Segmenting Handwritten Arabic Text

Ramzi A. Haraty and Alaa Hamid
Lebanese American University
P.O. Box 13-5053 Chouran
Beirut, Lebanon 1102 2801
Email: rharaty@lau.edu.lb

ABSTRACT

The segmentation and recognition of Arabic handwritten text has been an area of great interest in the past few years. However, a small number of research papers and reports have been published in this area. There are several major problems with Arabic handwritten text processing: Arabic is written cursively and many external objects are used such as dots, 'Hamza', 'Madda', and diacritic objects. In addition, Arabic characters have more than one shape according to their position inside a word. More than one character can also share the same horizontal space, creating vertically overlapping connected or disconnected blocks of characters. This makes the problem of segmentation of Arabic text into characters, and their classification even more difficult.

In this work a technique is presented that segments difficult handwritten Arabic text. A conventional algorithm is used for the initial segmentation of the text into connected blocks of characters. The algorithm then generates pre-segmentation points for these blocks. A neural network is subsequently used to verify the accuracy of these segmentation points. Another conventional algorithm uses the verified segmentation points and segments the connected blocks of characters.

Keywords: Artificial Neural Networks, and Optical Character Recognition.

1. Introduction

The segmentation and recognition of Arabic handwritten text has been an area of great interest in the past few years. However, a small number of research papers and reports have been published in this area due to the difficult problems associated with Arabic handwritten text processing [1][8][14]. There are several major problems with Arabic handwritten text processing: Arabic is written cursively and many external objects are used such as dots, 'Hamza', 'Madda', and diacritic objects. In addition, Arabic characters have more than one shape according to their position inside a word. More than one character can also share the same horizontal space, creating vertically overlapping connected or disconnected blocks of characters. This makes the problem of segmentation of Arabic text into characters, and their classification even more difficult. In addition, no substantial research has been found in the literature, which deals with segmenting and/or recognizing Arabic handwritten text.

The main problem with handwritten text processing is the presence of lines, non-character objects, and noise or 'salt-and-pepper' in the scanned image. Characters can also be written in many different sizes, writing instruments (varying thickness and stroke quality), and slants (causing character shearing along the horizontal axis).

Artificial Neural Networks (ANNs) have been successfully applied to many areas of pattern recognition, especially in the field of character recognition. Some researchers have used conventional methods for segmentation and recognition, while others have used ANN-based methods for the character recognition process [4][5][6][7].

This research describes a hybrid method to segment Arabic handwritten text. The method contains two main components. The first is a heuristic algorithm, which is responsible for scanning handwritten text, extracting blocks of connected characters, and then extracting features to be used in the second component. It is also responsible for generating pre-segmentation points, which are validated by the second component, the ANN. The ANN verifies whether all the segmentation points found are correct or incorrect.

This research was conducted primarily due to the challenging scientific nature of the problem and secondly its industrial importance. The latter arises from the numerous applications of handwritten recognition systems. Some of these include postal address recognition, reading machines for the blind, processing manually filled-out forms, bank-check recognition, and others. Automating the processing of handwritten text can result in significant cost savings in many fields.

The remainder of this paper is divided as follows. Section 2 presents the difficulties and obstacles found when processing handwritten text. Section 3 describes the proposed approach used to segment Arabic handwritten text. The design of the heuristic and neural network components is also presented. Section 4 presents the experimental results. Section 5 presents related work. And finally a conclusion is drawn in section 6.

2. Character segmentation obstacles

There are several major problems associated with processing handwritten Arabic text. They can be classified into three main categories: general difficulties, handwritten text specific difficulties, and Arabic text specific difficulties.

2.1 General difficulties

The following difficulties are common among character segmentation and recognition methods in general:

Problem 1. Presence of lines and other non-character objects.
Problem 2. Presence of noise or 'salt-and-pepper' in the scanned image.
Problem 3. Linguistic problems, i.e., if a dictionary is used as a spelling checker to improve the accuracy of the recognition process, then proper names, acronyms, or other words that are not likely to be in the lexicon will decrease recognition accuracy.

2.2 Handwritten text specific difficulties

The following difficulties are specific to handwritten text segmentation and recognition methods:

Problem 1. Variety in character size, i.e., characters may be written in many different sizes without changing their meaning.
Problem 2. Variety in writing instrument, i.e., characters may vary in line thickness, color, and/or stroke quality. Thick stroke causes the following problems:

i. Touching characters.
ii. Holes in letters, e.g., ق or ف letters get filled up partially or completely.
iii. Thin stroke or low contrast may result in broken characters. Gaps in the stroke may also cause a lot of errors.

Problem 3. Different writers and the same writer under different conditions will slant their letters differently, i.e., their handwriting undergoes a shear along the horizontal axis.
Problem 4. Translation problems, i.e., characters are not always written in the same position relative to the enclosing borders of the scanned image.
Problem 5. Introduced and dropped spaces between characters make the task of word separation more difficult.
Problem 6. Handwritten text consists of elongated strokes making the separation of lines a difficult task when they overlap, as shown in Figure 1.

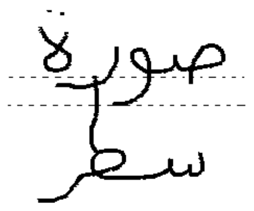


Figure 1. Two lines occupying a shared vertical space.

