

$$
\begin{aligned}
L(M')(w_1) &= I \text{ o } f_{bb}^* \text{ o } F \\
&= \vee [(f_{bb}^* \text{ o } F)(q_1) \wedge I(q_1)] \\
&= (f_{bb}^* \text{ o } F)(q_1) \wedge I(q_1) \\
&= (f_{bb}^* \text{ o } F)(q_1) \\
&= \vee [f_{bb}^*(q_1, q_3) \wedge F(q_3)] \\
&= (0 \wedge 1) \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
L(M')(w_2) &= I \text{ o } f_{cb}^* \text{ o } F \\
&= \vee [(f_{cb}^* \text{ o } F)(q_1) \wedge I(q_1)] \\
&= (f_{cb}^* \text{ o } F)(q_1) \wedge I(q_1) \\
&= (f_{cb}^* \text{ o } F)(q_1) \\
&= \vee [f_{cb}^*(q_1, q_3) \wedge F(q_3)] \\
&= (0.3 \wedge 1) \\
&= 0.3
\end{aligned}
$$

$L(M')(w_0) = 1$, $L(M')(w_1) = 0$, $L(M')(w_2) = 0.3$.
First compare w_0 and w_1, the difference of $L(M')(w_0)$ and $L(M')(w_1)$ is 1. Similarly, compare w_0 and w_2, the difference of $L(M')(w_0)$ and $L(M')(w_2)$ is 0.7. So comparing these values that w_0 is more similar to w_2 than to w_1.

Example 2

Consider another example. Consider the finite automaton $M = (Q, \Sigma, q_1, \delta, F)$ where $Q = \{q_1, q_2, q_3, q_4, q_5\}$, $\Sigma = \{a, b, c, d\}$, Start state = q_1 , $F = \{q_4, q_5\}$and δ is given by

$$
\delta_a = \begin{pmatrix} 0&1&0&0&0 \\ 0&1&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{pmatrix}
\quad
\delta_b = \begin{pmatrix} 0&1&0&0&0 \\ 0&0&1&0&0 \\ 0&0&1&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{pmatrix}
\quad
\delta_c = \begin{pmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{pmatrix}
\quad
\delta_d = \begin{pmatrix} 1&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&1 \\ 0&0&0&0&0 \end{pmatrix}
$$

Let w_0 = "abbcd", w_1 = "dabbc", w_2 = "ddbdd"
$R(w_0, w_1) = 4$, $R(w_0, w_2) = 4$. So decision cannot be made whether w_0 is closer to w_1 or w_2. Hence construct the fuzzy automaton using the above given finite automaton and the following given fuzzy transition matrix. g_s matrix given below
.

$$
g_s = \begin{pmatrix} 1.0 & 0.2 & 0.3 & 0.1 \\ 0.2 & 1.0 & 0.2 & 0.1 \\ 0.3 & 0.2 & 1.0 & 0.2 \\ 0.1 & 0.1 & 0.2 & 1.0 \end{pmatrix}
$$

Fuzzy automaton $M' = (Q', f, I, F')$ where
$Q' = \{q_1, q_2, q_3, q_4, q_5\}$, $I(q_1) = 1$ and $I(q_2) = I(q_3) = I(q_4) = I(q_5) = 0$, $F'(q_4) = F'(q_5) = 1$ and $F'(q_1) = F'(q_2) = F'(q_3) = 0$.
Using (*), we obtain the values for f_a, f_b, f_c and f_d matrices as given below.

