Sub-word Based Offline Handwritten Farsi Word Recognition Using Recurrent Neural Network

Mohammad Fazel Younessy Ghadikolaie, Ehsanolah Kabir, and Farbod Razzazi

In this paper, we present a segmentation-based method for offline Farsi handwritten word recognition. Although most segmentation-based systems suffer from segmentation errors within the first stages of recognition, using the inherent features of the Farsi writing script, we have segmented the words into sub-words. Instead of using a single complex classifier with many (*N*) output classes, we have created *N* simple recurrent neural network classifiers, each having only true/false outputs with the ability to recognize sub-words. Through the extraction of the number of sub-words in each word, and labeling the position of each sub-word (beginning/middle/end), many of the sub-word classifiers can be pruned, and a few remaining sub-word classifiers can be evaluated during the sub-word recognition stage. The candidate sub-words are then joined together and the closest word from the lexicon is chosen. The proposed method was evaluated using the Iranshahr database, which consists of 17,000 samples of Iranian handwritten city names. The results show the high recognition accuracy of the proposed method.

Keywords: OCR, Handwritten recognition, Sub-word, PAW, Recurrent Neural Network, Farsi, Persian, Arabic.

Manuscript received June 15, 2015; revised Feb. 29, 2016; accepted Apr. 5, 2016.

Mohammad Fazel Younessy Ghadikolaie (corresponding author, mf.younessy@qaemshahriau.ac.ir) and Farbod Razzazi (razzazi@srbiau.ac.ir) are with the Department of Electrical and Computer Engineering, Islamic Azad University, Science and Research Branch, Tehran, Iran.

Ehsanolah Kabir (kabir@modrares.ac.ir) is with the Department of Electrical and Computer Engineering, Tarbiat Modarres University, Tehran, Iran.

I. Introduction

Although people these days use computers, tablets, and mobile phones to generate data, writing on paper remains a commonly used method of documentation. Optical scanning is a proper method for preserving digital copies of such documents. Moreover, to reduce the volume of such data saved on computers, an accurate optical character recognition method is needed. Certain applications including processing, automatic bank check reading, and postal address or zip code reading are also necessary.

In general, an offline recognition system consists of digital image scanning. A preprocessing stage is generally needed to enhance the quality of the image. After this phase, the image may be divided into segments, such as sub-words, letters, and even sub-letters. The next stage is feature extraction. Two types of features are typically generated by a system: statistical and structural. Structural features have visual writing properties including dots, loops, branch points, and end points, whereas statistical features, such as the density of black pixels, moments, and Fourier descriptors, are measured numerically from an image. Such features are passed to the classifier stage to be used in the training and testing phase. Some of the most commonly used classifiers in this field include a hidden Markov model (HMM), neural network (NN), and *k*-nearest neighbors (*k*-NN). Finally, post processing is a good idea for correcting and improving the resulting accuracy based on the lexicon. A lexicon is a set of words in the vocabulary of the system, which make the classes. To read more about offline Farsi/Arabic handwriting recognition methods, refer to [1] through [3].

Recognition systems generally use a holistic or segmentation-based method. Segmentation-based systems [4]–

[7] break a word into parts called segments, which can be a letter or even a sub-letter, and try to recognize each segment individually and concatenate the results. On the other hand, holistic approaches see a word as a whole, and apply a recognition phase [8]–[10]. Segmentation-based systems generally suffer from low segmentation accuracy and may generate errors in the first stages of recognition. In contrast, holistic approaches may fail in large lexicon systems, whereas segmentation-based (or analytic) systems are used in large lexicon systems.

More than 300 million people around the world speak Farsi, Arabic, and their derivatives [11]. Farsi and Arabic are naturally written cursively in both handwritten and typewritten modes. Farsi and Arabic are written similarly with some slight differences, such as the pronunciation and meaning of certain words. In comparison to Latin, Farsi and Arabic seem to be more complex. For example, many letters in these languages have complementary diacritics such as dots, madda, and zigzag bars. In addition, the letters have different shapes at different locations of the word. Interestingly, Farsi and Arabic writing, both in handwritten and printed form, are naturally written in certain parts called sub-words. A sub-word consists of one or a few characters connected to each other. A word may have one or more sub-words.

In comparison to Latin and Chinese handwriting recognition systems, there are only a few approaches dedicated to Farsi and Arabic. The initial works were conducted in the 1970s [2]. In [12], a holistic approach using an M-band packet wavelet transform is applied. In [8], Dehghan reported a holistic system in which a discrete HMM and Kohonen self-organizing vector quantization have been used as classifiers. In [9], [13], and [14], a slanted frame feature extractor is used and combined with HMM-based classifiers. A segmentation-based system was proposed in [15]. The researchers divided words into sub-words, and then into candidate letters, and applied a *k*-NN classifier. A combination of two HMM-based models is used in [16] along with some statistical features derived from a sliding window.

In this work, we propose a segmentation-based system. We divide a word into its sub-words, which has few errors in comparison to systems that divide a word into characters or even sub-characters. Some key statistical features are extracted from each sub-word. A recurrent neural network (RNN) classifier is then used to recognize each sub-word class. In the final stage, the recognized sub-word candidates are joined together to make a word from a lexicon. In the following section, we provide a brief introduction to the Farsi language. Section III describes the details of our proposed recognition method. In Section IV, we report our experimental results, and in the final section, some concluding remarks and areas of future work are provided.

Table 1. Farsi characters in four possible positions.