2.3 Arabic text specific difficulties

The following difficulties are specific to Arabic text processing:

Problem 1. Arabic is written cursively, i.e., more than one character can be written connected to each other, forming a block of characters (BC).
Problem 2. Arabic uses many types of external objects, such as dots, 'Hamza', 'Madda', and diacritic objects. A 'Hamza' is shown in Figure 2. These make the task of line separation and text segmentation into BCs and characters more difficult.

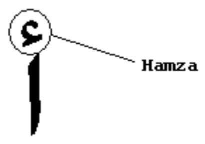


Figure 2. A Hamza external object.

Problem 3. Arabic characters can have more than one shape according to their position inside a BC: beginning, middle, end, or standalone, as shown in Figure 3.

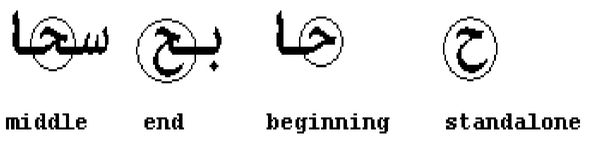


Figure 3. The shapes the character ح takes according to its position inside a word.

Problem 4. Different writers and the same writer under different conditions will write some Arabic characters in completely different ways, as shown in Figure 4.

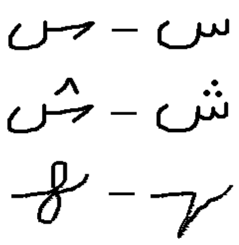


Figure 4. Three characters written in completely different ways.

Problem 5. The letter س also complicates segmentation and recognition when it occurs in the middle of a word as shown in Figure 5 .

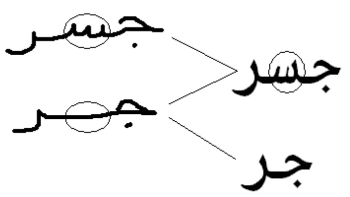


Figure 5. The letter سـmay be missed when appearing in the middle of a word.

Problem 6. Other characters have very similar contours and are difficult to segment and recognize especially when non-character and external objects are present in the scanned image. Figure 6 shows a list of such characters.

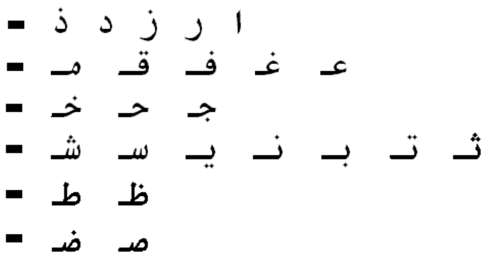


Figure 6. Characters with similar contours.

Problem 7. Characters that do not touch each other but occupy a shared horizontal space increase the difficulty of BC segmentation, as illustrated in Figure 7.

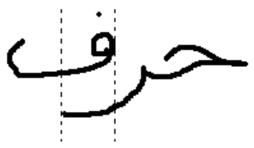


Figure 7. Two characters occupying a shared horizontal space.

Problem 8. Arabic uses many ligatures especially in handwritten text. Ligatures are connected characters that occupy a shared area, as shown in Figure 8. Segmenting ligatures into their constituent characters is very difficult because in most cases no vertical or horizontal line exists to segment a ligature.

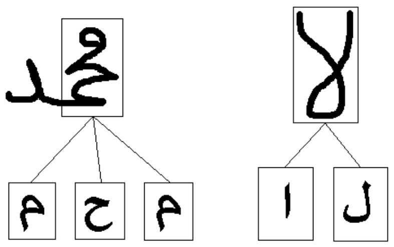


Figure 8. Arabic ligatures and their constituent characters.

3. Proposed techniques

There are a number of steps that need to be taken before handwritten text can be recognized by a computer. These include sample data collection and analysis, binarization, extracting connected BCs, skeletonization, feature extraction, segmentation, and character recognition.

3.1 The data set

Since there is no standard benchmark database for Arabic handwritten text, samples were acquired randomly from various students and faculty members around the university. They were asked to write down their own mailing address on A4 sized paper. These addresses were then scanned using an Agfa SnapScan 1212p scanner at 150 pixels per inch and saved in monochrome Windows Bitmap (BMP) format. The images had different sizes ranging from 260 x 140 pixels to 1200 x 400 pixels. A sample address is shown in Figure 9.

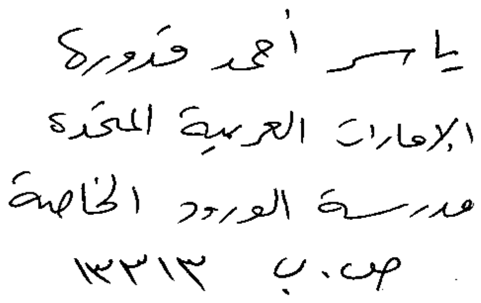


Figure 9. A sample handwritten address.

3.2 Binarization

After the images were acquired, they were converted into monochrome bitmap form. Before any segmentation or processing could take place, it was then necessary to convert the images into binary representations. A heuristic algorithm generated a matrix of ones (1's) and zeros (0's) for each image. Each black pixel was represented with a 1 and each white pixel with a 0. In this form, segmentation and preprocessing could take place more easily.

3.3 Extracting connected BCs

Segmentation deals with the process of attempting to isolate classifiable units from the handwritten text image. It plays an important role in the overall process of character recognition. It has been argued that the overall success rate of any recognition system can be expressed as a product of two factors: the success rate of the segmentor and the success rate of the recognizer [1][11].

