A Neural Network String Matcher

Conference Paper · August 2007

DOI: 10.1007/978-3-540-74272-2_97 · Source: DBLP

Abdolreza Mirzaei
Isfahan University of Technology
54 PUBLICATIONS **361** CITATIONS
SEE PROFILE

Reza Safabakhsh
Amirkabir University of Technology
115 PUBLICATIONS **1,431** CITATIONS
SEE PROFILE

A Neural Network String Matcher

Abdolreza Mirzaei, Hamidreza Zaboli, and Reza Safabakhsh

Department of Computer Engineering, Amirkabir University,
Tehran 15914, Iran
{amirzaei, zaboli, safabakhsh}@aut.ac.ir

Abstract. The aim of this work is to code the string matching problem as an optimization task and carrying out this optimization problem by means of a Hopfield neural network. The proposed method uses TCNN, a Hopfield neural network with decaying self-feedback, to find the best-matching (i.e., the lowest global distance) path between an input and a template. The proposed method is more than ‘exact’ string matching. For example wild character matches as well as character that never match may be used in either string. As well it can compute edit distance between the two strings. It shows a very good performance in various string matching tasks.

Keywords: String matching, Parallel, Chaotic Neural Network, TCNN, and Optimization using Hopfield NN.

1 Introduction

The general string-matching problem is to find shift s for which the pattern X appears in Text. The most straightforward approach in matching string is to test each possible shift s in turn, which is hardly optimal. In string-matching-with-errors problem the goal is to find the location in text where the pattern X is close to the substring or factor of text. This measure of closeness is chosen to be an edit distance. The edit distance is the minimum number of fundamental operation needed to transform the test string into prototype string. Thus the string-matching-with-error problem finds use in the same type of problem as basic string matching. The only difference being that there is a certain tolerance for match [1].

The string matching problems just outlined are conceptually very simple. The challenge arises when the problems are large. Survey and comparison of well known string matching algorithms can be found in [2], [3], [4]. Although several linear time algorithms have been developed in the last two decades, they all need a preprocessing step [4], [5], [6], [7]. Automata-based methods also need a preprocessing step to create automaton for each input pattern [4], [5], [8]. Among the methods proposed for string-matching-with-errors we can mention dynamic programming methods [9], generalized Boyer-Moore algorithm [10], and automata-based method [5], which all these algorithms require extra time and space for preprocessings. Since the VLSI technology has been developed rapidly, hardware approaches for string matching have also been proposed [11], [12], [13], [14]. Since software algorithms which use preprocessings with look up table methods can not be applied to design special purpose hardware string matcher [15], we need algorithms suitable for hardware implementation.

It is known that neural networks are capable of yielding high-quality solutions to large-scale problems [16]. This paper uses chaotic Hopfield neural network for string matching. The advantage of this method over traditional methods includes massive parallelism and convenient hardware implementation. Another advantage is that the procedure of Hopfield network is more general for application and a range of string matching problems can be solved by it.

The remainder of this paper is organized as follows. Section 2 reviews the theoretical basis of edit distance algorithms. Section 3 introduces the Hopfield neural network and its application in optimization. Section 4 gives a description of proposed method and Section 5 discusses the performance of the developed technique. Section 6 derives the main conclusions from this work.

2 Problem Description

We start with the calculation of edit distance between two strings. In section 5 we will see that the proposed algorithm can be used to solve other problems too. Edit distance between x and y describes how many fundamental operations are required to transform x into y . Theses fundamental operations are:

- *Substitutions:* A character in x is replaced by the corresponding character in y .
- *Insertion:* A character in y is inserted into x , thereby increasing the length of x by one character.
- *Deletion:* A character in x is deleted, thereby decreasing the length of x by one character.