		1	2	3	4
	Character	Beginning	Middle	End	Isolate
1	Alef	ا (ا)	ـا	ـا	ا (ا)
2	Be	بـ	ـبـ	ـب	ب
3	Pe *	پـ	ـپـ	ـپ	پ
4	Te	تـ	ـتـ	ـت	ت
5	Se	ثـ	ـثـ	ـث	ث
6	Jim	جـ	ـجـ	ـج	ج
7	Che *	چـ	ـچـ	ـچ	چ
8	He	حـ	ـحـ	ـح	ح
9	Khe	خـ	ـخـ	ـخ	خ
10	Dal	د	ـد	ـد	د
11	Zal	ذ	ـذ	ـذ	ذ
12	Re	ر	ـر	ـر	ر
13	Ze	ز	ـز	ـز	ز
14	Zhe *	ژ	ـژ	ـژ	ژ
15	Sin	سـ	ـسـ	ـس	س
16	Shin	شـ	ـشـ	ـش	ش
17	Sad	صـ	ـصـ	ـص	ص
18	Zad	ضـ	ـضـ	ـض	ض
19	Ta	طـ	ـطـ	ـط	ط
20	Za	ظـ	ـظـ	ـظ	ظ
21	Ayn	عـ	ـعـ	ـع	ع
22	Ghayn	غـ	ـغـ	ـغ	غ
23	Fe	فـ	ـفـ	ـف	ف
24	Ghaf	قـ	ـقـ	ـق	ق
25	Kaf	کـ	ـکـ	ـک	ک
26	Gaf *	گـ	ـگـ	ـگ	گ
27	Lam	لـ	ـلـ	ـل	ل
28	Mim	مـ	ـمـ	ـم	م
29	Noon	نـ	ـنـ	ـن	ن
30	Waw	و	ـو	ـو	و
31	He	هـ	ـهـ	ـه	ه
32	Ye	یـ	ـیـ	ـی	ی

II. Characteristics of Farsi Handwriting

In this section, we provide a brief introduction of the Farsi (and Arabic) language. The important aspects of this language are listed below:

1) The Farsi language has 32 letters (Arabic has 28 letters)

http://dx.doi.org/10.4218/etrij.16.0115.0542

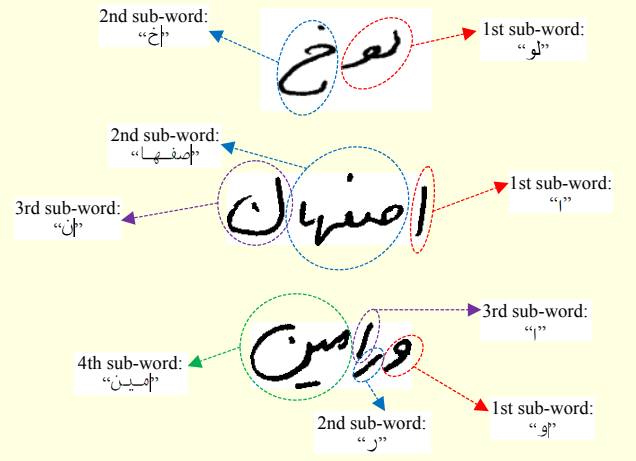


Fig. 1. Examples of some sub-words.

written from right to left.

2) Farsi and Arabic are always written cursively in both handwritten and typewritten form.
3) Depending on its location (beginning, middle, end, and isolated form), each letter can have up to four different forms. Table 1 shows the Farsi letters and their different forms at different positions. The letters marked by a star are not included in the Arabic language.
4) More than half of the letters have complementary diacritics such as upper or lower dots, a zigzag bar, or madda.
5) Seven common letters (ا , د , ذ , ر , ز , ژ , and و) have only isolated and ending forms. This means they only connect from the right side to other letters, or appear in isolated form.

The last aspect of this language is an interesting feature. Such ending letters divide the words into parts called sub-words (in Arabic, this is called a piece of Arabic word, or PAW). Figure 1 shows some examples of the segmentation of a word into two, three, or four sub-words.

We define a position for each sub-word within a word. For example, the lower word in Fig. 1, which has four sub-words, has one sub-word at position 1 (beginning of the word), one at position 2 (middle right), one at position 3 (middle left), and finally, one at position 4 (the end).

III. Proposed Word Recognition System

1. System Overview

An overview of our system is shown in Fig. 2. As shown in Fig. 2, the system consists of training and recognition phases. In the training phase, the system first sends all available words in a database to the preprocessing unit with enhancement operations, such as noise cancellation and baseline detection. A sub-word extraction algorithm then applies the segmentation and breaks the words into sub-words.

Let us define each sub-word as s, which is formulated as follows:

$$S = \{s_i | i = 1, \ldots, N\}, \tag{1}$$

where N is the total number of all sub-word classes among all words available in a database.

In the sub-word extraction phase, we process the database, and determine each word's sub-word count, as well as the position of each sub-word within a word (we define the position in Section II). Please note that a sub-word might exist in only one or more classes of words, even at different positions.

As a result, for each sub-word, s_i, we defined two vectors, L_i and G_i. Here, L_i is a vector containing all possible positions where a sub-word can be located within all sub-word positions in the database, and G_i is a vector consisting of the numbers of sub-words of a word that s_i can be used in. The expression below shows these two sets.

$$\begin{aligned} \mathbb{L} &= \{L_i | i = 1, \ldots, N\}, \\ \mathbb{G} &= \{G_i | i = 1, \ldots, N\}. \end{aligned} \tag{2}$$

In Section III, the extraction of the features used in this system is completely expressed.

In our system, we do not use a conventional method utilizing a single complex classifier with many outputs and multiple layers, which is difficult to train and test. Instead, for each sub-word, one RNN is used with a true/false output. This means we have many simple classifiers, each responsible for recognizing one sub-word class. The set of RNNs is shown below.

$$\text{RNNs} : \{\text{RNN}_1, \ldots, \text{RNN}_N\}. \tag{3}$$

Each RNN corresponds to each sub-word class, and has been trained by applying a sub-word feature vector to a neural network.

At this point, after the training phase, the system is ready to start the recognition phase. Similar to the training phase, a preprocessing of the word image is applied, and sub-words inside the test word are then extracted and labeled.

Let us define an extracted sub-word set for input word image, I, as below:

$$\text{S}_\text{R} = \{\text{s}_{\text{R}\,j} | j = 1, \ldots, M\}. \tag{4}$$

Now, instead of a single word image (I), we have some sub-word images (S_R). The position of each extracted sub-word, (I_j), and the number of sub-words inside a word image, M, are saved. Our task is to recognize each sub-word image, S_R, and at the end concatenate all sub-words to recognize the final word.