BC extraction is the first step of the segmentation phase. The use of dots, diacritics, and other external objects made the task of linking them to main character objects, to create BCs, a very difficult process. A recursive heuristic algorithm was implemented with a 94 % accuracy, scanned the whole binary matrix of the image and performed the following steps:

Step 1. Before any processing could occur, invalid isolated black pixels, or '*pepper*' were removed. A black pixel was identified as pepper and discarded if it had a maximum of one black neighbor pixel.
Step 2. Recursively, identify each group of connected black pixels as an object.
Step 3. Classify objects as child or parent ones. If the weight, or black pixel density, of the object is less than half of average weight of all objects, then it is marked as a child. Otherwise, it is marked as a parent.
Step 4. For each child object, determine the distance to all parents.
Step 5. For each child object, determine the distance range, *R,* for all possible parents, which is equal to 150% of the distance to the nearest parent. This formula was found by experimentation and yielded the best results.
Step 6. Merge all children to all parents who are at a maximum distance of *R*. This will lead to child objects being duplicated and merged to more than one parent. This is done to solve the problem of children that are located near objects other than their parent ones.

3.4 Skeletonization

Skeletonization, or thinning, is an image-processing step that reduces BCs to their skeletons, i.e., transforming characters into arc segments one pixel thick. A skeletonization algorithm must not alter the shape of a BC. This includes the preservation of connected components and the number of cavities and holes. The skeletonization process is required in order to extract certain features like corner points, end points, and fork points.

An adaptation of Rosenfeld's skeletonization algorithm [12] produced acceptable results with few enhancements and modifications. An important feature of this algorithm is that it preserves the connectivity feature of BCs. Before describing the algorithm, it is necessary to define four properties of black pixels.

3.4.1 Black pixels properties

A black pixel is said to be *eight-simple* if changing it from black to white (1 to 0) does not alter the connectivity property of the remaining black pixels of the BC, as shown in Figure 10. In other words, it does not disconnect the BC if it was discarded. The most difficult task of the thinning algorithm is the decision of whether a point has this property or not.

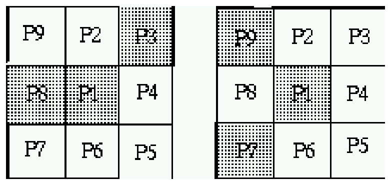


Figure 10. Two examples where P1 is not an eight-simple point.

A black pixel is said to be *eight-isolated* if it has no neighboring black pixels. An *eight-endpoint* is a black pixel that has exactly one black neighbor pixel, as shown in Figure 11. Eight-endpoints should not be discarded because if we went on deleting endpoints the whole BC would shrink to only one pixel.

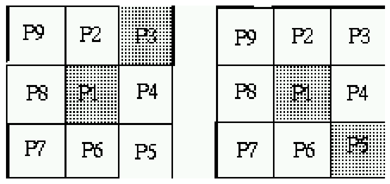


Figure 11. Two examples where P1 is an eight-endpoint.

A black pixel is said to be an *east border point* if its east neighbor is a white pixel. Similarly, west, north, and south border points can be defined. Four examples of border points are shown in Figure 12.

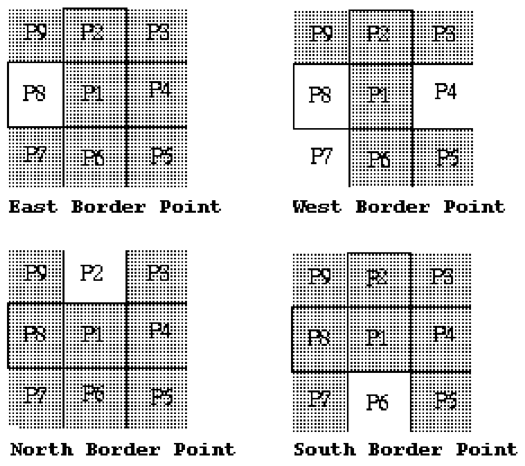


Figure 12. Four examples where P1 is a border point.

3.4.2 Skeletonization algorithm

A thinning iteration is divided into four sub-iterations. Thinning iterations should be repeated until they discard no more black pixels. Each step in these iterations is a thinning sub-iteration in one direction. Every sub-iteration deletes only points of one direction type; the others remain, irrespective of their simplicity. This way we can make sure that pixels are removed symmetrically from the borders. Every thinning iteration removes a layer until finally only the skeleton remains. The four sub-iterations are as follows:

Sub-iteration 1. Discard all east border points that are eight-simple but are neither eight-isolated nor eight-endpoint.
Sub-iteration 2. Discard all west border points that are eight-simple but are neither eight-isolated nor eight-endpoint.
Sub-iteration 3. Discard all north border points that are eight-simple but are neither eight-isolated nor eight-endpoint.
Sub-iteration 4. Discard all south border points that are eight-simple but are neither eight-isolated nor eight-endpoint.

Note that the order of the sub-iterations is important: east, west, north, and then south. This order produced the best results. Points are examined one by one and discarded if possible. A BC skeletonized by this algorithm is illustrated in Figure 13.

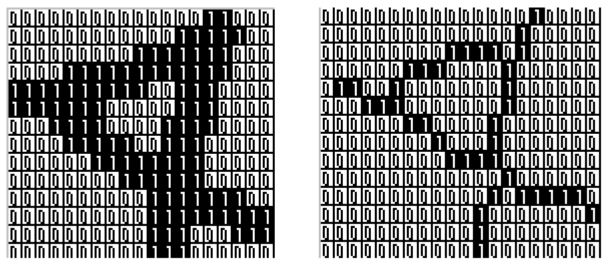


Figure 13. An example of a skeletonized BC.

