ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/255656706

A New Algorithm for Arabic Optical Character Recognition

Article *in* WSEAS Transactions on Information Science and Applications · April 2006

CITATIONS
11

READS
451

4 authors, including:

Omar Al-Jarrah
Jordan University of Science and Technology
69 PUBLICATIONS **955** CITATIONS
SEE PROFILE

Samer Al-Kiswany
University of Waterloo
37 PUBLICATIONS **443** CITATIONS
SEE PROFILE

Mohammad Fraiwan
Jordan University of Science and Technology
28 PUBLICATIONS **153** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Pattern Recognition View project

Project artificial intelligence and image processing View project

All content following this page was uploaded by Omar Al-Jarrah on 25 July 2014.

The user has requested enhancement of the downloaded file.

A New Algorithm for Arabic Optical Character Recognition

amed Fraiwan Hani Khasawneh
B
Department of Computer Engineering Samer Al-Kiswany
Omar Al-Jarrah
Jordan University of Science and Technology
P.O. Box 3030
Irbid 2211
JORDAN

r we present a new approach for designing an Optical Character Recognition (OCR) bet. Our approach addresses mere text images; using customized techniques. We have hand into three phases: preprocessing and line extraction, segmentation, and character histogram and thresh-holding were considered in line extraction, image filtering and line algorithm was proposed for segmentation, and a neural network was used for he segmentation algorithm is based on a new and novel idea that exploits the nature of ntroduces the concept of the main line to tokenize the text, and it generates a set of 33 resent the 28 Arabic characters and their different shapes and variation. The system e text and compares them with the reference tokens. Experimental results illustrate the em using different font types and sizes with an average recognition rate of 87%.

Character Recognition, Arabic Alphabet, Image Processing, Character Recognition, etworks Classifier.

ition systems are key stones ithout the advances in OCR e set of applications are e applications of the OCR nd-alone applications to an more complex systems. An the scanner related software e from a scanned page, thus e amount of time that were ing. An example of the later und in PDAs. In addition, te the design of the check is used in banks to fasten and it is used by highway tes.

be broadly categorized into nd off-line OCR systems. In ognition is performed at the s the case in PDAs. While, h scanned images. Online provided with assistance and that guides their operation, d the ending points of the character. Furthermore, online systems usually facilitate writing separate characters, not complete words.

A number of researchers attempted to provide efficient OCR solutions. Despite the attractive performance found in online OCR system [4][13], the adopted approaches cannot be directly used in offline OCR systems due to the dependency of such systems on the online nature of writing. However, a few number of researchers addressed offline OCR systems for Arabic language. The Arabic language poses a unique OCR challenges (section 3.1 addresses Arabic language properties in details), making most of the techniques proposed for English or other languages inefficient. Hence, a number of new approaches attempted to present efficient OCR solutions for Arabic language.

A remarkable approach was adopted in the Word Level Arabic Recognition WLAR OCR system [1]. In WLAR OCR, instead of trying to recognize every character separately, the word level recognition was proposed. The WLAR computes a vector of image-morphological features on the input word image. This vector is matched next against a precompiled database of vectors from lexicon of Arabic words.

Vectors from the database with the highest match score are returned as hypotheses for the unknown image.

Despite the effectiveness of the WLAR in avoiding single-character errors when recognized separately, it achieved overall inadequate level of accuracy. Since the WLAR depended on extracting a set of features which depended on the location and size of the feature, so differences in stroke width, height or presence of noise can make difference between the presence and absence of a feature. This becomes especially true when the input words differ in only one character, or even one dot (e.g. recognizing (زوج) and (روج)). The authors stated that increasing the recognition performance beyond 65% proved to be difficult.

Another deficiency imposed by the word level recognition, is the introduction of the new phrases and words into the language, which is common in scientific writing. The WLAR system will not be able to recognize them correctly. Also, since the word level recognition depended on a database of word's features, the system will be totally dependant on the language, and it will need an expensive training to be used in another language. For example the Arabic and the Farsi written languages use the same characters, but their dictionaries are totally different.

In the Arabic Text Recognition System proposed in [2], the authors introduced the over-segmentation approach. In their system, the input image is firstly preprocessed to segment the input image into lines of text; these lines of texts are segmented into atomic segments, such that no atomic segment could be larger than any single character. These atomic segments are combined into groups (combined segments). The combined segments are processed by neural network classifier to recognize them, and finally in the last stage, the text line is constructed again in Unicode text.

Although the over-segmentation presented promising results, it imposes a considerable overhead. The over-segmentation stage will produce too many atomic segments which are either ignored if small, or combined into combined segment. The number of the combined segment before recognition is considerable high, for example one word (رتبت) of four characters will produce 12 atomic segments combined latter into 7 combined segments.

To boost the recognition system accuracy, in [5] the authors presented an Arabic character recognition system that is mainly composed of three recognition systems; namely: Over-segmentation, Sliding neural network system, and whole word recognition system. The three systems are operated in parallel on the input word, trying to recognize the text without segmenting or extracting the character. The best performance obtained from the three subsystems was around 80% as the authors stated in their paper.

Another technique to boost the performance of the neural network based character recognition system is to employ disambiguation rules and robust spelling correction to enhance the recognition [6].

Hidden Markov Models (HMM) was also used for optical character recognition [3][8]. These systems were inspired by continuous speech recognition (CSR) systems. In [3] the authors presented a language independent OCR system depending on the HMM and by utilizing the BBN BYBLOS continuous speech recognition system [10]. The presented system was identical to the BYBLOS CSR system except for the preprocessing and feature extraction stages.

A remarkable feature of the system presented in [3], that it is segmentation free. The system does not segment the image of a line-of-text neither to characters nor even to words. Instead of segmentation, the line is divided into a set of overlapping frames.

In this paper, we present a new and novel Arabic optical character recognition (AOCR) system. The system accepts a scanned-page image containing a set of text lines, and affected by typical noise level. The system preprocesses the image before the separate lines are handled to the character extraction phase. Our approach depends on segmenting the line into tokens and extracting characters features in order to recognize the characters and to assemble the output line of text. A neural network is used to process the features and to classify them into one of the characters.

The rest of the paper is organized as follows: in the second section, we introduce the architecture of the new AOCR system. The third section is devoted for introducing the new character recognition, while the fourth section describes the character recognition system. The construction of textual output (the assembly) is presented in the fifth section. In the sixth section, we present the results of the experimental study. Finally, we conclude our work in the seventh section.

