EVOLUTION OF MULTIPLE STATES MACHINES FOR RECOGNITION OF ONLINE CURSIVE HANDWRITING

RAMIN HALAVATI, SAEED BAGHERI SHOURAKI, SAEED HASSANPOUR
Computer Engineering Department, Sharif University of Technology, Iran
halavati@ce.sharif.edu, bagheri-s@sharif.edu, hassanpour@ce.sharif.edu

ABSTRACT
Recognition of cursive handwritings such as Persian script is a hard task as there is no fixed segmentation and simultaneous segmentation and recognition is required. This paper presents a novel comparison method for such tasks which is based on a Multiple States Machine to perform robust elastic comparison of small segments with high speed through generation and maintenance of a set of concurrent possible hypotheses. The approach is implemented on Persian (Farsi) language using a typical feature set and a specific tailored genetic algorithm and the recognition and computation time is compared with dynamic programming comparison approach.

KEYWORDS: Online Handwriting Recognition, Elastic Pattern Matching, Evolutionary Training

1. INTRODUCTION
Recognition of cursive handwritings is a very hard task due to the none-existing segmentation points. Persian handwriting is among such handwritings as it is highly cursive, characters stick together and overlap, they can be written in several different forms, and each character has several forms when it is written in different positions of a word. These complexities make Persian one of the most complex handwritings for automatic recognition systems and therefore, there are few contributions trying to work out this problem.

To mention some, an approach has been introduced in [1] that uses a neural network to recognize online Arabic cursive or [2], [3], and [4] which recognize offline segmented Arabic handwriting or [5] which uses a decision tree to recognize segmented Arabic text. The major problem of all mentioned methods is the high-dependability on small writing perturbations and very vast variety of writing styles which results in slow, yet imprecise results.

In [6], Persian online handwriting is recognized using a set of fuzzy features and the comparison is done using pruned dynamic programming. The major point of the approach was to use small segments and a semi-simultaneous segment grouping and recognition. The approach presented good recognition rate and robustness versus perturbations, but its recognition speed was quite low in cases where the number of segments was more than 10-12 parts and also, no training method was specified.

To overcome these two problems, this paper presents a novel Multiple States Machine as a general tool for elastic pattern recognition and use an evolutionary approach to create these machines. The major idea behind the machines is to develop and maintain different hypotheses about the given sequence of segments and gradually prove or prune them to reach a single final decision.

The approach is presented as follows: Section two briefly presents the feature set and primary segmentation approach. Section three presents the multiple states machine comparison method and section four represents the evolutionary training system. Experimental results are stated in section five and at last come the conclusions and future works.

1 This handwriting is used in most of Persian or Arabic speaking countries, making a population of more than half billion people.
2. FEATURE SET AND SEGMENTATION

At the first step, the input is segmented into a sequence of lines and arcs using two simple heuristics which define line as a sequence of points with almost similar angles and an arc as a sequence of vectors with almost similar change of angles. When this is done, each segment is described using 4 features:

- Relative Length: the ration of segment's length to input length represented by 5 fuzzy terms, namely ignorable, short, average, long & very long.
- Angle: the angle from the first point of segment to its last point, represented by 8 fuzzy terms representing the 4 main directions and 4 sub-directions.
- Curvature: the ration of the direct distance between the first and the last point of the segment to the sum of distances between each to consecutive points, represented by 3 fuzzy sets, namely line, arc, & semi-circle.
- Direction of Curvature: whether the curvature is clockwise or counter-clockwise.

This feature set is used both for representing inputs and patterns. Figure 1 presents a sample input with its representation.

![Figure 1. A sample of segmentation and representation. Segments are separated using dots. C stands for Clockwise Curvature and CC for counter clockwise curvature.](image)

To compare a pattern with an input sequence of segments, one must find a mapping between the input segments and the pattern segments. It must be noted that this mapping is not necessarily one-to-one and one part of input either can be mapped to two or more pattern segments or no pattern. This is due to the unavoidable perturbations of primary segmentation which are a direct result of writer's speed and style, the writing device, the sampling rate and the segmentation algorithm. Thus, instead of trying to find the ideal segmentation, we will try to have a recognition approach with robustness versus segmentation.