3.5 Feature extraction

Scanned images often contain too much non-essential information, so a process called feature extraction is used to get information suitable for use in segmentation and recognition. Feature extraction lowers the number of features in order to lower the dimensionality of the input and thus the computational complexity of the system.

Each feature is represented by a number of attributes. These attributes quantify the nature of the feature, by, for example, specifying its position or size. An attribute may be represented as a continuous value, a discrete value, or a binary value.

The features that are extracted from the raw data image are one aspect of its representation. A second aspect is the relationships that exist between features. In spatial terms we can think of simple binary relationships such as 'above' or 'close-to' that describes the relative position of two features. This information is also important in segmentation and recognition systems.

The following is a description of the most important extracted features. Holes are islands of white pixels completely surrounded by black pixels. A hole is shown in Figure 13. Holes are the most difficult feature to extract. A recursive method was used which scanned all white pixels of an image and for each pixel searched for borders in all directions. If the pixel was completely surrounded by black pixels and/or image edges then all traversed white pixels were marked as hole pixels.

A corner point is a black or white skeleton point that has at least two connected branches on right angles to each other, as shown in Figure 14.

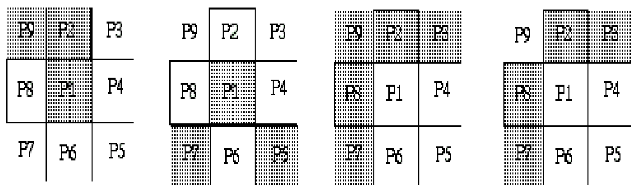


Figure 14. Four examples of corner points.

A fork point is a black skeleton point that has at least three connected branches as shown in Figure 15.

In order to decide whether a column is a valid segmentation point or not, the features listed in Table **1** were extracted for each column of an image.

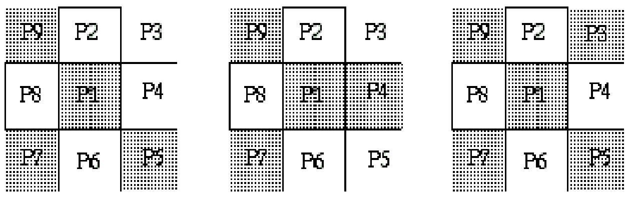


Figure 15. Three examples where P1 is a fork point.

Table 1. Major features extracted for each column of BC matrix.

Feature	**Attributes**	**Description**	**Att. Type**	**Att. Value**
Image width and height		Image width and height in pixels.	Discrete	(0,∞)
Black pixel density	Black pixel density / height	Number of black pixels in the column divided by the image height.	Cont.	[0,1]
	Density minima	Does the column cross a density minimum?	Binary	0 or 1
	Density maxima	Does the column cross a density maximum?	Binary	0 or 1
Transitions	Number of transitions crossed	Scan the image vertically and count the number of foreground-background and background-foreground transitions crossed by column.	Discrete	[0,height)
Holes	Number of holes crossed	Count the number of holes (or islands of white pixels completely surrounded by black pixels) crossed by column.	Discrete	[0,height)
	Total hole densities / height	Total number of hole pixels crossed by column divided by the image height.	Cont.	[0,1]
Endpoints	Number of endpoints crossed	Number of endpoints crossed by column.	Discrete	[0,height]
Corner points	Number of corners crossed	Number of corner points crossed by column.	Discrete	[0,height]
Fork points	Number of fork points crossed	Number of fork points crossed by column.	Discrete	[0,height]
Relative index of column in image		Index of column divided by image width.	Cont.	[0,1]
Upper and lower contours	Upper and lower contour index / height	Index of upper most and lower most black pixel crossed by column divided by image height.	Cont.	[0,1]
	Upper and lower contour minima or maxima	Does the column cross an upper or lower contour minima or maxima?	Binary	0 or 1
Feature relationships	Index of nearest left and right feature / 100	Index of nearest left and right feature in a 100-pixel width range.	Cont.	[0,1]

3.6 Pre-segmentation point generation

The objective of this module is to over-segment all the BCs based on the features extracted for each column. A simple heuristic segmentation algorithm was implemented which first scans the BC looking for minimas and maximas in black pixel densities, holes, endpoints, corner points, fork points, and upper and lower contours common in handwritten cursive text. Based on these features, the algorithm decides on pre-segmentation points.

Finally, the algorithm performs a final check to see if one pre-segmentation point was not too close to another by comparing the distance between these points with the average character width. The average character width was determined from the average BC height. Assuming that the width of a character is, in most cases, less than its height, an approximation of character width was estimated as a percentage of the average BC height.

The final result is a set of pre-segmentation points that are going to be validated by the ANN.

3.7 Pre-segmentation point validation by ANN

The objective of the segmentation ANN is to validate the accuracy of the pre-segmentation points generated by the heuristic algorithm presented in the previous section. To train the ANN with both accurate and erroneous segmentation points, the output from the heuristic segmentation algorithm was manually separated into valid and invalid points and saved to a file together with the extracted set of features and desired output for each point.

3.7.1 ANN architecture

A generalized feed forward neural network was used to validate the accuracy of the proposed segmentation points. This neural network is a generalization of the multi-layer perceptron (MLP) such that each layer feeds forward to all subsequent layers. In theory, a MLP can solve any problem that a generalized feed forward network can solve. In practice, however, generalized feed forward networks often solve the problem much more efficiently [2][13].

