Automated System for Arabic Optical Character Recognition with Lookup Dictionary

Inad Aljarrah*, Osama Al-Khaleel*, Khaldoon Mhaidat*,
Mu'ath Alrefai¶, Abdullah Alzu'bi§, and Mohammad Rabab'ah§
Department of Computer Engineering
Jordan University of Science and Technology,
Irbid, Jordan 22110
Email*:{inad,oda,mhaidat}@just.edu.jo
Email¶:mmalrefai07@eng.just.edu.jo
Email§:{akalzubi07,mbalrbai07}@cit.just.edu.jo

Abstract— **In this paper an Arabic Optical Character Recognition system is implemented. The system takes a scanned image of an Arabic text as an input and generates an editable text out of it. The system starts by segmenting the document which is presented as an image into lines, then each line is also segmented into separate words, after that each word is further segmented to sub-words. Each word or sub-word is segmented into separate characters, and then a features extraction process is applied on each character to calculate its features vector. The feature vector is then compared with templates of feature vectors for each of the Arabic alphabet with their variations. The minimum distance classifier is used in the classification stage. A recognition rate of 93.5% is attained. To improve the accuracy of the system, a lookup dictionary is employed to correct some of the misclassified characters. This resulted in improving the accuracy to 96.1%. The results achieved are promising regardless that Arabic Optical Character Recognition is considered many times harder to handle than its counterparts in other languages like English due to the continuity between the letters in the same word.**

Index Terms— **Arabic OCR, Arabic characters, Segmentation, Recognition, Image processing**

I. Introduction

Optical Character Recognition (OCR) is the process of recognizing a typewritten or handwritten scanned image and converting it to an editable text. Arabic Optical Character Recognition (AOCR) is a special case of OCR which deals with Arabic language. OCR is used in many applications such as those that include processing of filled forms like healthcare or in banking where checks are preferred to be automatically processed without human association. It also made searching documents stored as images viable through converting them into text that is easily handled and processed by computers. Any OCR system consists of many stages, in each stage a specific task is performed and the output of each step is provided as an input to the next stage. Usually the main stages that any OCR system consists of include preprocessing, segmentation, feature extraction and classification. In this work, an Arabic Optical Character Recognition system is implemented. The system uses a novel character recognition-based segmentation method to segment and recognize Arabic characters. The input to the system is firstly preprocessed by converting the gray-level image to a binary image, this step is justified because the information in the gray-level value of the pixel is useless and gives no extra meaning and also binary images are much easier to handle than gray-level images since each pixel is represented by only one bit. The next step in the preprocessing stage involves applying a median filter on the binary image to eliminate noise such as isolated pixels. The last step in the preprocessing stage is to invert the image so that the pixels that represent characters have values of ones while the background has values of zeros. The second stage is segmentation which includes line segmentation, word segmentation, sub-word segmentation, and character segmentation. It is worth mentioning that the segmentation stage in AOCR is not as easy as OCR for other languages like English due to two main reasons, the first is the continuity of characters in Arabic which makes finding the borders of the given letter hard, and the second is that many of the Arabic letters have many forms that change according to the position of the letter whether it is in the beginning, the middle, or the in end of the word. The feature extraction phase includes calculating the features vector, which consists of seven features: area of the letter, length of the letter, width of the letter, difference between the upper and lower portions of the letter, first moment, second moment, and number of objects in the letter. The last phase is the classification stage in which the resultant feature vector is matched to a database of feature vectors that includes samples of each of the twenty nine alphabetical Arabic letters in all of their forms. The minimum distance classifier is used in this phase, where a value is calculated to represent the distance between the extracted feature vector and the stored feature vectors. Obviously the letter is matched with the sample that has the lowest value. To improve the accuracy of the system, a lookup dictionary is employed to correct some of the misclassified characters. After calculating the distance between the extracted feature vector and the stored templates, a penalty value is added to the distance of each template that results in a word that does not exist in the dictionary. The idea behind this method stems

doi:10.4304/jetwi.4.4.362-370

from the fact that the probability of having a word in the recognized text that does not exist in the dictionary is much lower than a word that exists in the dictionary. By adding the penalty value, the classification process moves from the simple minimum distance classifier that assumes the classified classes probabilities to be equally likely to the optimal classifier which takes in account that some classes are more probable and so the simple minimum distance classifier is not the best option.