2 Arabic optical character recognition System Architecture

Typical an AOCR program should accept, as an input, an image containing multiple lines with reasonable noise which is typically a scanned page of text. It should achieve reasonable efficiency in recognizing the words found in the image, and it should generate a textual clone of them. To achieve this broad requirement, we are proposing a new AOCR system. Our system Architecture is composed of four main phases, namely: preprocessing, character extraction, character recognition, and assembly.

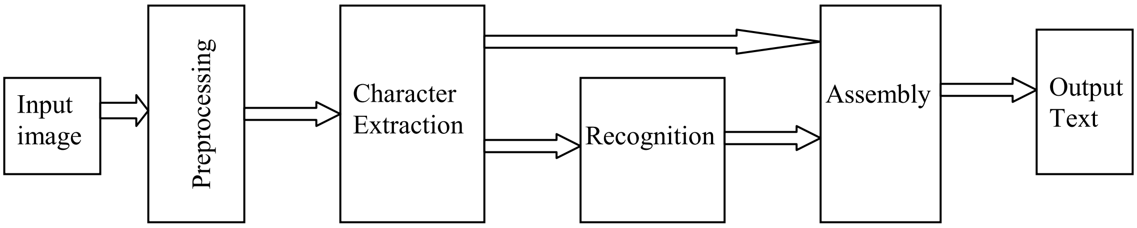


Fig. 1: System Architecture

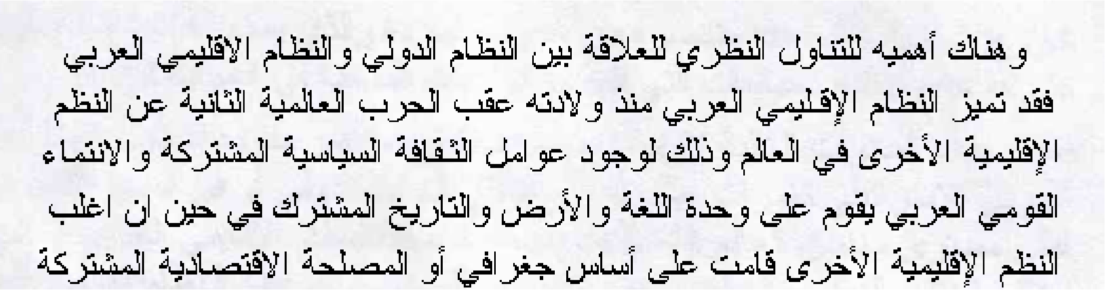

وهناك أهميه للتناول النظري للعلاقة بين النظام الدولي والنظام الاقليمي العربي
فقد تميز النظام الإقليمي العربي منذ ولادته عقب الحرب العالمية الثانية عن النظم
الإقليمية الأخرى في العالم وذلك لوجود عوامل الثقافة السياسية المشتركة والانتماء
القومي العربي يقوم على وحدة اللغة والأرض والتاريخ المشترك في حين ان اغلب
النظم الإقليمية الأخرى قامت على أساس جغرافي أو المصلحة الاقتصادية المشتركة

Fig. 2: Sample image containing typed Arabic text

In the preprocessing phase, the input image is processed in order to reduce the noise found in it, and to extract a complete line of the text. The role of the character extraction comes next to segment this line of text into tokens, and to extract all the available information. These tokens are processed in the recognition phase to recognize the set of possible characters that the token may be one of. Finally in the assembly phase, all the gathered information through the previous steps is used to construct the output textual version line. This architecture is shown in the figure 1.

In the preprocessing phase, the input image is subjected to three main processing stages: thresholding, morphological noise reduction, and line extraction. A thresholding technique is used to convert the image from colorful image into binary image of black background and white text. The colored input image is subjected to thresholding in order to compute the suitable intensity (threshold) of the image to convert it to a binary - Black and White - image. Our system employs Otsu's [7] algorithm to create the threshold of the image, handling the problem of different background colors, and the presence of multiple regions in the image.

Simple noise symptoms that may be found in binary images are greatly reduced by the use of morphological algorithm to remove isolated pixels, without the removal of dots and punctuation marks. Finally, line's boundaries in the image were identified by generating a histogram of the horizontal lines (i.e. a line of pixels). The histogram of lines between the text lines will appear as local minima (the histogram value approaches zero at these points).

3 Character Extraction

After preprocessing the input image (noise elimination and line extraction), we end up with an image containing a single line of text. The image contains a line of characters with different features and drawing characteristics. The main goal of the character extraction phase (as the name implies) is to segment the line into separate characters and to extract all the characters distinguishing features.

Extracting a token (a character or part of a character) is not simple as the reader may assume, especially if the characters are connected (cursive writing), and come in different sizes even within the same font size as the case in Arabic written text. Figure 2 shows an image containing typed Arabic text. Not only the Arabic written characters are connected to each other, but also the connection point with respect to the character comes in different places (from top, middle, or bottom). This variation in character drawing from no connection to a

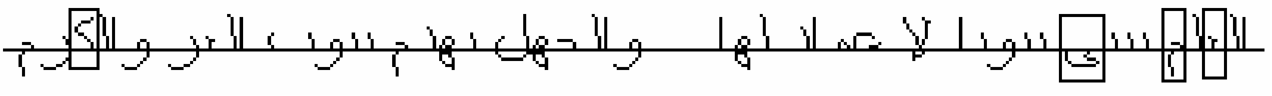


Fig. 3: The line of text after thinning and dots removal showing the main line and the bounding rectangle of some characters

connected in different places makes the extraction difficult. The following characteristics of the Arabic written language make the extraction a challenging problem:

- The Arabic written characters come in different non proportional sizes in the same font size, which makes the characters occupy unpredictable areas within the word.
- The same character can appear in completely different shapes and sizes within the same word, making extraction that depend on certain character features inefficient.
- The characters can appear in different orientations, since the Arabic characters within a word are not aligned and they can extend above or below the line.
- The presence of dots and hamza (ء) in different shapes above the characters.
- The font size highly effects characters and dots features and shapes (e.g. the three dots become connected in small fonts)
- Different Arabic fonts pose a dramatic change in the character's drawing and features with different font types. It is also noticed that different characters in different fonts may appear similar.