The first criterion is to use an efficient technique for training. Training of the network was done using the error back-propagation technique. This technique gets its name from the fact that the network is presented with an input pattern, for which an output pattern is calculated. Then the error between the desired and actual output can be determined, and passed backwards through the network. Based on these errors, weight adaptations are calculated, and errors are passed to a previous layer, continuing until the first layer is reached. The error is thus propagated back through the network. A training set of 48,000 exemplars was used.

The second criterion is the network size. The number of processing elements (PEs) in a hidden layer is associated with the mapping ability of the network. The larger the number, the more powerful the network is. However, if one continues to increase the network size, there is a point where the generalization gets worse. This is due to the fact that we may be over-fitting the training set, so when the network works with patterns that it has never seen before the response is unpredictable.

The third criterion is the termination of the training process. Cross validation was used, which is a highly recommended method for stopping network training. It monitors the mean square error on an independent set of data and stops training when this error begins to increase. This is considered to be the point of best generalization. The cross-validation set consisted of 10,000 exemplars.

There is no rule of thumb to determine good network architecture just from the number of inputs and outputs. It depends critically on the number of training cases, the amount of noise, and the complexity of the function or classification the network is supposed to learn. There are cases with one input and one output that require thousands of hidden units, and cases with a thousand inputs and a thousand outputs that require only one hidden unit, or none at all. To solve these issues many different networks with different number of hidden units were tried at first, starting with the smallest possible number of PEs. The generalization error for each one was estimated, and the network with the minimum estimated generalization error that learned best to identify correct segmentation points was chosen.

The best ANN architecture reached consisted of 52 inputs, 1 output, and 4 hidden layers. The 52 inputs were feature attributes of a pre-segmentation point and the output was the validity of the point. The ANN architecture is summarized in Table 2.

Table 2. Architecture of segmentation ANN.

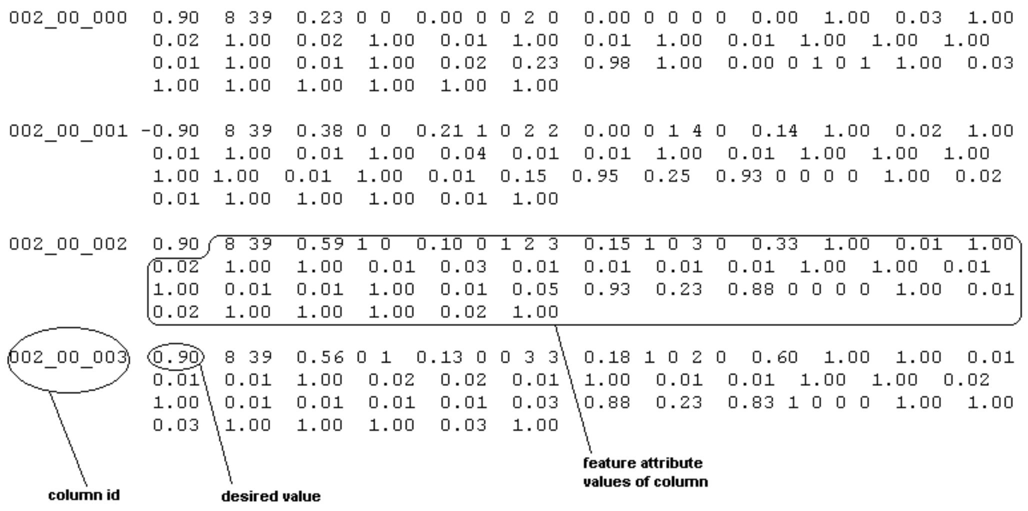
	Processing Elements	Transfer function	Learning Rule	Step Size	Momentum
Layer 1	41	Tanh	Momentum	1.0	0.7
Layer 2	27	Tanh	Momentum	0.1	0.7
Layer 3	20	Tanh	Momentum	0.01	0.7
Layer 4	16	Tanh	Momentum	0.001	0.7
Output Layer	1	Tanh	Momentum	0.0001	0.7

3.7.2 ANN input design

68,000 pre-segmentation points were evaluated manually and divided into three parts as shown in Table 3.

Table 3. Manually evaluated input sets points.

Input Set	Number of exemplars
Training set	48,000
Cross-validation set	10,000
Testing set	10,000



```
002_00_000  0.90  8 39  0.23 0 0  0.00 0 0 2 0  0.00 0 0 0 0  0.00  1.00  0.03  1.00
                  0.02  1.00  0.02  1.00  0.01  1.00  0.01  1.00  0.01  1.00  1.00  1.00
                  0.01  1.00  0.01  1.00  0.02  0.23  0.98  1.00  0.00 0 1 0 1  1.00  0.03
                  1.00  1.00  1.00  1.00  1.00  1.00

002_00_001 -0.90  8 39  0.38 0 0  0.21 1 0 2 2  0.00 0 1 4 0  0.14  1.00  0.02  1.00
                  0.01  1.00  0.01  1.00  0.04  0.01  0.01  1.00  0.01  1.00  1.00  1.00
                  1.00 1.00  0.01  1.00  0.01  0.15  0.95  0.25  0.93 0 0 0 0  1.00  0.02
                  0.01  1.00  1.00  1.00  0.01  1.00

002_00_002  0.90  8 39  0.59 1 0  0.10 0 1 2 3  0.15 1 0 3 0  0.33  1.00  0.01  1.00
                  0.02  1.00  1.00  0.01  0.03  0.01  0.01  0.01  0.01  1.00  1.00  0.01
                  1.00  0.01  0.01  1.00  0.01  0.05  0.93  0.23  0.88 0 0 0 0  1.00  0.01
                  0.02  1.00  1.00  1.00  0.02  1.00

002_00_003  0.90  8 39  0.56 0 1  0.13 0 0 3 3  0.18 1 0 2 0  0.60  1.00  1.00  0.01
                  0.01  0.01  1.00  0.02  0.02  0.01  1.00  0.01  0.01  1.00  1.00  0.02
                  1.00  0.01  0.01  0.01  0.01  0.03  0.88  0.23  0.83 1 0 0 0  1.00  1.00
                  0.03  1.00  1.00  1.00  0.03  1.00
```