3. THE MULTIPLE STATES MACHINE

To compare a pattern with a sequence of input segments, a mapping between the input segments and the pattern segments must be found. It must be noted that this mapping may not be one-to-one and one part of input may be mapped to zero or more pattern segments due to the unavoidable perturbations of recording and segmentation.

To make this mapping, we used the idea that the system must keep a pool of possible hypotheses on the mapping and follow them all, until some fail and some prove to be matching. To do so, each hypothesis can be assumed as a state of a machine and the state must specify the pattern segment that is to be matched now and the quality of the matching for the previous segments of the pattern. Hence, we define a state as a quintet (pos, pm, pn, cm, cn) with the following definitions:

- **Position (pos):** the current pattern segment to be matched.
- **Previous Match (pm):** the compatibility of the previous pattern segments with the previous input segments.
- **Previous Noise (pn):** the previously tolerated amount of noise.
- **Current Match (cm):** the matching length of previous input segments with the current pattern segment.
- **Current Noise (cn):** The amount of non-matching parts during the matching of current pattern segment.
Based on these definitions, and before a formal description of the matching algorithm, let's see an example of our ideal path of matching in Figure 2.

In this sample, the input has 4 segments (I1-I4) and the pattern has 3 segments (P1-P3). The matching sequence starts with a single hypothesis which is "we are reading pattern segment 1, there is no previous match, and nothing is sensed yet as a noise or a match which becomes (1,?,0,0,0)." The first input segment (I1) is an extra serif, it is regarded as a noise and its length is added to the current noise, making the current hypothesis (1,?,0,0,1). Then the second input stroke (I2) completely matches the first pattern segment (P1) and, the machine can move to accepting P2 state, and transfer the current noise to previous noise, as the state number is increased. The third input segment (I3) matches two pattern parts, P2 completely and P3 partly, but its similar part with P2 is not as long as P2 requires, thus, it decreases the matching to 75%, move to next pattern segment and increase its current match by half length of I3, which results in state (3,75%,1,4,0). Now, by receiving I4, the remaining of P3 is matched with I3 and we transfer to Done state, with 75% match and 1 noise.

To have the above ideal path of matching, we can assume a none deterministic finite states automata (NDFSA) with the following modifications:

1. We maintain a set of current states instead of just one current state and with each input, move to all next states, from all current states.
2. Each current state has the above specified five properties, where the first one states its position and the others, its matching features.
3. Once a current state loses some features, it is eliminated.

In this machine, no backward state transfer is allowed and based on some state transfer rules, other transfers may be done.

The machine starts with one current state (#1,100%,0,0,0) (we use 100% as the previous match instead of '?' as the first next pm will override it). With each input segment, it creates all possible next states from all current states based on 3 state transfer rules, namely Skip, Accept & March, and Accept, March & Wait, assuming Eq.(1) definitions.
Input Segments : $I_1, \ldots, I_m$
Pattern Parts : $P_1, \ldots, P_t$

\[ \text{Current State : } (\text{pos, pm, pn, cm, cn}) \]  \quad \text{Eq.(1)}

Length($I_1$): The length of input segment $x$.

\[ \text{LenMatch}(x, p): \text{The membership value of } x \text{ in fuzzy length specified for pattern part } p. \]

\[ \text{LowMatch}(I_1, P) = (E_{B_{ij}}, SM_{ij}, E_{A_{ij}}), \]

based on pairwise comparisons, representing $SM$ as the similar length of $i$ & $j$, and $EB$ & $EA$ as the nonesimilar length of $i$ & $j$ respectively before and after the similar part.