After studying the character extraction methods used in commercial products [9][11] and those found in literature, we observed that they adopted substantially different approaches. One of the approaches avoided the problem completely and treated the line of text as a stream of frames [3], While others used word-level recognition depending on a database of Arabic words [1][5]. In [2] the over-segmentation approach was proposed. In over-segmentation approach the word is segmented into atomic segments, and then the segments are combined and processed by neural network based classifier [2]. Other approaches depended greatly on the knowledge of characters starting and ending points [4][8][13], while this assumption is valid in online writing systems, as in PDAs, it is not justifiable for scanned images.

For the character extraction phase, we propose our new extraction algorithm. The algorithm was built in light of a lengthy statistical study performed on a set of Arabic typed texts with different font types and sizes. In what follows, we will summarize the results of our statistical study for the Arabic typed text features, and then we will present the extraction algorithm with a detailed explanation and an example.

3.1 Arabic typed-text main features

In this section, we summarize the findings of our study of the characteristics of Arabic characters and words. This study was done on the normal text line as well as on the line after thinning. The discussion below will stress on the silent features used in the extraction algorithm which will be explained in the next subsection.

Throughout the rest of the paper, we are going to use the following words to mean certain meanings. Firstly, the object is any character, part of a character, hamza or dot; it occupies an area, which is the number of pixels the object occupies. We may use the words: pattern and token interchangeably to mean the same thing (any isolated and extractable object from the image). Finally, the boundary rectangle is the minimum (in terms of area) rectangle that surrounds the object. We assume that the input image to this stage contains only one single line of text and nothing but an Arabic text (no figures, other language characters, or symbols), and that the line does not suffer from skewness.

In order to simplify the algorithm presentation and discussion we assume that no two characters overlap. This case could be handled by our algorithm, since any overlapping characters are going to be recognized as one entity, and a new token will be generated. Our statistical study showed that there is a limited cases were such occurrences can happen, they happen mainly with the character 'Kaf' ك, when it comes in the middle of a word, i.e. ' ـكـ '.

One of the most observable Arabic typed-text-line features is that there is always a horizontal reference line that must cut or touch every character in the line no mater of character’s shape or location with respect to other characters, we call this line the *Main Line*. Another important feature found in *most* of the characters is that the character occupies rectangular space in the line, so no two characters can extend over each other. These rectangles and the main line are shown in the figure 3. As it can be seen in figure

The Extraction Algorithm

```
1   (InputImg,LineHeight) = PreprocessingPhase (ImageFile)
2   (SkeletonImg,DotsImg) = DotsRemoval (InputImg)
3    SkeletonImg = Thinning (SkeletonImg)
4   CObjectsImg = ClosedObjectsDetection (SkeletonImg)
5   (TokensImg,MainLine) = MainLineRemoval (SkeletonImg)
6   TokensImg = InsertCObjects (TokensImg,CObjectsImg)
7   (TokensImg,NumberOfObjects) = LabelObjects (TokensImg)
8   TokensImg = UnifyObjects (TokensImg)
9   TokensImg = MainLineInsertion (TokensImg,MainLine)
10  TokensImg = ConnectCloseObjects (TokensImg)
11  (DotsImg,NumOfDots) = LabelObjects (DotsImg)
12  DotsRec = GetBoundingRectangle (DotsImg)
13  SpacesInfo = FindSpaces (TokensImg,LineHaight)
14  For (I = 1 : NumberOfObjects)
15      Character = ExtractChar (TokensImg,I)
16      CharachterRec= GetBoundingRec (Character)
17      CharacterHole = IsHole (Character)
18      CharacterRecognition = RecognitionPhase (Character)
19      CharactersInfo[I]=(CharacterRecognition,CharacterHole,CharacterRec,SpaceInfo)
20  End
21  OutputText = AssemblePhase (CharactersInfo, DotsImg, DotsRec)
22   return
```

3, every character will start from the main line, touch it or end at it.

The third feature in the Arabic characters is that there are certain ratios and relations between the character and different objects in the line. These ratios do not change when changing the font size. And these ratios and relations from different font types are nearly equal. This feature facilitates safe character resizing without affecting its features. For example, the character always covers bigger area than any dot or hamza around it, and the space within the word (i.e. between the characters of the same word) has a certain ratio with respect to the line height. Also when the character is written in different sizes its parts maintain a size ratio with respect to each other.

Finally, Arabic characters are built from lines with no sold filled areas, a characteristic that makes the output of the thinning stage preserves the input text skeleton; consequently the character's different parts are still there.

3.2 Extraction Algorithm

The extraction algorithm is built based on the characteristics described in the previous subsection. Next, we will present the algorithm in pseudo code followed by a detailed explanation. Also, a complete example is included for illustration. We refer to the output of the preprocessing stage as the *input*, which consists of an image of a single line of text. The input image is a binary image of zeros and ones arranged to form the characters and the background, where the background is black and the text is white.

The algorithm starts by preprocessing the input image to produce the input and an integer representing the height of the line as shown in figure 4. The second step in the algorithm is used to remove the dots and hamzas around the character. This is done by labeling every disconnected object with a unique label (ObjectID). Consequently, every dot or hamza will have a different label than the word they belong to. Since the area covered by the character (and hence the word area too) is greater than any dot or hamza regardless of the font type or size as mentioned previously, the dots can be easily identified and removed to another binary image, (DotsImg). As a result of this step the InputImg is separated into two images, one holding the line image without dots (SkeletonImg), and other holding the dots and hamzas (DotsImg).

Figures 5 and 6 show the results after apply the second step to image shown in figure 4. It is noted that the (ﻻ) character is divided into two parts among the two images. But the part that appears in the dots image can be safely ignored; since the remaining part in the characters image forms a unique token and does not look like any other character (i.e. the remaining part can alone identify the (ﻻ) character).