Figure 16. Sample ANN input file.

Each pre-segmentation point is represented by a row in the input files. Each row starts with the identifier of the column, its desired value, and then the feature attributes are listed. A sample input file is shown in Figure 16.

3.7.3 ANN implementation

The design of the segmentation ANN described was implemented using NeuroSolutions, version 4.17 by NeuroDimensions, Inc. The implemented ANN is shown in Figure 17

Each of the axons shown represents a layer, or vector, of PEs. All axons are equipped with a summing junction at their input and a splitting node at their output. This allows axons to accumulate input from, and provide output to, an arbitrary number of components.

Axons sum up all of their inputs and then apply a function to that sum. The applied function may be either linear or nonlinear. The input axon, for example, simply applies an identity (linear) map between its input and output activity. All hidden and output axons apply the hyperbolic tangent map between its input and output. The Tanh axon used in these layers also applies a bias to each neuron in the layer. This will squash the range of each neuron in the layer to between -1 and 1.

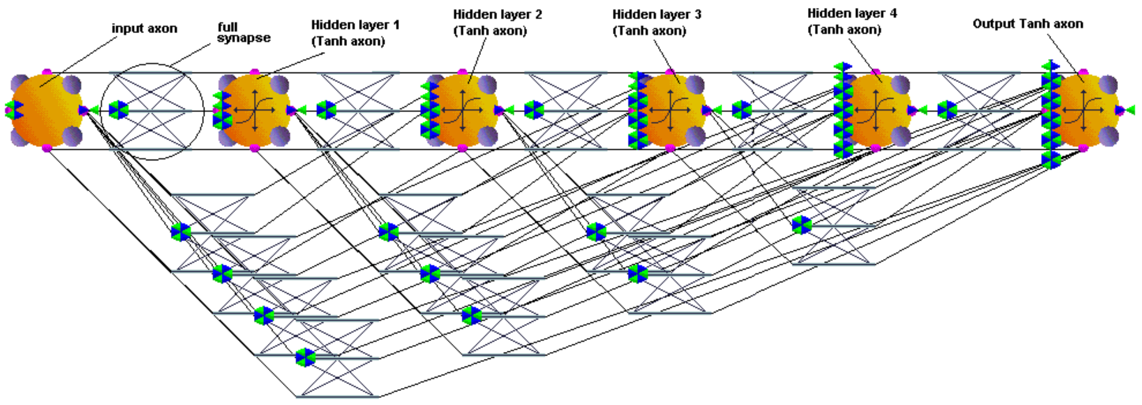

input axon
full synapse
Hidden layer 1 (Tanh axon)
Hidden layer 2 (Tanh axon)
Hidden layer 3 (Tanh axon)
Hidden layer 4 (Tanh axon)
Output Tanh axon

Figure 17. Segmentation ANN used to validate pre-segmentation points. (Source: NeuroSolutions Software)

4. Experimental Results

The output range of the ANN was between –0.9 and +0.9. A positive value indicated that a point is a valid segmentation point; a negative value indicated that a point should be ignored.

A heuristic algorithm checked the results of the segmentation ANN on a test set of 10,000 exemplars. The algorithm defined the segmentation of a BC as correct when each known segmentation point was covered by an approved segmentation point by the ANN. Adequate coverage of a segmentation point is achieved when the distance from the known segmentation point to the closest approved point is less than 15% of the average character size.

There were some objects that were rather difficult to segment in handwritten Arabic text. Every 100 BCs of the collected data, contain 10.16 un-segmentable ligatures and 13.02 characters with miss-located external objects. In addition, 9.24 س and ش characters occur in every 100 BCs, which are almost always un-segmentable. Other miscellaneous un-segmentable BCs include characters like the letter ض, which is always segmented into the letters ع or م, and ن.

Table 4 shows the results of the segmentation ANNs trained on the 48,000 training exemplars and tested on 10,000 exemplars. Many experiments were performed varying settings such as the network type, the number of hidden layers and the number of processing elements in each layer. For each experiment the number of inputs remained the same: 52 input features for each column.

Table 4. Segmentation ANN results using 48,000 training exemplars, tested on 10,000 exemplars.