- **SKIP RULE**: By this rule, the input is ignored and regarded as a noise and added to $cn$ property. To make it formally, input segment $I$ can transfer state $(\text{pos, pm, pn, cm, cn})$ to state $(\text{pos, pm, pn, cm, cn} + \text{Length}(I))$. This rule applied in the first step of the previous sample.

- **"ACCEPT & MARCH" RULE**: This rule accepts the input as some part of the current pattern segment and possibly some next pattern segments. In this case, the current cell may move one or more cells ahead. As forward march can be done with different lengths, all possible next states are generated and added to the pool. This is specified formally in Eq. (2) and happened for input segments 2 and 4 of the previous sample.

\[
\begin{align*}
\text{if} & \quad \left\{ \\
& \quad pos + n \leq M + 1 \\
& \quad n = 1 \lor (\forall x; \text{pos} < x < \text{pos} + n \mid E_{B_{ij}} > 0) \\
& \quad n = 1 \lor (\forall x; \text{pos} \leq x < \text{pos} + n - 1 \mid E_{A_{ij}} > 0) \\
& \quad n > 0
\end{align*}
\]

\[ \text{Add to Next Possible States:} \]

\[ \begin{align*}
( \text{pos} + n, \\
& \quad \text{pm} \times \text{LenMatch}(cm + SM_{(pos)}) \times \prod_{\text{par}=c_{\text{par}+1}} \text{LenMatch}(SM_{(pos)}) \\
& \quad \text{pn} + E_{B_{(pos)}} + E_{A_{(pos+1)}}, 0, 0)
\end{align*} \]

- **"ACCEPT, MARCH, & WAIT" RULE**: This rule is quite similar to the previous one, with the minor difference that the given input segment may match a set of pattern segments, but it does not finalize the last matched pattern segment and let it be completed with later input segments. This is formally specified in Eq. (3) and took place in input segment 3 of the previous sample.

\[
\begin{align*}
\text{if} & \quad \left\{ \\
& \quad pos + n \leq M \\
& \quad n = 1 \lor (\forall x; \text{pos} < x < \text{pos} + n \mid E_{B_{ij}} > 0) \\
& \quad n = 1 \lor (\forall x; \text{pos} \leq x < \text{pos} + n - 1 \mid E_{A_{ij}} > 0) \\
& \quad n \geq 0
\end{align*}
\]

\[ \text{Add to Next Possible States:} \]

\[ \begin{align*}
( \text{pos} + n, \\
& \quad \text{pm} \times \text{LenMatch}(cm + SM_{(pos)}) \times \prod_{\text{par}=c_{\text{par}+1}} \text{LenMatch}(SM_{(pos)}) \\
& \quad \text{pn} + E_{B_{(pos)}} + \text{pos} \times [n \neq 0] + E_{A_{(pos+1)}}, [n = 0], \\
& \quad \text{SM}_{(pos+1)} \times \text{pos} \times [n = 0] + E_{A_{(pos)}})
\end{align*} \]

Using the above rules, all possible next states are generated by each input segment, but to avoid fusion of states, the new states whose $pm$ property is below a certain limit or either $pn$ or $cn$ properties are above a certain limit are pruned as they either show low match or high noise. Among the states that pass out of the last cell, the one which has the highest $pm$ is regarded as the similarity of the pattern and the input.
4. THE TRAINING ALGORITHM

To train the required machines, we have used a genetic algorithm with a specific mutation operator. Each chromosome includes the descriptions of all machines and each machine description consists of a sequence of segments, each described with the similar feature set as of section 2. The chromosomes are initiated using randomly selected samples from the training set. The cross over operator simply replaces the descriptions of some machines between two chromosomes, but the mutation operator is tailored specifically for manipulation of machine descriptions.

As expressed in Table 1, the mutations are targeted to make a meaningful change in the description such as breaking a line into two consecutive lines or an arc into two lines with a difference in angle. Using this type of mutations and defining the fitness function as the recognition rate of the entire chromosome, the genetic algorithm is run for enough iterations till the best machine reaches a certain recognition rate.