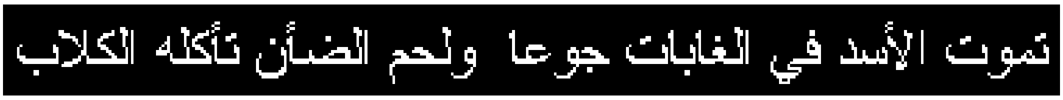


Fig. 4: Input Image

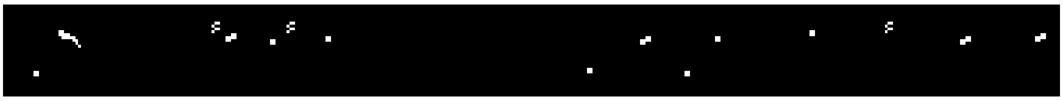


Fig. 5: Dots part of the input image (DotsImg)

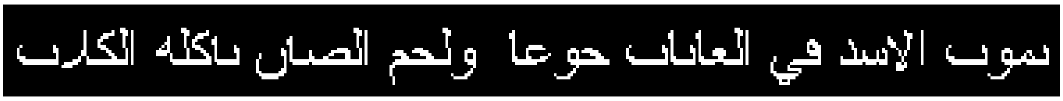


Fig. 6: Characters part of the input image (SkeletonImg)

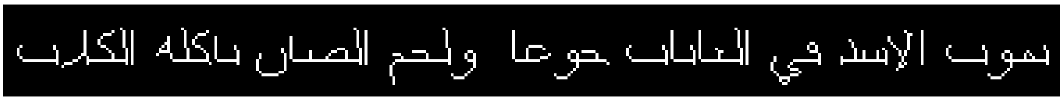


Fig. 7: SkeletonImg after thinning

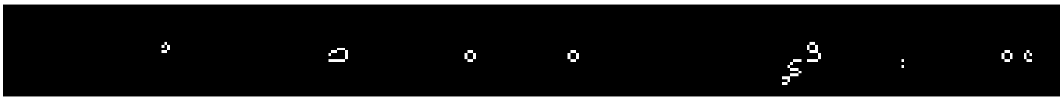


Fig. 8: CObjectsImg image

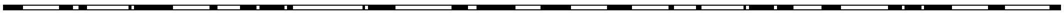


Fig. 9: Main line

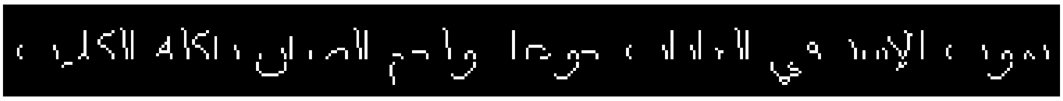


Fig. 10: TokensImg Image after main line removal

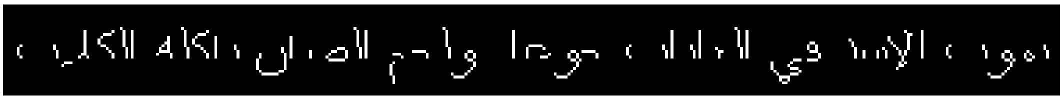


Fig. 11: TokensImg with the closed objects inserted

The third step will thin the characters image so the line width of the line constructing the characters will become one. This step works efficiently regardless of the original font size. The characters shape is preserved through thinning as described previously. Figure 7 shows the result of applying the thinning on the image shown in figure 6. In the fourth step, all objects forming a closed shaped are detected and copied to a separate image (CObjectsImg) to preserve them. Figure 8 shows the result after processing the image shown in the figure 7. This step is very useful to protect the shape from distortion that can happen when the main line is removed in the next step.

The main line is removed from the skeleton image in the fifth step. The main line connects all the characters and represents a reference line that all the characters must touch. An important feature of the main line of any typed text is that it is the most dens line in the image. So by simply generating the horizontal histogram of the image the main line can be identified. The line is copied to a separate image (MainLine) and the rest of the image is copied to a token image (TokensImg). The main line of our input image is shown in figure 9. The token image is shown in figure 10 and one can see that the image contains a set of tokens that characterizes the text. This step is followed by combining the image of closed objects with the token image to restore any distortion to the close-form objects as shown in figure 11.

In the seventh step, we label each object by replacing all object's pixels content (originally contain ones) with the object's label. Looking back at the figure 11, we notice that the token image contains parts of the characters or separate characters. Also, it is noticeable that some characters have some parts over the main line and others under

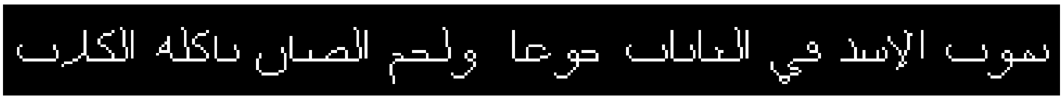


Fig. 12: TokensImg after inserting the main line

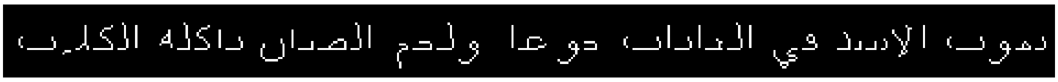


Fig. 13: TokensImg with the different objects separated apart

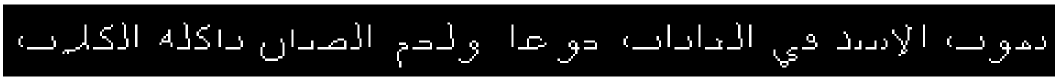


Fig. 14: SkeletonImg with the close objects are connected

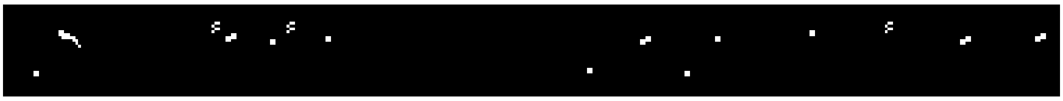


Fig. 15: The dots of the input image (DotsImg)

the main line. Therefore by removing the main line and labeling the disconnected object with different labels, these parts will have different labels while they belong to the same character. In the eighth step, these characters are detected and are given the same label. For example, if we look at the word (لحم) the figure 11, we can see that the last character of the word (م) is divided into two parts, one over the line and another underneath it. Since these parts belong to the same character, and in order to minimize the number of tokens generated by the extraction method, the two parts are given the same label.

The main line is reinserted in the ninth step. During the insertion process, each segment of the main line is labeled with the same label of the object that crosses that segment, since the segment was originally part of that character. The result of inserting the main line on the image shown in figure 11 is shown in the figure 12. For the purpose of illustration, figure 13 shows the objects with unique labels separated apart. If the character is formed by two parts around the main line, where one of them is completely above the line and the other is complete underneath it, these two parts should carry the same label. The tenth step takes care of this issue as show in figure 14. We can see that the (لآ) character in the last word was divided into two tokens in figure 13 which are combined into one token in figure 14.

In step 11, objects in the DotsImg are labeled to facilitate further processing as shown in figure 15. In order to know where the dot resides with regard to the main line, and to which character it belongs, we need to know the bounding rectangle of each dot (the rectangle with the minimum area surrounding the object). The twelfth step calculates the bounding rectangle for each dot.