After the sub-word extraction, the features of each sub-word

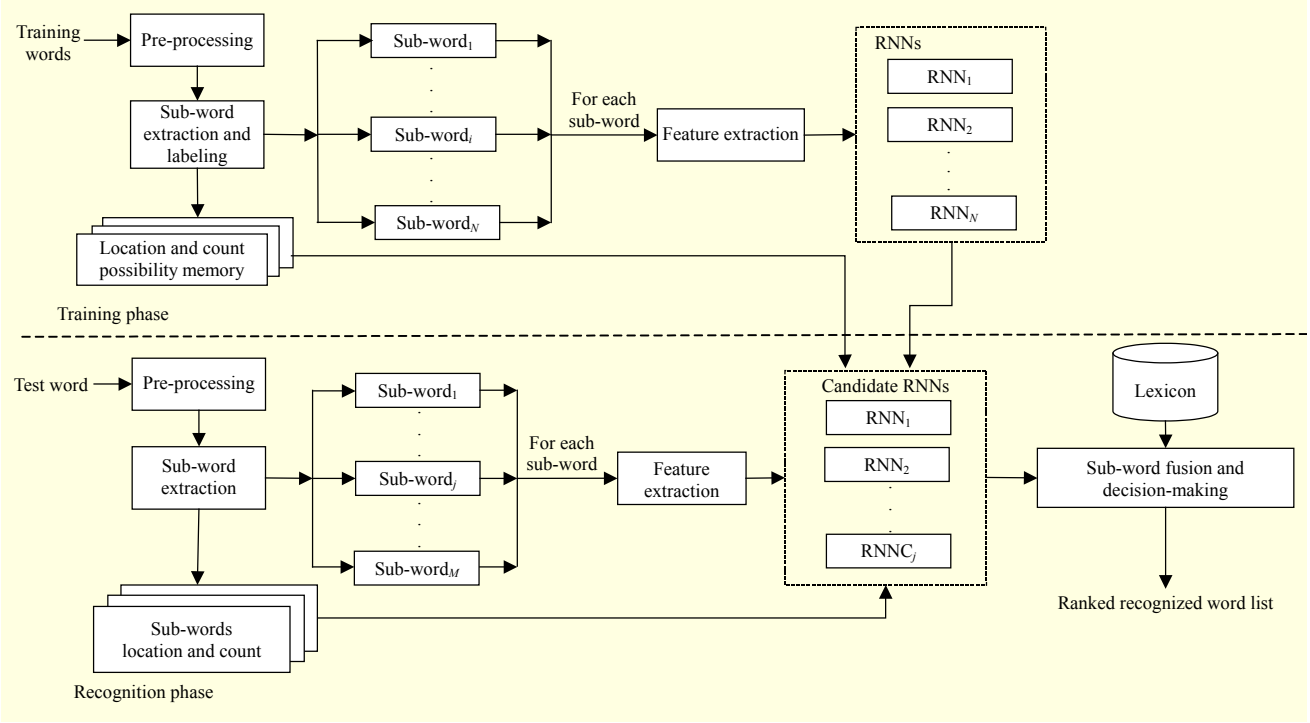


Fig. 2. Block diagram of proposed system.

are calculated and saved. For the next step, depending on the position of each sub-word, as well as the number of sub-words within a word, and using the sets $\mathbb{L}$ and $\mathbb{G}$, the proper RNN classifiers are chosen, and each sub-word is passed to the candidate sub-word RNN classifiers. This is an important point because many classes will be pruned for the current test sub-word, and the final recognition will be achieved among a limited set of sub-word classifiers.

Ideally, each classifier has a true/false result. This means a classifier belonging to a sub-word has a "true" result if the input sub-word is the same as the trained sub-word; otherwise, a "false" result occurs if the input word is not a trained sub-word.

Thus, the candidate classifiers are chosen based on two criteria:

- Training using sub-words that are capable of being at the same position as the unknown extracted sub-word.
- Training using sub-words that are used in words whose total number of sub-words is equal to the number of sub-words of the unknown input word.

In other words, candidate networks set for sub-word $s_{R\,j}$ can be expressed as (5).

$$\left.\begin{array}{l}\forall k, L_k \bigcap l_j \neq \varnothing \\ M \in G_k\end{array}\right\} \rightarrow \mathrm{RNN}_k \in \text{Candidate RNNs}. \quad (5)$$

For example, if the test word originally has four sub-words, it means we have a sub-word at position 1 (beginning of the word), position 2 (middle-right), position 3 (middle-left), and position 4 (end of the word). First, the number of sub-words (four, here) indicates that a reduced set of words in a lexicon is valid for this word image. Second, for each position, for example, position 1, we have a limited set of sub-word classifiers that can be used in this word with four sub-words at position 1. Hence, the total number of RNN classifier candidates is reduced. This approach dramatically reduces the recognition error and time because irrelevant results are pruned.

Finally, we have some recognized sub-words candidates for each sub-word position. Through the fusion of these candidates, we can find the word representing the original image from the lexicon.

Each of the steps mentioned above are described in detail in the following sub-sections.

2. Preprocessing

A. Noise Removal

Our images in the database are scanned in black and white. However, some small dots mainly occur because of the binarization, and hence, we omitted these images through a morphological closing and opening operation using a 3 × 3 disk-shaped structure element [17], [18].

B. Word-Breaks Correction

In a few of the images, some parts of the word are broken into two parts. These the parts have a space of less than 2 pixels,

Fig. 3. Example of word-break as a result of binarization.

Fig. 4. Incorrect baselines (red dashed lines) versus the correct baseline (green solid line).

which is very short, in comparison to the sub-word spaces. Figure 3 shows an example of this break in the second sub-word, where the above part is disconnected from the lower part (red circle). Thus, we attached the connected components that are very close to each other and have a space of only one or two pixels between them.