The rest of this paper is organized as follows: Section II presents the related work. Section III discusses the proposed methodology. Section V talks about the experiments and gives some experimental results. Section IV presents the employment of lookup dictionary in order to improve the recognition rate. Finally, the conclusion and the future work are presented in Section VI

II. Related Work

Most of the work conducted on Arabic OCR (AOCR) concentrated on tackling the main problem of AOCR which is the segmentation of characters; this is mostly due to the cursive nature of Arabic language. In [1], [2], and [3], the authors used a morphological operations approach to segment lines into characters . Authors in [4] applied the contour representation to spot segmentation points. In [5], an Arabic OCR system that uses a recognition-based segmentation technique is proposed. The goal was to avoid the segmentation problems intrinsic to AOCR. In their approach they separate the horizontally overlapping Arabic words/subwords. However, their work is not a character based segmentation. [6] tackled the problem of overlapping characters using a contour-following algorithm which labels the detected contours. The joint fragments connecting the characters are spotted by measuring the line thickness. Another part of AOCR that attracted a lot of research is feature selection and classification. [7] employed measurements of moments and relationships among them as features and used a Bayesian classifier for classification is presented in. On the other hand, the work in [8], [9], [10], and [11] used neural networks to classify features.In [12] the authors implemented a system that uses a set of language independent features. though, these features are baseline dependent, which means that any slight error in locating the baseline will result in a feature extraction problem. In order to improve the recognition rate of the OCR systems, [13] suggested using a dictionary based error correction method . A context based error correction method is implemented in [14] for finding and correcting OCR non-word and real word errors. They claim increasing the error correction rate by 4 folds, but the main concern about the results is the small sample that the system was tested against . In our approach, we use a novel character recognition-based segmentation method to segment and recognize Arabic characters. Up to the knowledge of authors, this technique has not been reported in literature. The approach combines the segmentation and recognition stages. This idea improves the ability of detecting the borders of the connected characters.

III. Methodology

As shown in Figure 1 the input of the system is a gray level scanned image of an Arabic text document and the output is an Arabic text file.

Image of an Arabic text
Arabic OCR System
Arabic text file

Figure 1: General view of the system

A detailed view of the system is illustrated in Figure 2 where the image goes through seven steps in its way to be converted into a text file. Each of these steps will be discussed independently later on.

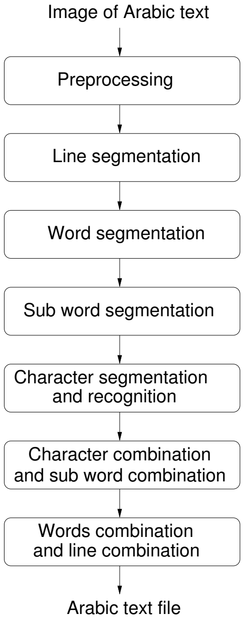
Image of Arabic text
Preprocessing
Line segmentation
Word segmentation
Sub word segmentation
Character segmentation and recognition
Character combination and sub word combination
Words combination and line combination
Arabic text file

Figure 2: proposed OCR system

A. Preprocessing

The OCR System starts by applying a preprocessing phase as shown in Figure 2. Usually OCR systems deal with two-color image documents: The color of the text and the color of the background. Since dealing with black and white images is much simpler than dealing with other image color scales (gray and colored), the scanned document is converted into binary image as a preprocessing step. This of course would simplify the features extraction process. Moreover it would greatly help in the lines, words, and sub words splitting during the processing phase.

Converting an image into its binary scale is simply done in MATLAB using the built in function im2bw (Image, level). In image processing, noise in images is one of the major sources for errors. One possible way to overcome the effect of the noise is to apply noise filter on the image. In the proposed system, the median filter is adopted for noise filtering. The median filter sets the pixel to the median value in the window that is applied on it. As a result, the small objects with size less than a given size (the size of the dot that appears in some arabic characters) will be filtered out.

Since the OCR here deals with binary scale images, a pixel with value 1 surrounded by many pixels of value 0 is an example of a noise. This is illustrated in Figure 3. Applying a median filter of a 3x3 window on this part of the image would definitely remove this noise and the pixel with 1 value would be assigned 0 value.

```
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0
```

Figure 3: Noise example

In binary scale images, 0 represents black and 1 represents white. Text document usually has the text written in black and the background is white. This means that in the scanned image of a text document if the image converted into black and white, the 0 pixel values are the text and the 1 pixel values are the background. The OCR system does the processing on the text and this involves different arithmetic operations like summation and scaling and so on. Also, there may be a need for comparisons. Doing these operations on zero values would give invaluable results. Therefore, as a very important preprocessing step, the black and white image is inverted such that the background is black and the text is white.