The calculation of edit distance between x and y can be visualized by aligning the character of x along the X-axis and the character of y along the Y-axis. Now calculation of edit distance is equivalent to finding the best-matching (i.e. the lowest global distance) path between x and y . Let C be a $m \times n$ matrix of integers associated with a cost or distance and let $\delta(.,.)$ denotes a generalization of Kronecker delta function, having value 1 if the two arguments (characters) match and 0 otherwise. The variables m and n are equal to the length of x and y denoted by $|x|$ and $|y|$. The basic equations for calculation of edit distance are

$$C[0,0] = 0 \tag{1}$$

$$C[i,0] = i \ \text{ for i} = 1 \text{ to m} \tag{2}$$

$$C[o,j] = j \ \text{ for j} = 1 \text{ to n} \tag{3}$$

$$C[i,j] = \min[\underbrace{C[i-1,j]+1}_{insertion}, \underbrace{C[i,j-1]+1}_{deletion}, \underbrace{C[i-1,j-1]+1-\delta(x[i],y[j])}_{nochange/exchange}] \tag{4}$$

Equation 4 is used to find the minimum cost in each entry of C column by column. As it can be seen from this equation the only valid paths to a given point must pass through a point from the left and/or below it. The final distance $C[m,n]$ gives us the overall matching distance for the x and y patterns.

3 The Transiently Chaotic Neural Network

A Hopfield neural network consists of a set of neurons, where the output of each neuron is connected to the input of all other neurons except itself. The weights on these connections are fixed and are constrained to be symmetrical. The network is characterized by an energy function, which decreases with the changes in neural states with time. Once started, the network relaxes towards a stable state corresponding to a local minimum of its energy function. The work of Hopfield and Tank [17] showed that neural networks can be applied to solving combinatorial optimization problems. They suggested that a nearoptimal solution of traveling salesman problem (TSP) can be obtained by finding a local minimum of an appropriate energy function, which is implemented by a neural network. Typically, the network energy function is made equivalent to the objective function, which is to be minimized, while each of the constraints of the optimization problem is included in the energy function as penalty terms. Clearly, a constrained minimum of the optimization problem will also optimize the energy function.

Unfortunately, a minimum of the energy function does not necessarily correspond to a minimum of the objective function due to the fact that there are likely to be several terms in the energy function, which contribute to many local minima. Furthermore, even if the network does manage to converge to a feasible solution, its quality is likely to be poor compared to other techniques, since the Hopfield network is a descent technique and converges to the first local minimum it encounters. We use the transiently chaotic neural network (TCNN) proposed by Chen and Aihara[18]. TCNN, a hopfield neural network with decaying self-feedback, has been developed as a new approach to extend the problem solving ability of the standard hopfield neural network.

The difference equations describing the dynamics of the TCNN are as follows

$$x_{ij}(t) = f(y_{ij}(t)) = \frac{1}{1+e^{-y_{ij}(t)/\varepsilon}} \tag{5}$$

$$y_{ij}(t+1) = ky_{ij}(t) + \alpha\left(\sum_k \sum_l w_{ijkl}x_{kl}(t) + I_{ij}\right) - z_{ij}(t)(x_{ij}(t) - I_0) \tag{6}$$

$$z_{ij}(t+1) = (1-\beta)z_{ij}(t) \tag{7}$$

where $x_{ij}(t)$ is the output of neuron ij, $y_{ij}(t)$ is the internal state of neuron ij, w_{ijkl} is the connection weight from neuron ij to neuron kl ($w_{ijkl} = w_{klij}$), I_{ij} is the input bias of neuron ij, α is a positive scaling parameter for neuronal input, k is the damping factor of the nerve membrane ($0 \le k \le 1$), $z(t)$ is the self-feedback connection weight (refractory strength), β is the damping factor of $z(t)$ ($0 \le \beta \le 1$), I_0 is a positive parameter and ε is the steepness parameter of the output function($\varepsilon > 0$).

4 String Matching Using Hopfield Neural Network

To solve the string matching problem, the Hopfield network is organized as a $n \times m$ matrix of neurons, where n and m are the total number of character in the pattern and text. A neuron n_{Xi} explores the hypothesis that $character_X$ in pattern word corresponds to $character_i$ in text. We will consider the constraints existing on the matching process (or that we can impose on that process) and use those constraints to come up with an efficient algorithm. The constraints we shall impose are straightforward and not very restrictive:

- Every character in the pattern and the text must be used in a matching path.
- Monotonic condition: the path will not turn back on itself, both the i and j indexes either stay the same or increase, they never decrease.
- Continuity condition: The path advances one step at a time. Both i and j can only increase by 1 on each step along the path.
- Local distance scores are combined by adding to give a global distance.

We said that every character in the pattern and the text must be used in a matching path. This means that if we take a point (i, j) in the matrix (where i indexes the pattern character, j the text character), then previous point must be $(i-1, j-1)$, $(i-1, j)$ or $(i-1, j-1)$.