C. Baseline Detection and Slope Correction

Although the baseline detection in a single word, particularly a handwritten word, is difficult to achieve, we used the common method of utilizing a horizontal projection profile of the word image [6], [19], [20] with some modifications. Through this method, the black pixels of the binary image are added to the x-axis coordinate. We applied a moving-average filter with length N ($N = 9$) to the horizontal profile to smooth it and remove unwanted spikes. We then saved the local maximum of each hill. The peak of these local maximums indicates the baseline position. In some rare cases, the baseline is misplaced. Figure 4 shows an example in which the baseline is calculated incorrectly at the top and bottom of the image. We therefore added some simple criteria to avoid this problem: if the baseline is too close to the upper or lower image, it is ignored, and we switch to the next local maximum and recheck the criteria. In Fig. 4, the red dashed lines are extracted based on the maximum local peaks, but are not correct (being so close to the vertical extremes of the image), and the third candidate (green solid line) is properly chosen.

Although a Farsi manuscript is written on a baseline, we define a zone containing the baseline, which is called the core zone. The core zone is derived from the first local minimums of the horizontal profile above and below the baseline. These local minimums should not be greater than 80% and less than 25% of the baseline peak; otherwise, the next minimum will be chosen or the zone will be limited to 25% of the peak, respectively.

To correct the small slope in the image, we also rotated the image from –5 to +5 degrees with a step of 0.5 degrees, and recalculated the maximum peak of the horizontal projection, saving the peak values for each rotation. The maximum of these peaks shows us the best rotation that should be applied to the original image.

3. Sub-word Extraction Algorithm

In this section, we focus on a segmentation of words into their constitutive sub-words. Fortunately, sub-words are written in a segmented state; however, the existence of dot(s), madda, and zigzag bars makes the segmentation a challenge. We call the parts in a word a connected component (CC). A CC is a set of black pixels connected to each other. A sub-word itself consists of one principal CC, which we call the "main part," and some other CCs (such as upper/lower dots, madda, and zigzag bars), which are called "complementary parts."

Figure 5 shows a city name consisting of six sub-words and

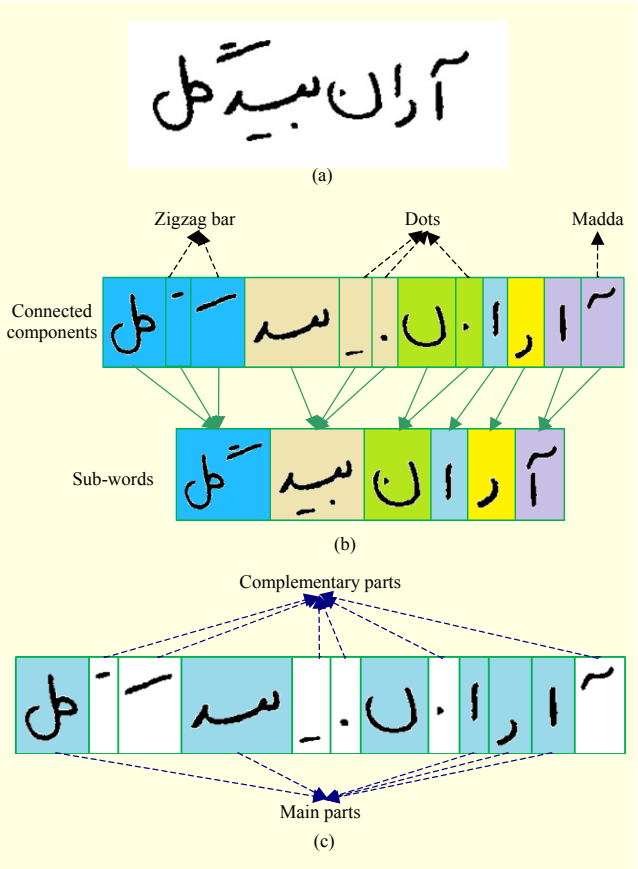


Fig. 5. CCs of the city name "آران بیدگل." (a) Main word, (b) CCs in the upper figure and the corresponding sub-words in the lower table, and (c) main part and complementary parts of the word.

12 CCs. Thus, our algorithm should change these CCs into six sub-words by determining which CC is the main part and which is the complementary part, as shown in Fig. 5(c).

We divide the word image into five zones. The core zone is the zone containing the baseline. Each of the upper and lower remaining parts of the image are then divided into two equal zones separately. We can see these five zones in Fig. 6. We labeled the zones from 1 to 5, where zone 1 is the core zone. The thicknesses of zones 2 and 3 are equal, as are those for zones 4 and 5.

Thus, after the detection of each CC [21], according to the characteristics of the Farsi language, we run the following criteria (Algorithm 1) for each CC to extract the main and complementary parts.

Algorithm 1. Main/complementary part extraction algorithm.

Suppose N_C is the number of CCs inside a word image.
For $i = 1$ to N_C *do* the following criteria check:
If CC(i) is located in three zones or more *then* it is a main part;
Else if CC(i) is located in zone 2 or 5 (the upper- and lower-most zones), *then* it is a complementary part (a madda, zigzag bar, or dot);
Else if CC(i) is located in zone 3, and its bottom has no other CCs with more than a 20% vertical overlap, *then* it is a main part (not an upper dot); *else* it is a complementary part;
Else if CC(i) is located in zone 4, and there are no other CCs above it with more than a 20% vertical overlap, *then* it is the main part (not a lower dot); *else* it is a complementary part;
Else if CC(i) is located in zones 2 and 3, and CC(i) below it is another CC with more than a 20% vertical overlap, *then* it is a complementary part (a zigzag bar or madda);
Else if CC(i) is located in zones 4 and 5, and there is at least one other CC above it with more than a 20% vertical overlap, *then* it is a complementary part;
Else if CC(i) and another CC(j) are both in zone 1 (core zone), and have a complete vertical projection overlap, *then* the bigger one is a main part and the smaller one is a complementary part;
Else if CC(i) is located in zones 1 and 3, and no other CCs exists below it, or its vertical overlap to other CCs is less than 20%, *then* it is a main part; *Else* it is a complementary part;
Else if CC(i) is a main part.
Else if .
Else for.