B. Line Segmentation

The first step in the recognition process is to segment the text image into lines. This step can be done by two ways:

1. Take horizontal projection.
2. Remove dots then take the horizontal projection.

In the horizontal projection scheme, the sum of all pixels in each row is calculated. If the sum is zero then start new line and end the current line. The problem here is that the dots in a line may be recognized as a separate line. For example consider the single line:

بيت

This line will be recognized as three different lines: an upper line that contains the two dots of the ت, a second line that contains the skeleton of the letters, and a third line contains the three dots under that line. The distance between the last pixel in any line and the first pixel in the next line is greater than the maximum distance between any two pixels in the same line. For example consider the two lines:

بيت

خالد

In the first line, the distance between the lowest pixel in the ي and its dots or the upper pixel in the ت and its dots is smaller than the distance between the lowest pixel in the first line and the upper pixel in the second line. This fact is employed to overcome the problem of splitting a single line into two or three lines by looking for more than one row of zeros after a none-zero row. The detection of a sequence of zero rows guarantees capturing the whole line. The number of sequential zero rows or threshold to be detected before splitting the line depends on the distance between the rows which depends on the font size. The pseudo code for the line splitting is given in Algorithm 1.

Algorithm 1: Line Splitting

```
Input: Threshold
Output: Starts and ends of lines
/* HP:Horizontal Projection */
/* Th:Thresold */
begin
    Counter ⟵ 0
    while HP=0 do
        Read new row
        Calculate HP
    repeat
        Get HP of next line
        if HP=0 then
            Counter++
            if Counter≥ Th /* # of zero rows */
            then
                Store row number
                Counter⟵0
        else
            Counter⟵0
    until Until End
end
```

In the second way, dots and hamza and other small objects that are placed above or under letters are removed. The line is then split based on the horizontal projection without the need for detecting a number of zero rows after the line. However, this method is computationally intensive because it requires a segmentation procedure to detect the mentioned small objects in order to remove them.

C. Word Segmentation

The next step after splitting the lines is the word splitting or segmentation within each line. There are two possible ways for this process:

1. Take the vertical projection of the line.
2. Eight bits connectivity.

In the first methods, if the sum of vertical line of pixels is zero then a gap occurred and a guess of having the

0	0	1
1	x	0
1	1	1

Figure 4: 8 bits neighbors of a pixel

word is made. However, this gap may occur between two unconnected letters within the same word. For example, considering the line:

لا حول ولا قوة الا بالله

The vertical projection with just looking for a zero column will split the word قوة into two words قو and ة which is a problem. To solve this problem same approach like in the case of the line segmentation of waiting to a consecutive rows of zero is adopted. The only difference is this time looking for consecutive columns of zeros with the fact that the distance between two words is greater than the distance between the sub words within the same word. After splitting each word into different segment, the word is then split into sub word segments for further processing. Assume having threshold 2 (number of zero columns to wait) greater than the maximum distance between sub words. The pseudo code for word segmentation is illustrated in Algorithm 2.

Algorithm 2: Word Splitting

```
/* Th2:Threshold 2                                  */
Input: Line of Arabic text,Th2
Output: Words of the line
begin
    Trim the zero columns from left and right
    index1 ⟵ index of first column in line
    index2 ⟵ index1+1
    repeat
        if $sum \neq 0$ and Counter $\geq$ Th2 then
            Store Line from index1 to index2
            index1 ⟵ index of current column
            index2 ⟵ index of next column
            Counter ⟵ 0
        else if $sum=0$ then
            Counter++
    until Until End
end
```

In the second method of word segmentation, the 8-bit connectivity, each pixel like X has 8 bits neighbors as illustrated in Figure 4. If the coordinates of X are (u, m), then the coordinates of the 8 bit neighbors are given by: $(u+1, m)$,$(u-1, m)$,$(u, m+1)$,$(u, m-1)$,$(u+1, m+1)$,$(u-1, m-1)$,$(u-1, m+1)$,$(u+1, m-1)$. Two bits are considered connected if they are both have the same value. Before applying this method the dots and hamza are removed. Therefore this method requires intensive computation time.

After having each word in a different segment, sub word segmentation is applied to make the letter segmentation (which is the next step) easier. The sub words pseudo code is illustrated in Algorithm 3.