According to the above mentioned constraints, we introduce the following energy function:

$$\begin{aligned} E = {} & \frac{A}{2}\sum_{i=1}^{n}\left(1-\sum\nolimits_{j=1}^{m} P_{ij}\right)^2 \\ & + \frac{B}{2}\sum_{j=1}^{m}\left(1-\sum\nolimits_{i=1}^{n} P_{ij}\right)^2 \\ & + \frac{C}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k>i}^{n}\sum_{l<j}^{m} P_{ij}.P_{kl} \\ & + \frac{D}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k<i}^{n}\sum_{l>j}^{m} P_{ij}.P_{kl} \\ & + \frac{E}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} P_{ij}.d_{ij} \end{aligned} \qquad (8)$$

where A, B, C, D and E are positive constants. P_{ij} is the output of neuron n_{ij} that takes on values from 0 to 1. The first two terms basically enforce the uniqueness constraint. It can be seen that if more than one neuron is excited in one row or column of the network, the energy of the system tends to increase. This constraint is used (with small coefficient A and B) in order to make the solution more monotonic (if one character is matched with two character, one of them must be deleted or inserted to make two string equal). The minimum energy is obtained only if one of the neurons in one column or row is excited. The third and forth terms are the inhibiting terms to implement the fact that the matching paths cannot go backwards in time. For any on neuron, they inhibit its upper left and lower right as shown in Figure 1.

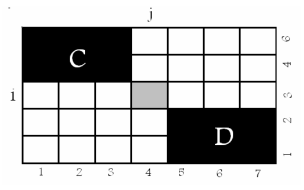

Fig. 1. Inhibiting areas due to monotonicity constraint

The fifth term is the main term, which is used to enforce the similarity constraint. If the two characters are equal, the less the cost of the energy function will be, and the two characters are a correct matching pair with a high probability. d_{ij} is given by

$$d_{ij} = 1 - \delta(x[i], y[j]) \tag{9}$$

The objective now is to compute the network parameters, which include the external inputs of the network and the network connection weights. Comparing equation (8) with the energy function of a Hopfield neural network, given in equation (10), we can compute the properties.

$$H_H = -\frac{1}{2}\sum_i\sum_j\sum_k\sum_l W_{ij,kl}P_{ij}P_{kl} - \sum_i\sum_j I_{ij}P_{ij} \tag{10}$$

where each neuron has the output level of P_{ij} (bounded by zero and one) and the bias current (or negative threshold) I_{ij}. The weights, which determine the strength of the connections from ij to kl, are given by $W_{ij,kl}$.

5 Experimental Results

To evaluate the proposed method, 2 types of experiment are performed. In the experiments the TCNN parameters used are: $k = 0.9$; $I_0 = 0.65$; $z(1) = 0.08$; $\beta = 0.003$; $\alpha = 0.015$ which are the same as the ones used by Chen and Aihara[18]. In our experiments every neuron n_{ij} is initialized to a value inversely proportional to d_{ij}. The criterion $\sum_{i,j=1}^{N}\left|p_{ij}(t+1) - P_{ij}(t)\right| < 5\times10^{-5}$ is used to see whether the network has converged or not. At the end of each experiment, every neuron that has an output greater than a predefined threshold (0.5) is considered as fired. A post-processing step is performed for rows and columns with no fired neuron, in which the greatest activation in that row or column will be replaced by 1(the neuron is considered as fired).

The first experiment is to illustrate the use of TCNN in exact string matching. In this case the number of insert operation (right move) before first character of the text shows a valid shift (i.e. variable s). In this experiment some random character are added to the pattern “book” to create the text with size of 10, 15, 20, 40, 60, 80, 100, 120 characters. Then the ability of network to find a valid shift (for which the pattern "book" appears in the text) is evaluated. Table 1 summarizes the results of 100 different runs in terms of rate of correct estimation of valid shift and the number of iterations for convergence. These results are shown in Figure 2. We note that for a constant value of β, with the increase in the text size the number of iteration for convergence asymptotically become constant and text size has not significant effect on it. On the other hand the accuracy of matching degrades with the increase of text size. This is due to the fact that using the same number of iteration for larger matching problems the network has not enough time to reach the optimal solution. So the value of β must be selected such that for larger problem the number of iteration for convergence increases. To assess the effect of β in network convergence and the quality of obtained paths, we analyzed the matching of a text with size 40 and the pattern "book" with β=0.001, 0.003, 0.005, 0.01, 0.015, 0.02 for 100 runs. The results of this experiment is shown in the Table 2 and Figure 3. In TCNN, the variable $z(t)$ corresponds to the temperature in the stochastic annealing process and β determines the cooling speed. By increasing β the chaotic phase of the network finishes earlier and the network converges with fewer iterations (it can be seen in figure 3(b)). However the searching capabilities of network and therefore the quality of solutions decrease in this case (see figure 3 (a)). So, in order to achieve acceptable results for matching large text, β must be selected small enough.