After labeling the CCs as the main or complementary parts, we should generate the sub-words. The complementary parts should be attached to the related main parts. We simply attach the complementary parts to the nearest possible main part and label them as a single sub-word. For example, in Fig. 5(c), the two zigzag bars will be attached to the main part CC located at the left-most area, and the madda will be attached to the main part CC located at the right-most area of the city image; in addition, as complimentary parts, the dots will also be attached to the nearest main part.

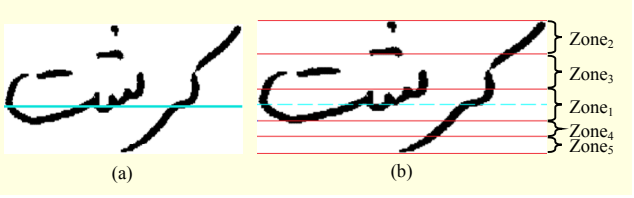


Fig. 6. (a) Word with baseline and (b) zones of the word image.

The sub-words are now ready, and we have some images containing sub-words with known locations that should be passed on to the next stage, that is, the feature extraction.

4. Feature Extraction

Our feature extraction method is based on a sliding window with a local feature extraction technique, which is very common in offline text recognition [8], [9], [13], [14], [22]. Using such methods, we can make a temporal (spatial) set of features across an image by shifting the sliding window from right to left along the image. While shifting, the feature vector is calculated for each frame.

Our method is similar to that in [8] with certain modifications and improvements. The image area of the sub-word is divided into five horizontal zones similar to the sub-word extraction section. We also divide the image into a set of vertical fixed-width frames with a width of W_F, from right to left. There is an overlap of W_{OV} between two neighbor frames. Overlapping is an interesting phenomenon preserving the connectivity of the frames. Using our simulation results, we fixed W_F to eight pixels and W_{OV} to six pixels. Note that the height and total number of frames for an image depend on the size of each image. Figure 7 shows an example of such frames and zones.

We then extract the feature vectors for each sub-word. Here, F_i is a feature vector inside each window, and sliding from right to left makes V number of feature vectors for each sub-word image, as given by the following relation:

$$\mathbb{F} = \left[F_1 : \ldots : F_V\right]_{U \times V},$$
$$F_i = \begin{bmatrix} f_1 \\ \vdots \\ f_U \end{bmatrix}, \tag{6}$$

where U is the dimension of each feature vector, F_i.

Here, the sliding window is shifted over the contour of the sub-word image. We extract 30 features inside each frame ($U = 30$). Each frame has five zones in which the histogram of the contour chain code [23], [24] is calculated. There are four possible slopes for each contour direction (0°, 45°, 90°, 135°), hence making four components in each zone, and a total of 20 feature components in one frame (f_1 through f_{20}). We normalize the feature components by dividing the

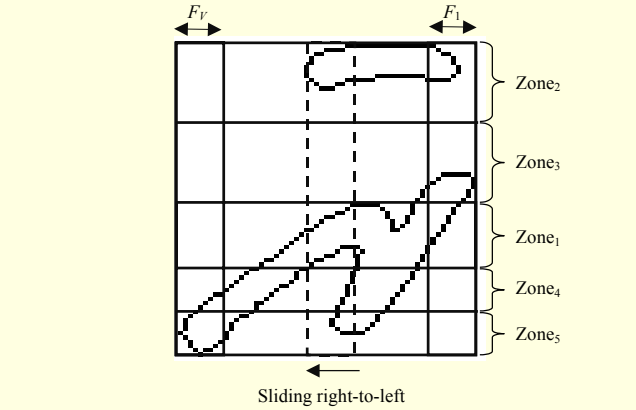


Fig. 7. Feature extraction frames and zones.

respective histogram to the height of each zone to remove the effect of the height of the image.

We also added some other features to the feature vector. Next, feature (f_{21}) is the number of black pixels (foreground) divided by the total number of pixels inside the frame. Feature f_{22} is the center of gravity of the black pixels in the frame. The eight features, f_{23} through f_{30}, are the number of vertical transitions from black to white, or vice versa, in each vertical column of pixels (in our work, the width of the frame is fixed to eight pixels).

As a result, the feature vector consists of $U = 30$ features that characterize the sub-word image within a frame. The total number of frames (V) is dependent on the length of the test sub-word. The matrix of the features ($U \times V$), which indicates their temporal sequence, is ready to be passed to the RNN classifiers.

5. Classifier

In our work, instead of using one huge classifier with multiple outputs, we employed several small classifiers (equal to the number of sub-word classes) with only two outputs (true/false). In our case, each class (sub-word) has its own classifier. Ideally, if a sample of this sub-word is given to the classifier, the output is true, and if other sub-words are fed into the classifier, the output is false. By counting the number of sub-words in each word image, and also knowing the position of each extracted sub-word, only some of these small classifiers are tested, and hence a good performance is guaranteed.

In our work, we used an RNN classifier, which is a simple neural network with a memory unit added to it, allowing the sequence of inputs to be learned. An RNN, as opposed to a neural network, has a sequence (time series) of inputs and outputs. We expect a classifier to select a class at the end, not a sequence of outputs. Thus, a decision layer should be added to the output of the RNN. Some valuable studies on this have

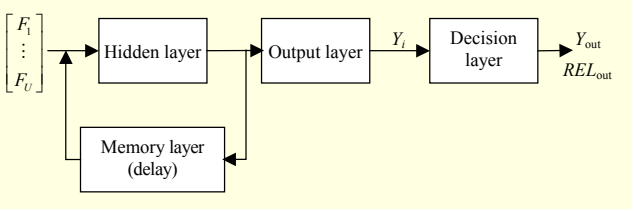


Fig. 8. Block diagram of RNN classifier and decision layer.

been conducted [25]–[27]; however, they all seem to be very complex mathematical paradigms that make the classifier more complicated and slow. In contrast, we are trying to create simple classifiers, and instead use the characteristics of the Farsi language to help the classifier.