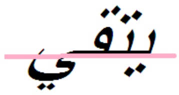

Figure 5: An example of an Arabic word with dots

Algorithm 3: Sub Word Splitting

```
Input: Word of Arabic text
Output: Sub Words of the word
begin
    Trim the zero columns from left and right
    index1 ⟵ 1
    while True do
        Get sum of next column
        if $sum=0$ then
            index2 ⟵ current column index
            store sub word from inndex1-index2
            break
        else
            continue
        while True do
            Get sum of next column
            if $sum=0$ then
                index1 ⟵ current column index
                break
end
```

D. Characters Splitting and Recognition

For character splitting, a deep study for the Arabic characters features is considered. Considering the following Arabic text line:

ومن لم يذد عن حوضه بسلاحه

يهدم ومن لا يتقي الشتم يشتم

The individual letters are contained by separate rectangle given that لا which is two characters is considered as a single character. As a second observation, in the word يتقي there is a straight segment that goes through all letters in the word as shown by Figure 5.

This segment is detected by taking the horizontal projection of the sub words which have the maximum sum of ones. All characters are connected by this segment and if removed, characters are separated. If the dots in the words are removed as illustrated by Figure 6, the vertical projection of the number of ones going from right to left is constant at the base line between the characters and varies when traveling through letters. However, letters like س and ض suffer from over segmentation. Also the ر may split into two parts or even three depending on the font size and scanner resolution. The problem that arises is that removing all dots requires lots of computations. There is no accurate algorithm to split the sub word. They

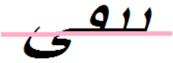

Figure 6: The Arabic word after removing the dots

are left combined. Splitting sub words into characters then recognize the characters in one step. Each letter in Arabic has special features that make it recognizable. Having features of all letters, one can decide if a given text is a letter or not. These features may include Area, Size, First moment, Number of objects in the letter, EulerNumber, Centroid, and Orientation. There are about 29 letters in Arabic that come in different formats based on the location of the character in the word: beginning, middle, end, and isolated.

E. Features Description

- Area of the character: which is estimated by the number of ones in the binary matrix of the character.
- Size of the character: which is a function of the length and the width of the character:
  - Length of the character is the number of rows in the binary matrix of the character.
  - Width of the character is represented by number of columns in the binary matrix of the character.
- First moment: which is calculated by firstly finding the centroid of the character then calculating the sum of the distances between the centroid and all the character pixels (with value equals to 1).
- Number of objects in the letter: which include the skeleton of the character, the dots above or under the character, and the other objects like the ء and so on.
- EulerNumber: It is given by the (number of objects – numbers of holes). Therefore, number of holes= number of objects – EulerNumber
- Centroid: Return the center of the input image x and y coordinate (mass center).
- Orientation: Put the input letter in ellipse then take the angle between the main diameter and the x axes. It gets a value between -90 and 90 (90 is added for normalization).

It should be pointed out that the given features are selected because they have the ability to distinguish among different characters as different Arabic characters have different size, area, centroid, etc. More details on these features can be found in [15].

A database that contains all letters in all shapes and their feature vector has been generated. It has been taken into consideration that the 'saad' is the widest letter and 'alef' is thinnest one. As an example to recognize the word illustrated in Figure 7 which is بعض, the system starts from widest character to the thinnest one in the Arabic characters. It then finds the feature vector and compares it with the letters vectors. Finally, it stores the character in an array to find the best match. This is repeated till the end of the sub word.

The example in Figure 8 demonstrates the whole system functionality.

IV. Employing Lookup Dictionary to Improve Recognition Rate

To improve the accuracy of the system, a lookup dictionary is employed to correct some of the misclassified characters. The idea is simple, if the detected character will result in a word that does not appear in the dictionary of Arabic language then give a penalty to that character matching value. This penalty will tend to move the ranking of that character down the list of potential characters. So if the character that got the second lowest matching value (minimum distance) results in a word that exist in the dictionary then it is possible that this character will end up with lower matching value after the penalty on the first character.

Algorithm 4: Dictionary based character matching

```
Input: Penalty value,Characters matching values
Output: Matching character
/* PV:Penalty value                        */
/* CMV:Character matching value            */
begin
    while True do
        getChar(CMV);
        getWord(Char);
        if Word exist in Dictionary then
            Break;
        else
            CMV⟵CMV+PV;
            Continue;
    getWord(Char(Min CMV));
end
```

The suggested approach tends to work well for AOCR, and the reason for that is due to the fact that Arabic language has many similar characters like (ح) and (خ), (ف) and (ق) and so on. A big portion of the errors that the system commits results from classifying one of those characters as being its similar character. Obviously this is due to the fact that similar characters tend to have similar feature vector values, and this results in a very close matching values, and hence a misclassification. The pseudo code for applying the lookup dictionary is illustrated in Algorithm 4.

It should be noted that not every non-dictionary word will be corrected to be a dictionary word. This is due to the fact that the predefined penalty value is finite and if the matching value difference between the first and the second potential characters is greater than this threshold then the character will not be modified even though the word does not appear in the dictionary.