The second experiment deals with the calculation of the edit distance between two strings. This experiment shows the capability of the approach to approximate

Table 1. The results of using TCNN to find a valid shift with different text size

Length of reference string		10	15	20	40	60	80	100	120
Rate of correct estimation of valid shift	Mean	100	100	100	97.8	96.6	87	79.8	76.2
	Std	0	0	0	4.91	4.27	7.42	11.01	12.19
Number of iterations for convergence	Mean	123.24	126.56	127.96	130.36	130.94	133.18	134	134.62
	Std	2.003	1.502	3.002	2.669	1.595	1.354	0.869	0.918

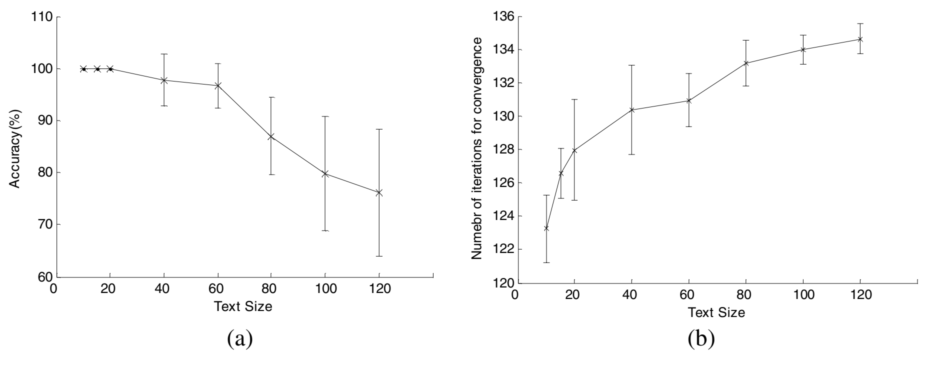

Fig. 2. The results of using TCNN to find a valid shift with different text size. (a)Rate of correct estimation of valid shift (b)Number of iterations for convergence.

Table 2. The results of Matching of a text with size 40 and the pattern "book" with different value of β

β		0.001	0.003	0.005	0.01	0.015	0.02
Rate of correct estimation	Mean	98.2	97.8	96.9	96.7	95.6	93.00
of valid shift	Std	3.8239	4.91	4.99	5.31	5.68	6.15
Number of iterations for	Mean	229.1900	130.36	84.2300	49.4900	35.8400	28.5300
convergence	Std	29.5626	2.669	1.5535	0.4886	0.2716	0.2359

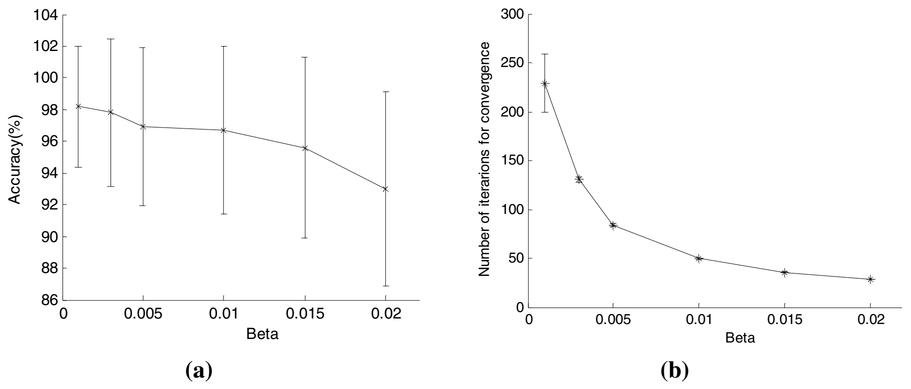

Fig. 3. The results of using TCNN to find a valid shift with different vale for β parameter. (a)Rate of correct estimation of valid shift (b) Number of iterations for convergence(b).