Figure 8 shows our classifier topology. The RNN has an input vector of 30 elements, a hidden layer, and an output layer. A feedback with a delay of T_{delay} was fed into the hidden layer as well. An output of the RNN consists of a tan-sigmoid function, which means the outputs are between –1 and +1 (instead of a discrete true/false output). Note that, as long as the input to the RNN is a two-dimensional (2D) sequence of features ($U \times V$), the output is also a 2D sequence ($2 \times V$). The positive outputs indicate that the input test sequence belongs to this classifier, whereas the negatives show that it does not. The decision layer, which was added to the output layer, makes one final output from the sequence of 2D outputs. The decision layer should give the probability of the test sub-word belonging to the classifier.

Suppose that V is the length of the input (features) sequence, y_{1i} is the true output of the RNN output layer, and y_{2i} is the false output of the RNN output layer.

$$Y_i = \begin{cases} 1, & \text{if } y_{1i} > y_{2i}, \\ 0, & \text{else}, \end{cases}$$
$$Y_{out} = \sum_{i=T_{delay}}^{V} \frac{Y_i}{V - T_{delay}} \times 100. \quad (7)$$

Y_{out} is the output of the decision layer. Note that T_{delay} is a system parameter showing the network recurrence time. In our simulation, $T_{delay} = 3$ worked best for us. The RNN outputs prior to that seemed to be valueless, and we therefore omitted them. The above equation shows that Y_{out} will be 100% if a sub-word similar to the classifier's label is ideally fed into, and zero percent if a non-similar sub-word is fed into. In a real case, Y_{out} is the matching probability of the input sequence to the specific sub-word classifier.

Our simulations showed that if we apply the input sequence to the classifiers in reverse order, we obtain a better result (about a 3% to 4% improvement); hence, in all of the classifiers, we first reversed the order of the feature sequence, and then fed them into the network. This is due to the fact that people tend to write the beginning of the sub-words more carefully than the end.

We also created a reliability function, REL_{out}, for the classifier at the decision layer. This reliability function, in fact, is based on the distance between y_{1i} and y_{2i} according to (8) in which Y_i is made. The longer the distance is, the more reliable the decision. This reliability helps in choosing better results in the next stage, sub-word fusion.

$$REL_{\text{out}} = \frac{1}{2} \sum_{i=T_{\text{delay}}}^{V} \frac{|y_{1i} - y_{2i}|}{V - T_{\text{delay}}}. \quad (8)$$

A. RNN Training

Our database has a large number of sub-words, each of which has a few samples. As the best case, if we train the RNN normally, most of the classifiers will be trained by about 95% false inputs and 5% true inputs. This leads the classifiers to be trained mostly by false inputs, and as a result, the classifiers mostly generate a false output at the test time. Thus, we increased the number of true samples virtually by adding some Gaussian noise to the true samples, making them about one-third the number of samples.

The input data, which are in the form of a feature matrix, should be standardized to have a mean of zero and a standard deviation of 1. Although the information does not change when using this procedure of adding input values suitable for activation functions, the performance improves significantly [28]. For the initial weights of the network, we used a Gaussian distribution with a mean of zero and standard deviation of 0.1. Although most of the training algorithms work fine for us, the resilient back-propagation (Rprop) algorithm [29] seems to work better.

We divided the data into three parts: 70% for training, 15% for training validation, and the remaining 15% for testing the network.

6. Sub-word Fusion

The goal of our work is to find a word, not a sub-word. Thus, after finding sub-words and applying them to the classifiers, we have a list of candidates for each sub-word position. For example, for a word having four sub-words, we have four lists of candidates with known locations. In Fig. 9, we inserted a sorted list under each sub-word location, with the highest probability being at the top of the list. Here, P_{ij} shows the probability of the candidate being at sub-word location i, and is in fact Y_{out} calculated in (7) for a specific classifier. Note that for location 1, we have C_1 possible sub-word candidates, and so on for the other locations.

The final word will be formed after placing together the candidates. The candidates are derived from the proper classifiers that are allowed to be applied to the sub-words. Using the output probability of the classifiers and the reliability criterion, we calculate the minimum cost of the edit distance (or replacement in our case) to make a word in a lexicon. Note that, to mostly quickly achieve the best word, we first try to replace the sub-word candidate with both the lowest probability and the lowest reliability function (REL_{out}). This finally leads to the list of word candidates for the test image.

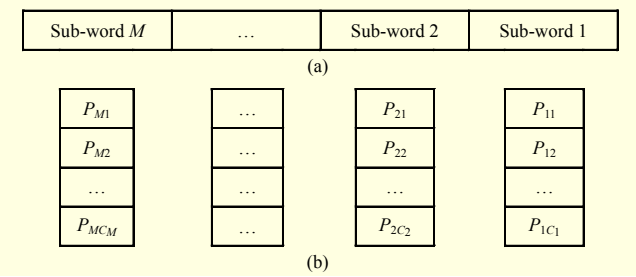


Fig. 9. (a) Sub-word locations and (b) sorted candidate probability list for each location.

IV. Experimental Results

1. Database

We tested our system on the Iranshahr database. In this database, there are 502 binary images of Iranian city names. There are about 17,000 images in the database, which means that more than 30 samples are ready for each word class. The database has also been used in [30]. There are also a total of 425 sub-word classes. The city names are constructed from one sub-word to a maximum of seven sub-words. Table 2 shows the characteristics of this database. For example, the second row shows that there are 160 words that have two sub-words. In addition, for a candidate sub-word at location 1 (beginning of the word) of a word with two sub-words, there are 112 possibilities, whereas there are 93 possibilities for location 2 (end of the word). As this table shows, the words with more sub-words have a lower possibility in terms of their sub-word positions, and hence have better results through our system.

2. Sub-word Extraction Results

Our simulation shows that the system finds the sub-words efficiently. A sub-word segmentation error rate of less than 8% was calculated. About 1% of these errors are related to misplaced dots or other diacritics in the database. Most of the other cases are over- or under-segmentation errors, in which the system finds more or fewer sub-words than expected. Most of the errors are related to under-segmentation. Table 3 shows an error in segmentation based on the number of sub-words. The

http://dx.doi.org/10.4218/etrij.16.0115.0542

Table 2. Characteristics of Iranshahr database.