An important part of Arabic word is the space within the word; this can be confused with the space between the words. In our statistical study, we found that the space between the characters of the same word no matter the size of the font, are proportional to the size of the text and its height. Typical values of the ratios are as follows: the space within the word will never go longer than 0.3 of the line height and the spaces between the words never goes shorter than 0.4 of the line height. This thirteenth step removes space within the word. Depending on this simple scheme for space recognition, our system achieved 100 percent efficiency, i.e. the system never (in all of our tests) added, omitted, or falsely recognized a space.

In the steps 14 through 20, we loop on all objects (a character or part of it) in the token image to collect the necessary data for recognition phase and assembly phase. We start by extracting the characters from the token image since these characters are now uniquely labeled. Figures 16–21 show some characters extracted from the image in figure 14. Then and for each character, we calculate the bounding rectangle to identify any related dots or hamzas and we check the character for holes. This is an important feature used if the tokens collide in the recognition phase. At this stage, the character can be identified and therefore, we call the recognition system to do this job. Finally and for statistical purposes, the system uses a simple data structure containing all the necessary information gathered through the character extraction and character recognition phases. It contains the output of the recognition phase, the information about the holes and the bounding rectangles. In addition to the information related to the spaces within and between the words. Finally and in the last step of the

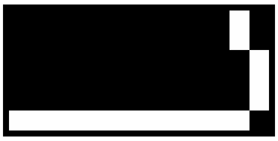


Fig. 16: Object number 2

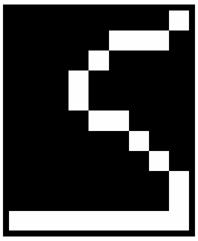


Fig. 17: Object number 5

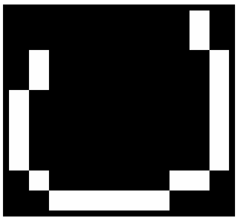


Fig. 18: Object number 13

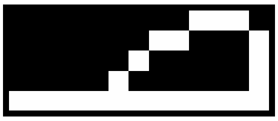


Fig. 19: Object number 16

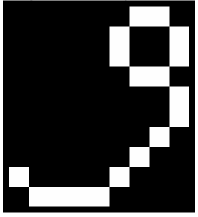


Fig. 20: Object number 22

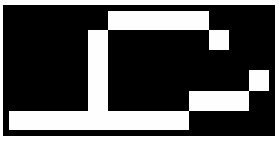


Fig. 21: Object number 24

algorithm, we call the assemble phase providing it with all the necessary information to assemble the line of text.

Our algorithm for character extraction generates 33 recognizable different tokens. Keeping in mind that the number of Arabic characters is 28 and that each character can appear in totally different shapes; generating only 33 different token is fairly excellent. This adds to the efficiency of our recognition phase, and to the system's overall accuracy.

4 Pattern Recognition

After the tokens have been extracted from the input image, they are passed to the recognition system. The recognition system receives one of 33 possible tokens augmented with all the extracted features and information about it; and tries to recognize it correctly.

Depending on a careful study and analysis of different design choices, we selected the 2-layer feed forward neural network. In order to boost our recognition system we firstly classified the input characters into two sets. The first set (set 1) contains all the tall characters (i.e. the ratio of the height to the width is greater than 1) and the second set (set 2) contains all the wide characters (i.e. the ratio of the height to the width is less than 1). For each set, we designed a separate neural network. Some characters, and due to differences in font types and sizes, may be considered tall in some cases and wide in others; consequently they will be members of the two sets. The members of each set are shown in appendix A.

After classifying the input pattern into one of the two sets, the pattern is resized into an appropriate size. All the characters belonging to the first set are resized to 10x5 image, while the members of the second sets are resized into 7x7 image. This classification followed by the appropriate resizing, further preserves the characters shape and adds to the accuracy of the recognition system.

The first neural network (for tall characters) is designed as two layers log-sigmoid/log-sigmoid network; the first layer consisted of 15 neurons and the second layer consisted of 26 neurons (one neuron for each token). We have used the standard log-sigmoid as a transfer function.

The second neural network (for wide characters) is also designed as a two-layer network; the first layer is composed of 15 neurons and the second layer is composed of 17 neurons. Also, the transfer function is the same log-sigmoid function. Figures 22 and 23 below show the architecture of the two neural networks.

The number of neurons in the hidden layer was selected after several experiments to produce the best recognition rate. Also, the log-sigmoid transfer function was picked because of its output range (0 to 1), which is suitable for classification purposes. The network is trained to output a 1 in the correct