$$
f_a = \begin{pmatrix} 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}
$$

$$
f_b = \begin{pmatrix} 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}
$$

$$
f_c = \begin{pmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}
$$

$$
f_d = \begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}
$$

$$
\begin{aligned}
L(M')(w_0) &= I \text{ o } f_s^* \text{ o } F \qquad (s = w_0) \\
&= \vee [(f_s^* \text{ o } F)(q_1) \wedge I(q_1)] \\
&= (f_s^* \text{ o } F)(q_1) \wedge I(q_1) \\
&= (f_s^* \text{ o } F)(q_1) \\
&= [f_s^*(q_1, q_4) \wedge F(q_4)] \vee [f_s^*(q_1, q_5) \wedge F(q_5)] \\
&= (0.1 \wedge 1) \vee (1 \wedge 1) \\
&= 0.1 \vee 1 \\
&= 1
\end{aligned}
$$

$$
\begin{aligned}
L(M')(w_1) &= I \text{ o } f_s^* \text{ o } F \qquad (s = w_1) \\
&= \vee [(f_s^* \text{ o } F)(q_1) \wedge I(q_1)] \\
&= (f_s^* \text{ o } F)(q_1) \wedge I(q_1) \\
&= (f_s^* \text{ o } F)(q_1) \\
&= [f_s^*(q_1, q_4) \wedge F(q_4)] \vee [f_s^*(q_1, q_5) \wedge F(q_5)] \\
&= (1 \wedge 1) \vee (0.2 \wedge 1) \\
&= 1 \vee 0.2 \\
&= 1
\end{aligned}
$$

$$
\begin{aligned}
L(M')(w_2) &= I \text{ o } f_s^* \text{ o } F \qquad (s = w_2) \\
&= \vee [(f_s^* \text{ o } F)(q_1) \wedge I(q_1)] \\
&= (f_s^* \text{ o } F)(q_1) \wedge I(q_1) \\
&= (f_s^* \text{ o } F)(q_1) \\
&= [f_s^*(q_1, q_4) \wedge F(q_4)] \vee [f_s^*(q_1, q_5) \wedge F(q_5)] \\
&= (0.1 \wedge 1) \vee (0.1 \wedge 1) \\
&= 0.1 \vee 0.1 \\
&= 0.1
\end{aligned}
$$

As $L(M')(w_0) = L(M')(w_1) = 1$ and $L(M')(w_2) = 0.1$.
Compare w_0 and w_1, the difference of $L(M')(w_0)$ and $L(M')(w_1)$ is 0. Similarly, compare w_0 and w_2, the difference of $L(M')(w_0)$ and $L(M')(w_2)$ is 0.9. So comparing these values that w_0 is more similar to w_1 than to w_2.

4. CONCLUSION

A fuzzy automaton allows to define individual levels of similarity for particular pairs of symbols or sequences of symbols and hence can be used as a base for providing a better string search operation. Fuzzy automata are more useful in performing comparison operations unlike finite automata which cannot decide how close two given strings are. Finite automata can help in determining whether a given string is accepted or not whereas fuzzy automata can tell us the extent to which the string is accepted. Hence fuzzy automaton is very useful in comparison of strings as shown by above examples.

5. REFERENCES

[1] John N. Mordeson, Davender S. Malik, Fuzzy Automata and Languages: Theory and Applications, 2002-03-19.

[2] Jiri Mockor, Fuzzy and Non deterministic Automata, Research Report No. 8, Institute for Research and Applications of Fuzzy Modeling, University of Ostrava, Chech Republic, 1999.

[3] Dr. V. Ramaswamy, Girijamma.H. A., Characterization of Fuzzy Regular Languages International Journal of Computer Science and Network Security, VOL 8, No. 12, December 2008.

[4] Dr. V. Ramaswamy, Girijamma.H. A., An extension of Myhill Nerode theorem for fuzzy automata, Advances in Fuzzy Mathematics, Research India Publications, ISSN 0973-533X Volume 4, Number 1 (2009).

[5] John E Hopcroft & Jefrey D. Ullman, Introduction to Automata Theory, Languages and Computation. Narosa publishing house, 1987.

[6] George J. Klir, Bo Yuan, Fuzzy sets and fuzzy logic Theory and Applications, Prentice – Hall of India Pvt. Ltd, 1997.