No. of sub-words	No. of words	Location of sub-word	No. of possible sub-word candidates
1	36	1	36
2	160	1	112
		2	93
3	163	1	80
		2	74
		3	60
4	105	1	46
		2	42
		3	35
		4	43
5	32	1	22
		2	15
		3	12
		4	15
		5	14
6	6	1	4
		2	5
		3	5
		4	4
		5	5
		6	5
7	1	×	1

Table 3. Sub-word segmentation error.

No. of sub-words	Over-segmentation error rate	Under-segmentation error rate
1	0 %	0 %
2	1.2 %	4.1 %
3	1.5 %	5.3 %
4	2.2 %	6.6 %
5	2.1 %	7.5 %
6	1.8 %	5.6 %
7	0 %	2.2 %
Overall error	1.6 %	5.3 %

last row shows a total over-segmentation rate of 1.6% and under-segmentation rate of 5.3% (total error rate of 6.9%).

We verified the errors visually, and found that the problem is mostly related to human writing mistakes, which make the words difficult to read even by the human eye. Some examples of such errors are shown in Table 4.

Table 4. Examples of handwriting mistake and their wrong sub-words extraction results.

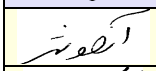
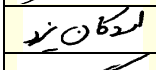
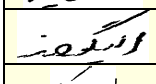
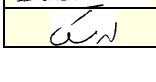
Sample	Typing form	Algorithm result	Correct number
	آزادشهر	3	5
	اردکان یزد	4	7
	الیگودرز	3	5
	ارسک	2	3

Table 5. System recognition results.

Method	Recognition rate			
	Top 1	Top 2	Top 5	Top 10
With segmentation error	84.3%	85.1%	88.5%	90.5%
Without segmentation error	90.3%	91.5%	93.5%	94.5%

Table 6. Comparisons with other word recognition systems.

Method	Recognition rate			
	Top 1	Top 2	Top 5	Top 10
Proposed method	84.3%	85.1%	88.5%	90.5%
M-band packet wavelet [12]	75.5%	76.4%	80.1%	82.2%
HMM [8]	63%	65.1%	67.4%	71%

3. Word Recognition Results

The results when applying the proposed method to the database are shown in Table 5. The recognition rates with and without considering the segmentation errors are also shown.

As shown above, the proposed method achieves an acceptable recognition rate, especially when not considering the segmentation errors.

To compare the results with those of other works, we applied the methods presented in (8) and (12) to our database. Table 6 shows the results along with those of the proposed method.

V. Conclusion

In this paper, we showed the efficiency of segmenting words into sub-words for recognition by an offline Farsi recognition system. Some proper statistical features were fed into RNN classifiers. Simple binary-output classifiers were used to reduce the complexity and time in both the training and testing phases. When the sub-words are inherently written separately, sub-word extraction is a good way to segment Farsi words. This method also helps other systems reduce their lexicon size.

Arabic researchers can also use this system because of the similarity between languages. Experimental results show that our system is relatively accurate, and generates a small number of errors.

As future work, we are developing a holistic system based on the RNN classifiers used in this study. We will train these simple RNNs with a true/false output layer. Likewise, the number of RNN classifiers is equal to the class count; however, each classifier is responsible for a word, not a sub-word. Using the sub-word extraction method, we only use the number of subwords to reduce the lexicon for each test word. A good idea will be utilizing a combination strategy combining the current segmentation-based approach with a future holistic approach.

References

[1] L.M. Lorigo and V. Govindaraju, "Offline Arabic Handwriting Recognition: A Survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 5, May 2006, pp. 712–724.

[2] M.T. Parvez and S.A. Mahmoud, "Offline Arabic Handwritten Text Recognition: A Survey," *ACM Comput. Surveys*, vol. 45, no. 2, Mar. 2013, pp. 23-1–23-35.

[3] O.H. Assma, O.O. Khalifa, and A. Hassan, "Handwritten Arabic Word Recognition: a Review of Common Approaches," *Int. Conf. Comput. Commun. Eng.*, Kuala Lumpur, Malaysia, May 13–15, 2008, pp. 801–805.

[4] K. Mostafa and A.M. Darwish, "Robust Base-Line Independent Algorithms for Segmentation and Reconstruction of Arabic Handwritten Cursive Script," *Proc. SPIE*, vol. 3651, 1999, pp. 73–83.

[5] T. Sari, L. Souici, and M. Sellami, "Off-Line Handwritten Arabic Character Segmentation Algorithm: ACSA," *Proc. Int. Workshop Frontiers Handwriting Recogn.*, Ontario, Canada, Aug, 6–8, 2002, pp. 452–457.

[6] L. Lorigo and V. Govindaraju, "Segmentation and Pre-recognition of Arabic Handwriting," *Int. Conf. Document Anal. Recogn.*, Seoul, Rep. of Korea, Aug. 3–Sept. 1, 2005, pp. 605–609.

[7] S.S. Maddouri, F. Ghazouani, and F.B. Samoud, "Text Lines and PAWs Segmentation of Handwritten Arabic Document by Two Hybrid Methods," *Int. Conf. Adv. Technol. Signal Image Process.*, Sousse, Tunisia, Mar. 17–19, 2014, pp. 310–315.

[8] M. Dehghan et al., "Handwritten Farsi (Arabic) Word Recognition: A Holistic Approach Using Discrete HMM," *Pattern Recogn.*, vol. 34, no. 5, May 2001, pp. 1057–1065.

[9] R.A.H. Mohamad, L. Likforman-Sulem, and C. Mokbel, "Combining Slanted-Frame Classifiers for Improved HMM-Based Arabic Handwriting Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 7, July 2009, pp. 165–1177.

[10] V. Margner, H. El-Abed, and M. Pechwitz, "Offline Handwritten Arabic Word Recognition Using HMM: A Character-Based Approach without Explicit Segmentation," *Proc. Colloque Int. Francophone sur l'Ecrit et le Document*, 2006.