<table>
<thead>
<tr>
<th>Mutation Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change Type</td>
<td>Randomly choose a segment and change its type: line $\Rightarrow$ Arc $\Rightarrow$ Semi-Circle</td>
</tr>
<tr>
<td>Change Direction</td>
<td>Randomly choose a segment and change its fuzzy direction term to an adjacent one: Up $\Rightarrow$ Up-Left $\Rightarrow$ Left $\Rightarrow$ Bottom-Left $\Rightarrow$ Bottom $\Rightarrow$ Bottom-Right $\Rightarrow$ Right $\Rightarrow$ Up-Right $\Rightarrow$ Up.</td>
</tr>
<tr>
<td>Change Length</td>
<td>Randomly choose a segment and change its fuzzy length term to an adjacent one. Ignorable $\Rightarrow$ Short $\Rightarrow$ Medium $\Rightarrow$ Long $\Rightarrow$ Very Long</td>
</tr>
<tr>
<td>Merge</td>
<td>Randomly choose two consecutive segments and merge them. The type, direction and length of the new segment are selected based on the previous two.</td>
</tr>
<tr>
<td>Split</td>
<td>Randomly choose a segment and split it into two segments by a mixture of following rules: Type: Line $\Rightarrow$ Line+Line, Arc $\Rightarrow$ Line/Arc+Line/Arc, SemiCircle $\Rightarrow$ Arc/SemiCircle+ Arc/SemiCircle. Direction: $\Rightarrow$ (Direction-1)+ (Direction+1). Length: Short $\Rightarrow$ Ignorable + Ignorable Medium $\Rightarrow$ Short+ Short Long $\Rightarrow$ Medium + Medium Very Long $\Rightarrow$ Long + Long</td>
</tr>
</tbody>
</table>

5. EXPERIMENTAL RESULTS

As there is no generally agreed test set for Persian handwriting, we collected a sample set from 10 different writers and each one writing all Persian characters 10 times. The patterns where evolved by 10 minutes of the genetic algorithm on a Pentium IV, 2.8 GHz in 10 different trails, Table 2 presents the average and best recognition results.

Also to test the speed performance of the presented approach, we have compared it with the dynamic programming algorithm of [6]. As depicted in Figure 3, while the DP approach's computation time increases exponentially by increasing the number of input segments, the MSM approach grows linearly and MSM is at least 100 times faster, when the number of segments exceeds 13 segments which is quite common for fully script languages.

And at last, to check the possible fusion of states in MSM, we logged the average and maximum number of simultaneous states during the above tests which resulted in average 10 and maximum 34 concurrent states.
Table 2: Recognition Rate of the two approaches

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Dictionary</td>
<td>84.5%</td>
<td>89%</td>
</tr>
<tr>
<td>Using Dictionary</td>
<td>92.5%</td>
<td>96.1%</td>
</tr>
</tbody>
</table>

Figure 3: Recognition speed of the two approaches, the horizontal axis represents the number of segments and the vertical axis represents time, in milliseconds.

6. CONCLUSIONS AND FUTURE WORKS

The elastic mapping of input segments and pattern segments is a vital component of online handwriting recognition systems for cursive languages such as Persian. This paper presented a novel approach based on a Multiple States Machine which generates and propagates a set of concurrent hypotheses on the given inputs and automatically designs the structure of machines using a specifically designed genetic algorithm.

The approach is tested over a set of Persian language test cases with 89% best recognition rate without dictionary and 96.1 with dictionary. It is also compared with pruned dynamic programming, showing an almost constant recognition speed while DP's computational time increases exponentially when the number of segments increase, resulting in more than 10 times faster results for 9 segment words and 100 times faster results for words with 13 segments.

As a next step, we are developing an approach for parallel comparison of multiple patterns and also testing this approach on speech recognition samples.

7. REFERENCES