ANN Architecture			MSE	Correct Points		Incorrect Points	
Network Type	Hidden Layers No	Hidden Layers PEs		Invalid points	Valid points	Invalid points marked as valid	Valid points marked as invalid
Feed forward MLP	2	41-27	0.81	1354 (13.54%)		8646 (86.46%)	
Feed forward MLP	3	41-27-20	0.72	2684 (26.84%		7316 (73.16%)	
Feed forward MLP	4	41-27-20-16	0.41	5311 (53.11%)		4689 (46.89%)	
				3767 (37.67%)	1544 (15.44%)	3326 (33.26%)	1363 (13.63%)
Feed forward MLP	5	41-27-20-16-13	0.56	4225 (42.25%)		5775 (57.75%)	
Feed forward MLP	6	41-27-20-16-13-11	0.50	4633 (46.33%)		5367 (53.67%)	
MLP	5	55-31-19-15-11	0.82	1332 (13.32%)		8668 (86.68%)	
MLP	7	55-31-19-15-13-11-9	0.59	3854 (38.54%)		6146 (61.46%)	
MLP	9	55-31-19-15-13-11-9-7-6	0.73	2336 (23.36%)		7664 (76.64%)	

The highlighted ANN architecture in 4 performed best in identifying correct segmentation points and discarding incorrect ones. The minimum MSE achieved was 0.41. The ANN was able to identify the accuracy of 5,311 points out of the 10,000-point testing set. Of the correctly identified points, 3,767 were invalid segmentation points and 1,544 were valid segmentation points.

The ANN incorrectly identified 4,689 points. 3,326 of these points were invalid segmentation points marked as valid, and 1,363 were valid points marked as invalid. It should be noted that the majority of incorrectly identified points were invalid segmentation points marked as valid, which implies that the ANN over-segmented the handwritten BCs.

After studying the distribution of the ANN results over the range [-0.9,+0.9], it was noted that the majority of these 3,326 points were found in the [0,+0.5] range, as illustrated in Figure 18.

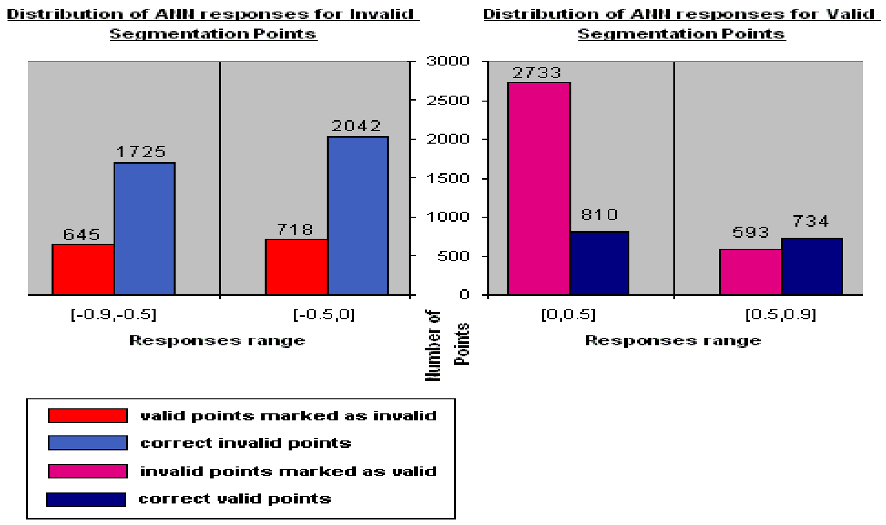


Figure 18. Distribution of ANN responses.

To decrease the number of incorrectly identified segmentation points, a threshold value was applied to the ANN output. A module was implemented which checked the ANN responses against a 0.5 threshold. Responses between 0 and +0.5 were therefore rejected. It should be noted that these rejected patterns also include 810 correctly identified valid segmentation points. However, the number of incorrectly identified points, 2,733, in the (0,0.5) range is much greater than the correctly identified points. Table 5 shows the results after rejecting these patterns.

Table 5. ANN results after rejecting patterns with responses in the (0,0.5) range.

Correct Points		**Incorrect Points**		**Rejected Points**	
Invalid points	**Valid points**	**Invalid points marked as valid**	**Valid points marked as invalid**	**Invalid points marked as valid**	**Valid points**
4501 (45.01%)		1956 (19.56%)		3543 (35.43%)	
3767 (37.67%)	734 (7.34%)	593 (5.93%)	1363 (13.63%)	2733 (27.33%)	810 (8.10%)

As shown in Table 5, rejecting patterns with responses in other ranges is not efficient because the number of correctly identified points is greater than the number of incorrectly identified points.

5. Related Work

The history of handwriting recognition systems is not complete without mentioning the optical character recognition (OCR) systems that preceded them. OCR is a problem recognized as being as old as the computer itself. There have been many papers and technical reports published reviewing the history of OCR technologies. Modern OCR was said to have begun in 1951 due to an invention by M. Sheppard called GISMO, a robot reader-writer. In 1954, a prototype machine developed by J. Rainbow was used to read uppercase typewritten letters at very slow speeds. By 1967, companies such as IBM finally marketed OCR systems. However in the late 60's, these systems were still very expensive, and therefore could only be used by large companies and government agencies. Today, OCR systems are less expensive and can recognize more fonts than ever before [15].

A number of systems have been developed to recognize Arabic text. One of these is TextPert 3.7 Arabic, produced by CTA Inc., which runs on the Macintosh Arabic system. Another is Al-Qari'al-Ali, a version of the program known as MULTREC, produced by Alamiah Software Co. Both of these programs were able to recognize certain computer printed texts of good quality with a reasonable degree of accuracy considering the difficulties of the Arabic text [3].

Another system designed by Fehri and Ben Ahmed used a hybrid of Radial Basis Function Networks and Hidden Markov Models to recognize printed Arabic text after identifying the used font. The results showed an increase in the recognition rate when the font is known prior to segmentation process [10].