matching (i.e. matching with error). To compute the edit distance, we count the extra ON neurons in rows and columns that have more than one neuron ON and add the results to the matching cost. Matching cost for two equal characters is zero. In order to provide a quantitative evaluation of the proposed method in calculation of the edit distance between two strings, the text "We are all pencils in the hand of God"[*] is corrupted by randomly substituting, deleting or adding a character from/to it, and then the modified text is matched against the original text. The result is considered as optimal if the edit distance computed from the network is the same as the one computed from the well known dynamic programming algorithm and also if the derived path is a valid path. The table 3 shows the ratio of finding the correct edit distance in 100 runs. This table indicates that if the distortion is moderate the network will find the optimal edit distance with a high probability.

Table 3. The result of finding the edit distance between two strings, a fraction of a sentence is corrupted and then it is matched against the original one

Distortion	5%	10%	20%	30%	40%
The ratio of finding the correct edit distance (%)	100	97	91	79	67

[*] Mother Teresa.

6 Conclusion

The main concept of this paper is a demonstration that string matching problem can be formulated as an optimization task. The advantage of this method which uses Hopfield network to solve the resulted optimization problem over traditional methods includes massive parallelism and convenient hardware implementation. Another advantage is that the procedure of Hopfield network is more general for application and a range of string matching problem can be solved by it.

References

1. Duda, R.O, Hart, P.E., Stork, D.G.: Pattern classification, 2nd edn. Wiley-Interscience publication, Chichester (2000)
2. Aho, V.: Algorithms for Finding Patterns in Strings. In: Leeuwen, J.V. (ed.) Handbook of Theoretical Computer Science, vol. A, pp. 257–297. Elsevier Science Publishers, Amsterdam (1990)
3. Jokinen, P., Tarhio, J., Ukkonen, E.: A Comparison of Approximate String Matching Algorithms. Software-Practice and Experience 26, 1439–1457 (1996)
4. Lecroq, T.: Experimental Results on String Matching Algorithms. Software-Practice and Experience 25, 727–765 (1995)
5. Baeza, R., Gonnet, G.H.: A New Approach to Text Searching. Comm. ACM 35, 74–81 (1992)
6. Cole, R., Hariharan, R., Paterson, M.: UriZwick: Tighter Lower Bounds on the Exact Complexity of String Matching. Siam J. Comput. 24, 30–45 (1995)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. McGrow Hill, New York (1990)
8. Knuth, D.E., Morris, J., Pratt, V.: Fast Patern Matching in Strings. SIAM J. of Comput. 6, 323–350 (1977)
9. Wagner, R.A.: The String-to-String Correction Problem. Journal of the ACM 21, 168–173 (1974)
10. Tarhio, J., Ukkonen, E.: Approximate Boyer-Moore String Matching. Siam J. Comput. 22, 243–260 (1993)
11. Cheng, H.D., Fu, K.S.: VLSI Architecture for String Matching and Pattern Matching. Pattern Recognition 20, 125–141 (1987)
12. Isenman, M.E., Shasha, D.E.: Performance and Architectural Issues for String Matching. IEEE Trans. on Computers 39(2), 238–250 (1990)
13. Mukherjee, A.: Hardware Algorithms for Determining Similarity between Two Strings. IEEE Trans. On Computers 38(4), 600–603 (1989)
14. Sastry, R., Ranganathan, N., Remedios, K.: CASM: A VLSI Chip for Approximate String Matching. IEEE Trans. on Pattern Analysis and Machine Intelligence 17, 824–830 (1995)
15. Park, J.H., George, K.M.: Parallel String Matching Algorithms Based on Dataflow. In: Proceedings of the 32nd Hawaii International Conference on System Sciences, vol. Track 3, IEEE, Los Alamitos (1999)
16. Tagliarini, G.A., Christ, J.F., Page, E.W.: Optimization using neural networks. IEEE Transaction On Computers 40(12), 1347–1358 (1991)
17. Hopfield, J.J., Tank, D.W.: 'Neural' Computation of Decisions in Optimization Problems. Biological Cybernetics 52, 141–152 (1985)
18. Chen, L., Aihara, K.: Chaotic simulated annealing by a neural network model with transient chaos. Neural Networks 8(6), 915–930 (1995)