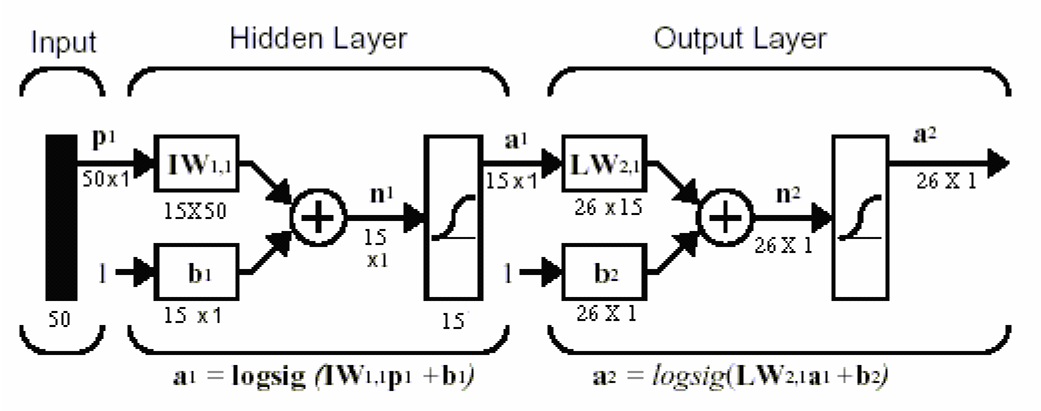


Fig. 22: Recognition Neural Network for the tall characters

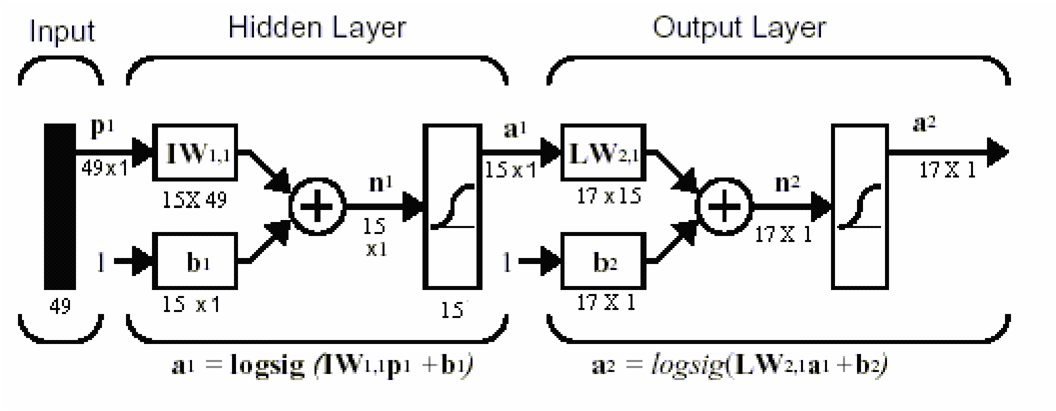


Fig. 23: Recognition Neural Network for the wide characters

position of the output vector and to fill the rest of the output vector with 0's. However, Input vectors may result in the network not creating perfect 1's and 0's. So after the network produces its results, the result is passed through the competitive transfer function that replaces the largest element of the output vector by 1, and places 0 in place of the other elements.

5 Assembly

The last stage in our AOCR is the assembly phase. In this phase, we use all the gathered information in the previous two phases (character extraction and recognition). This will finalize the character recognition to form complete words and to assemble the text line. Note that all the information related to the character recognition and line assembly were extracted in the character extraction phase, such as the dots image, their rectangle, space info …etc. Furthermore, the recognition phase classifies tokens into one of the 33 defined tokens. However, the 33 tokens are not complete characters. For example, the characters (خ ,ج ,ح), all of them will have the same shape without dots, and hence will be assigned to the same token in the recognition phase. In the assembly phase we will overcome this problem and other similar cases using the information gathered through the previous phases.

In order to complete the character recognition, we are in need to look to the related dots or hamzas. All what we have about the dots is an image containing them and the coordinates of the bounding rectangle. Therefore, we need firstly to recognize the dots and hamzas. Then, the assembly phase will integrate the recognized token with dots information to form possible characters. Finally, the assembly phase will fine tune the output and correct some recognition errors.

Most Arabic characters are surrounded by dots. A character may have one, two or three dots. And the

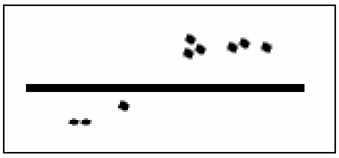


Fig. 24: Arabic dots possible combinations

Number of Dots	Eccentricity
1	0
2	>= 0.8
3	>0 and <0.8

Table 1: Typical value of eccentricity for the dots combinations

dots can be either above or below the character. The following figure shows all the possible patterns.

Differentiating between the hamza and the dots is relatively easy, since the hamza has a distinguishable structure from any dots patter, and the shape of hamza does not change with different font size. Consequently, the same classifier shown in figure 23 is used to recognize the hamza token. In addition, it is very simple to identify the relative location of the dots with regard to the character (above or below). However, what is not trivial is to count the number of the dots above each character because the dots do not come with standard drawing or shape and they depend on the font size. For instance, the three dots in big fonts appear as three separated objects, while in small fonts they appear as one entity.

To deal with this complication, we noticed that for large fonts we need to count the number of objects to determine the dot count since the dots are separated. However, in small fonts, the dots will be combined and appear as one object. So we depended on some observations of our statistical study that revealed an interesting feature in the dots found in small fonts. Our results show that the eccentricity of the dots varies significantly between the three possible dots combinations (one, two, or three). The eccentricity of the single dot is always very small, i.e. nearly zero. The eccentricity is very large for two dots and is moderate for three dots. The typical values are shown in the table 1. This property of the dots was used efficiently to recognize the dots in small fonts.

After combining the recognized tokens and dots, we have now meaningful characters. Note that there are possibly multiple characters for the same token as shown in Appendix A. A simple logic that links the position and number of dots to the token can uniquely identify the correct character. In addition, we need the hamza to identify the character in the case of the fourth token. For example, the fourth token could be on of the “بـ تـ ثـ فـ نـ يـ س ش ئـ” characters; the unique character is identified according to the previously mentioned logic.

Some errors could occur in the recognition phase, which maps to nonexistent characters. The assembly module can detect some of these errors and correct them by replacing the erroneous token with the most probable token according to the Arabic language structure (i.e. the character ‘د’ cannot have a dot beneath it, if such a combination occurred, the token is probably a mistaken ‘جـ’).

6 *Experimental Results*

An extensive testing was conducted to test the effectiveness of the presented AOCR. We firstly tested every phase separately and then performed a complete system testing. We have implemented the system using C++ and Matlab 6.1® [12]. The execution time depended on the text length and font size. After extensive testing, we found that the patterns in Appendix are good representative of the Arabic alphabets. The patterns can suffer from a small amount of variation according to the font size and type which is a natural result in Arabic written text.

The neural network was trained using a set of characters extracted from images containing Arabic text with different font types and sizes. Then the system was subjected to a blind test. The system was tested with three datasets: set 1 contained 5 scanned pages composed only of characters found in the training set. Set 2 includes images of 15 pages and contains text of 18pt size or greater scanned at 300dpi, and set 3 uses the same text in set 2 with font sizes less than 18pt and scanned at 300dpi. The results are shown in the table 2.

In addition to the previous results, thought the testing the recognition rate per pattern was also calculated for all the pages in the three sets. The rates are shown in the table 3 on the next page.

The results show a graceful degradation in system accuracy with small font sizes, this is due to the changes in the characters and the dots features among the different font sizes. The overall system recognition rate for testing data is 87% and 91% for Arial and Times New Roman fonts (frequently used fonts).