[11] M.P. Lewis, *Ethnologue: Languages of the World; Sixteenth Edition*, Dallas, TX, USA: SIL International, 2009.

[12] A. Broumandnia, J. Shanbehzadeh, and M. Rezakhah Varnoosfaderani, "Persian/Arabic Handwritten Word Recognition Using M-Band Packet Wavelet Transform," *Image Vis. Comput.*, vol. 26, no. 6, June 2008, pp. 829–842.

[13] R. El-Hajj, L. Likforman-Sulem, and C. Mokbel, "Arabic Handwriting Recognition Using Baseline Dependent Features and Hidden Markov Modeling," *Int. Conf. Document Anal. Recogn.*, Seoul, Rep. of Korea, Aug. 29–Sept. 1, 2005, pp. 893–897.

[14] R. El-Hajj et al., "Combination of HMM-Based Classifiers for the Recognition of Arabic Handwritten Words," *Int. Conf. Document Anal. Recogn.*, Parana, Brazil, Sept. 23–26, 2007, pp. 959–963.

[15] L. Dinges et al., "Offline Automatic Segmentation Based Recognition of Handwritten Arabic Words," *Int. J. Signal Process., Image Process., Pattern Recogn.*, vol. 4, no. 1, 2011, pp. 131–144.

[16] A. Maqqor, A. Halli, and K. Satori, "A Multi-stream HMM Approach to Offline Handwritten Arabic Word Recognition," *Int. J. Natural Language Comput.*, vol. 2, no. 4, 2013, pp. 21–33.

[17] R. Gonzalez and R. Woods, *Digital Image Processing (3rd Edition)*, Upper Saddle River, NJ, USA: Prentice Hall, 2006.

[18] M.Y. Chen, A. Kundu, and S.N. Srihari, "Variable Duration Hidden Markov and Morphological Segmentation for Handwritten Word Recognition," *IEEE Trans. Image Process.*, vol. 4, no. 12, Dec. 1995, pp. 1675–1688.

[19] F. Farooq, V. Govindaraju, and M. Perrone, "Pre-processing Methods for Handwritten Arabic Documents," *Int. Conf. Document Anal. Recogn.*, Seoul, Rep. of Korea, Aug. 29–Sept. 1, 2005, pp. 267–271.

[20] Y.H. Tay et al., "An Offline Cursive Handwritten Word Recognition System," *Proc. IEEE Region Int. Conf. Electr. Electron. Technol.*, Singapore, Aug. 19–22, 2001, pp. 519–524.

[21] R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision*, Volume I, reading, MA, USA: Addison-Wesley, 1992, pp. 28–48.

[22] D. Xiang et al., "Offline Arabic Handwriting Recognition System Based On HMM," *IEEE Int. Conf. Sci. Inform. Technol.*, Chengdu, China, July 9–11, 2010, pp. 526–529.

[23] I. Siddiqi and N. Vincent. "Text Independent Writer Recognition Using Redundant Writing Patterns with Contour-Based Orientation and Curvature Features," *Pattern Recogn.*, vol. 43, no. 11, Nov. 2010, pp. 3853–3865.

[24] G.H. Kim and V. Govindaraju, "A Lexicon Driven Approach to Handwritten Word Recognition for Real-Time Applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 4, Apr. 1997, pp. 366–379.

[25] A. Graves et al., "Connectionist Temporal Classification: Labeling Unsegmented Sequence Data with Recurrent Neural Networks," *Proc. Int. Conf. Mach. Learning*, Pittsburgh, PA, USA, June 25–29, 2006, pp. 369–376.

[26] A. Graves et al., "A Novel Connectionist System for Unconstrained Handwriting Recognition," *IEEE Trans. Pattern Anal. Mach. Intell*., vol. 31, no. 5, May 2009, pp. 855–868.

[27] V. Frinken et al., "Long-Short Term Memory Neural Networks Language Modeling for Handwriting Recognition," *Int. Conf. Pattern Recogn.*, Tsukuba, Japan, Nov. 11–15, 2012, pp. 701–704.

[28] Y. LeCun et al., "Efficient BackProp," in *Neural Networks: Tricks of the trade*, Heidelberg, Netherland: Springer, 1998, pp. 9–50.

[29] M. Riedmiller and H. Braun, "A Direct Adaptive Method for Faster Backpropagation Learning: The Rprop Algorithm," *IEEE Int. Conf. Neural Netw.*, San Francisco, CA, USA, Mar. 28–Apr. 1, 1993, pp. 586–591.

[30] E. Bayesteh, A. Ahmadifard, and H. Khosravi, "A Lexicon Reduction Method Based on Clustering Word Images in Offline Farsi Handwritten Word Recognition Systems," *Iranian Conf. Mach. Vis. Image Process.*, Tehran, Iran, Nov. 16–17, 2011, pp. 1–5.

Mohammad Fazel Younessy Ghadikolaie received his BS degree from the Department of Electrical and Electronics Engineering, Mazandaran University, Iran in 2004, and his MS degree from the Department of Electrical Engineering, Babolsar, Iran University of Science and Technology, Tehron, Iran in 2007. His research interests include offline handwriting recognition using artificial intelligence.

Ehsanollah Kabir received his BS and MS degrees in electrical and electronics engineering from Tehran University, Iran in 1985. He received his PhD degrees in electronic systems engineering from Essex University, Colchester, UK in 1990. He is currently a professor in the Department of Electrical and Computer Engineering, Tarbiat Modarres University, Tehran, Iran. His current research interests include pattern recognition methods and their applications in document image analysis and handwriting recognition systems.

Farbod Razzazi received his BS and MS degrees in electrical engineering from Sharif University of Technology, Tehran, Iran in 1994 and 1996, respectively. He received his PhD degrees in electrical engineering from Amirkabir University of Technology (formerly, Tehran Polytechnic), Tehran, Iran in 2003. He is currently an assistant professor in the Department of Electrical Engineering, Islamic Azad University, Science and Research Branch, Tehran, Iran. His current research interests include pattern recognition methods and their applications in speech recognition systems.