Finally, the best-known Arabic text recognition system ever developed is Sakhr OCR developed by Sakhr. The system uses an artificial neural network with a segmentation accuracy of 98% and a recognition accuracy of 99.8% for printed text [16].

6. Conclusion and Further Work

A heuristic segmentation technique used in conjunction with a generalized feed-forward multi-layer neural network has been presented in this paper. It was used to segment difficult handwritten Arabic text, producing promising results. With some modifications more testing shall be conducted to allow the technique to be used as part of a larger system.

The segmentation program over-segmented the BCs it was presented with. This allowed the segmentation ANN to discard improper segmentation points and leave accurate ones. Overall the whole process was very successful, however some limitations still exist. The segmentation phase proved to be successful in vertical segmentation of connected blocks of characters. However, in Arabic handwritten text, a lot of characters share the same horizontal space. A limitation of the presented technique is that it could not accurately segment horizontally these overlapping characters.

Another problem is that there are a lot of handwritten characters that can be segmented and classified into two or more different classes depending on whether you look at them separately, or in a word, or even in a sentence. In other words, character segmentation and classification, especially handwritten Arabic characters, depends largely on contextual information, and not only on the topographic features extracted from these characters. Arabic handwriting recognition is a difficult problem and it is not yet realistic to expect systems to achieve an acceptable accuracy in large vocabularies or where contextual information is of little use.

In future work, the segmentation technique will be improved in a number of ways. The heuristic component of the segmentation system will be enhanced further. Originally, one of the main aims of the heuristic algorithm was to keep the number of incorrect segmentation points to a minimum, so that errors and processing time could be reduced. As a result, under-segmentation was noticeable in some BCs. Looking for more features or possibly enhancing the current feature extraction methods can solve this problem.

References

[1] B. Al-Badr and R. Haralick, "A Segmentation-Free Approach to Text Recognition with Application to Arabic Text", *International Journal on Document Analysis and Recognition*, vol. 1, number 3, 147-166, December 1998.

[2] L. Almeida, "*Multilayer Perceptrons - Handbook of Neural Computation",* IOP Publishing Ltd and Oxford University Press, 1997.

[3] J. Bell and P. Zemanec, "Test of Two Arabic OCR Programs", Distributed on *Reader* 14 Jan 1995 and *Itisalat* 17 Jan 1995.

[4] M. Blumenstein and B. Verma, "A Segmentation Algorithm used in Conjunction with Artificial Neural Networks for the Recognition of Real-World Postal Addresses", *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA'97)*, Gold Coast, Australia, pp. 155-160, 1997.

[5] M. Blumenstein and B. Verma, "Neural Based Solution for the Segmentation and Recognition of Difficult Handwritten Words", *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, 1998.

[6] T. Breuel, "*Handwritten Character Recognition Using Neural Networks - Handbook of Neural Computation",* IOP Publishing Lomited and Oxford University Press, 1997.

[7] B. Eastwood, A. Jennings, and A. Harvey, "A Feature Based Neural Network Segmenter for Handwritten Words", *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA '97)*, Gold Coast, Australia, pp. 286-290, 1997.

[8] M. Fehri and M. Ben Ahmed, "Detection and Correction of Misspelled Isolated Words in Arabic Language", *Proceedings of the Fourth International Conference and Exhibition one Multilingual Computing (ICEMCO)*, London, April 1994.

[9] M. Fehri, and M. Ben Ahmad, "A Hybrid RBF/HMM Approach for Recognizing Multifont Arabic", *Laboratoire Riadi-Ensi*, Tunisia. 2001.

[10] K. Han, I. K. Sethi, "Off-line Cursive Handwriting Segmentation", *Proc. ICDAR '95*, Montreal, Canada*,* pp. 894-897, 1995.

[11] S-W. Lee, D-J. Lee, H-S. Park, "A New Methodology for Gray-Scale Character Segmentation and Recognition", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, pp. 1045-1051, 1996.

[12] A. Rosenfeld, "A *Simple Parallel Algorithm for Skeletonization*", http://www.cs.mcgill.ca/~laleh/rosen_alg.html. 2001.

[13] S. N. Srihari, "Recognition of Handwritten and Machine-printed Text for Postal Address Interpretation", *Pattern Recognition Letters,* pp. 291-302, 1993.

[14] R. Srihari, "Use of Lexical and Syntactic Constraints in Recognizing Handwritten Sentences", *Proceedings of the ARPA workshop on Human Language Technology*, Princeton, NJ, pp. 403-407, March 1994.

[15] B. Verma, M. Blumenstein, and S. Kulkarni, "Recent Achievements in Off-Line Handwriting Recognition Systems", *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA '98)*, Melbourne, Australia. Pp. 27-33, 1998.

[16] www.sakhrsoft.com. 2002.

Biographies

Ramzi A. Haraty is an Assistant Professor of Computer Science at the Lebanese American University in Beirut, Lebanon. He received his B.S. and M.S. degrees in Computer Science from Minnesota State University - Mankato, Minnesota, and his Ph.D. in Computer Science from North Dakota State University - Fargo, North Dakota. His research interests include database management systems, artificial intelligence, and multilevel secure systems engineering. He has well over 50 journal and conference paper publications. He is a member of Association of Computing Machinery, Arab Computer Society and International Society for Computers and Their Applications.

Alaa Hamid received his M.S. degree in Computer Science form the Lebanese American University in Beirut, Lebanon. His research interests include database management systems, neural networks, and software engineering.