Dataset	Pages	Lines	Words	Characters	Recognition rate
Set 1	5	156	645	3643	98.4%
Set 2	15	363	964	5464	88.6%
Set 3	8	186	964	5464	86.3%

Table 2: Test Results

Pattern number	Possible characters	Recognition rate	Pattern number	Possible characters	Recognition rate
1	ء	96.7%	18	ق	82.7%
2	ا أ	93.2%	19	ك	83.8%
3	ب ت ث ف ك	96%	20	كـ	95.7%
4	بـ تـ ثـ فـ نـ يـ س ش ئـ	81.2%	21	ل	86%
5	جـ حـ خـ	79.6%	22	لـ	83.4%
6	ـج ـح ـخ	84.9%	23	لا لأ	93.9%
7	ج ح خ	82%	24	ـلا ـلأ	85.4%
8	ر ز	81.7%	25	مـ	85%
9	د ذ	80.1%	26	م	83.2%
10	ن س ش ص ض	82.5%	27	هـ	93.7%
11	ص ض	94.3%	28	ـهـ	96%
12	ط ظ	94.9%	29	ـه ـة	94.8%
13	عـ غـ	91%	30	ه ة	96.3%
14	ـعـ ـغـ	87.1%	31	ـو	88%
15	ـع ـغ	86%	32	ي	93%
16	ع غ	89.1%	33	ي	93.9%
17	فـ قـ	95.4%			

Table 3: Recognition rate of each pattern

4 Conclusion

The Arabic language nature poses unique challenges for the optical character recognition field. The Arabic language is written cursively, with different shapes for the same character, also the presence of dots and hamzas adds to the system complexity.

In this paper, we presented a new and novel Arabic optical character recognition system, which depended mainly on extracting a set of features for each character, and provide all the extracted information to recognition and assembly phases. The system introduces a new approach which utilizes the characteristics of the Arabic language; it utilizes the information in the main line of text to create tokens that characterize the characters. Our system achieved a very promising result of being able to correctly recognize input text images with 87% percent accuracy across different fonts and different font sizes.

References:

[1] Erlandson, E. J., Trenkle, J.M., Vogt, R.C., "Word-level recognition of multifont Arabic text using a feature-vector matching approach" *Proceedings of the SPIE*, Vol. 2660-08, San Jose, 1996.

[2] Gillies, A.M, Erlandson, E.J., Trenkle, J.M., Schlosser, S.G., "Arabic Text Recognition System", *Proceedings of the Symposium on Document Image Understanding Technology*, Annapolis, Maryland, 1999.

[3] Zhidong Lu, Issam Bazzi,Andras Kornai, John Makhoul, Premkumar Natarajan,Richard Schwartz, " A Robust, Language-Independent OCR System " , In: Robert J. Mericsko (ed): *Proc. 27th AIPR Workshop: Advances in Computer-Assisted Recognition SPIE Proceedings* 3584 1999

[4] E. Gomez Sanchez, Y.A. Dimitriadis, M. Sanchez-Reyes Mas, P.Sanchez Garcia, J.M.

Cano Izquierdo, J. Lopez Coronado , "On-Line Character Analysis and Recognition with Fuzzy Neural Networks" , I*ntelligent Automation and Soft Computing*, Vol. 7,No. 3, pp.161-162, 1998

[5] Trenkle, J.M., Gillies, A.M, Erlandson, E. J., Schlosser, S.G., "Arabic Character Recognition" *Proceedings of Symposium on Document Image Understanding Technology*. Bowie, Maryland, pp. 191-195, October 24-25, 1995.

[6] Trenkle, J. M. and R. C. Vogt, "Disambiguation and Spelling Correction for a Neural Network-based Character Recognition System." *In Proceedings Document Recognition*, Proc. SPIE 2181, eds. Luc M. Vincent and Theo Pavlidis, San Jose, CA, pp. 322-333, 6-10 February 1994.

[7] N. Otsu, "A threshold selection method from gray- level histograms", *IEEE transactions on systems, Man and Cybernetics*, vol. 9, no. 1, pp 62-69, 1979.

[8] Starner, T., J. Makhoul, R. Schwartz, and G. Chou. "On-line Cursive Handwriting Recognition Using Speech Recognition Methods." *In IEEE Proceedings International Conference on Acoustics, Speech, and Signal Processing*, pp. 125-128, April 1994.

[9] Tapas Kanungo, Gregory A. Marton, and Osama Bulbul , "OmniPage vs. Sakhr: Paired Model Evaluation of Two Arabic OCR Products " ,*Proceedings of SPIE Conference on Document Recognition and Retrieval (VI)*, vol. 3651 San Jose, CA; January 27-28, 1999

[10] Richard Schwartz; Chris Barry; Yen-Lu Chow; Alan Deft; Ming-Whei Feng; Owen Kimball; Francis Kubala; John Makhoul; Jeffrey Vandegrift , "The BBN BYBLOS Continuous Speech Recognition System ", *Speech and Natural Language*: Proceedings of a Workshop Held at Philadelphia, Pennsylvania, February 21-23, 1989

[11] Tapas Kanungo, Gregory A. Marton, and Osama Bulbul, " Performance Evaluation of Two Arabic OCR Products" ,*Proceedings of AIPR Workshop on Advances in Computer Assisted Recognition*, SPIE vol. 3584 Washington, D.C.; October 14-16, 1998

[12] MathWorks ,"*Image Processing Toolbox User's Guide Version 3*", MathWorks

[13] Tim Klassen ,"T*owards Neural Network Recognition Of Handwritten Arabic Letters* " Thesis, Faculty of Computer Science, Dalhousie University

Appendix A:

Table 4 lists all the patterns and their corresponding characters and table 5 shows the patterns generated by the extraction algorithm. Note that in table 5, the color of the images is inverted from the one generated by the extraction function only to make the patters more clear to the reader.

Pattern number	Possible characters	Possible Set	Pattern number	Possible characters	Possible Set
1	ء	1	18	ق	2
2	ا أ	1	19	ك	1 or 2
3	ب ت ث ف ك	1	20	كـ	1
4	بـ تـ ثـ فـ نـ يـ س ش ئـ	1 or 2	21	ل	1
5	جـ حـ خـ	2	22	لـ	1
6	ـج ـح ـخ	1	23	لا لأ	1
7	ج ح خ	1	24	ـلا ـلأ	1
8	ر ز	1 or 2	25	مـ	2
9	د ذ	1 or 2	26	م	1
10	ن س ش ص ض	2	27	هـ	1 or 2
11	ص ض	2	28	ـهـ	1
12	ط ظ	1 or 2	29	ـه ـة	1
13	عـ غـ	2	30	ه ة	1
14	ـعـ ـغـ	2	31	ـو	1 or 2
15	ـع ـغ	1	32	ي	1 or 2
16	ع غ	1	33	ي	1 or 2
17	فـ قـ	1 or 2			

Table 4: Possible Characters for each pattern

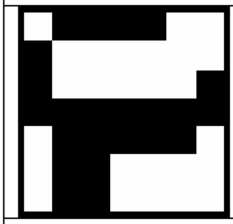
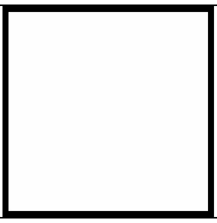
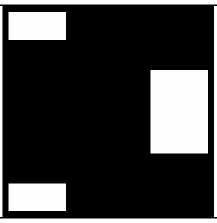
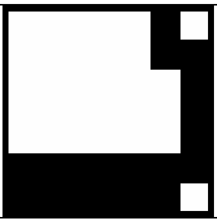
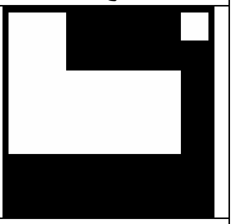
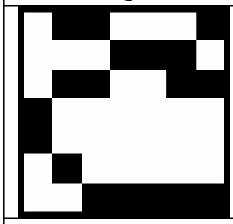
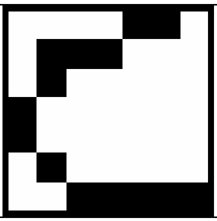
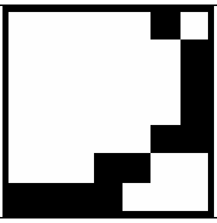
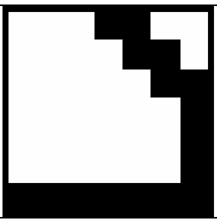
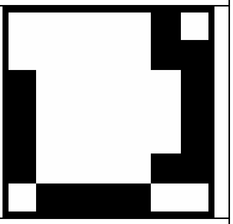
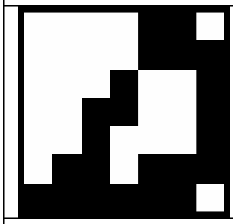
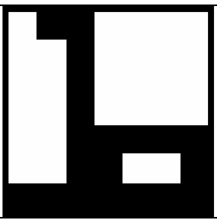
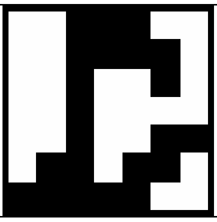
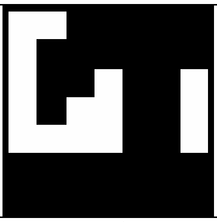
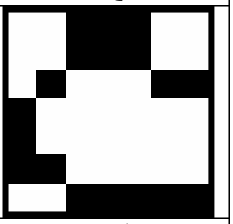
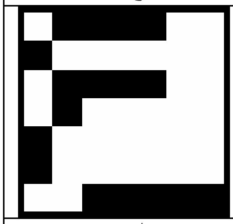
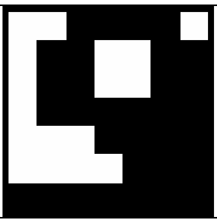
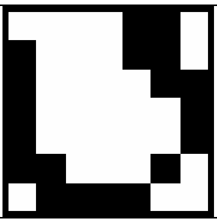
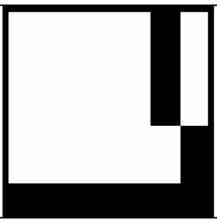
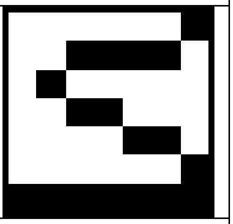
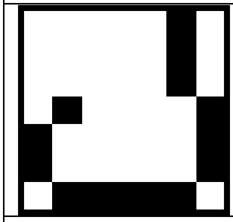
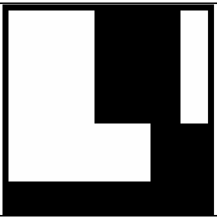
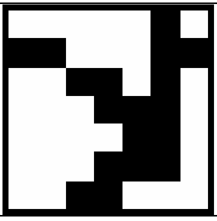
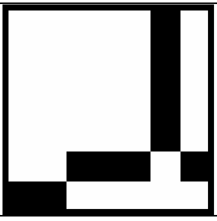
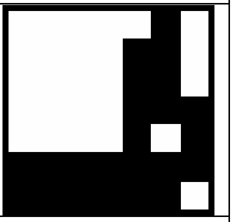
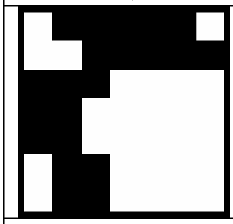
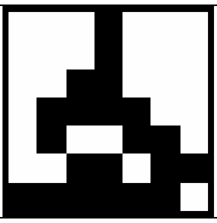
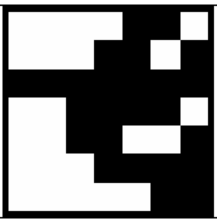
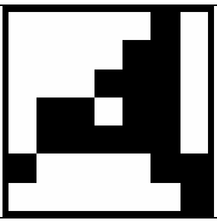
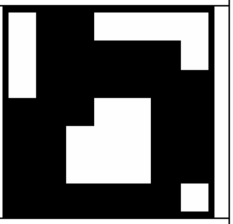
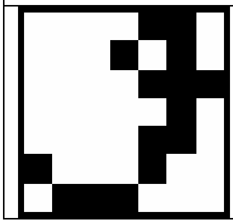
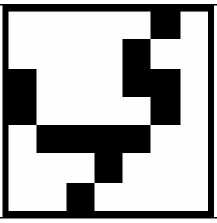
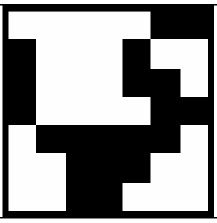
1 ء	2 ا	3 بـ فـ كـ	4 بـ تـ ثـ نـ يـ	5 ح
				
6 ح	7 ح	8 ر	9 د	10 ن س ص
				
11 ص	12 ط	13 ع	14 ع	15 ع
				
16 ع	17 فـ ق	18 ق	19 ك	20 ك
				
21 ل	22 ل	23 لا	24 لا	25 م
				
26 م	27 هـ	28 هـ	29 هـ	30 ه
				
31 و	32 ي	33 ي		
				

Table 5: Complete Pattern set of the Arabic alphabet

View publication stats