In order to demonstrate the employment of the dictionary in the system, three different examples are presented

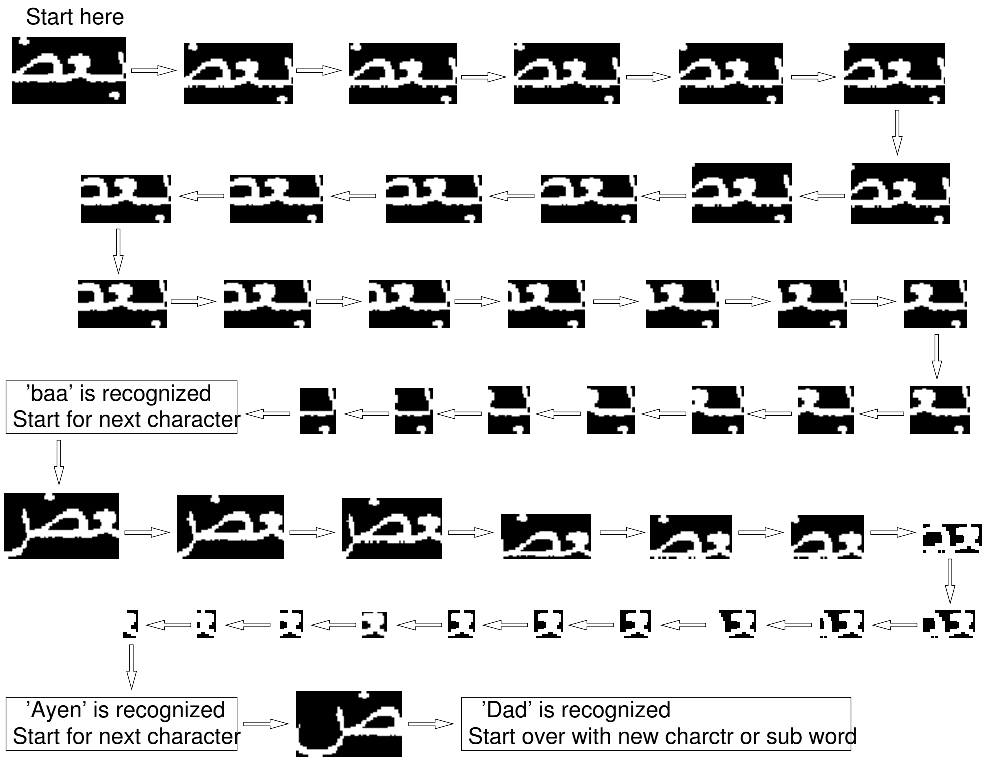

Figure 7: An example to illustrate the character splitting and matching process

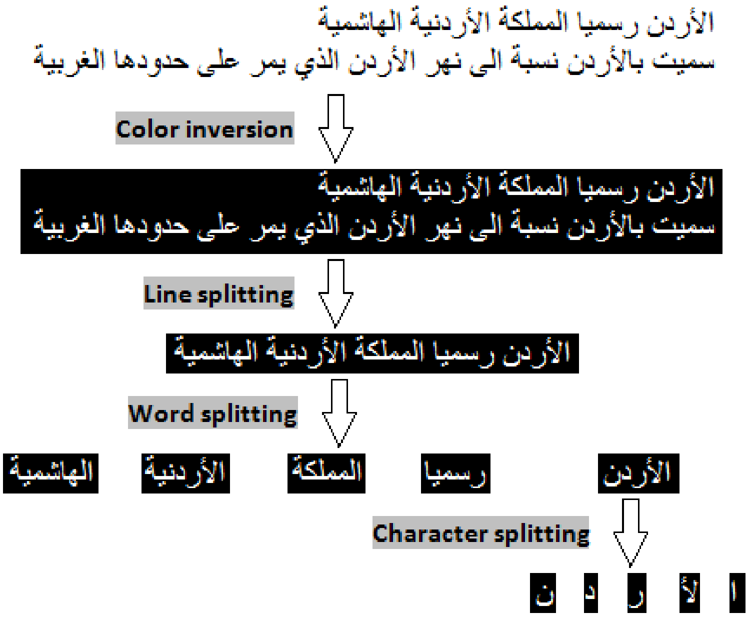
الأردن رسميا المملكة الأردنية الهاشمية
سميت بالأردن نسبة الى نهر الأردن الذي يمر على حدودها الغربية
Color inversion
الأردن رسميا المملكة الأردنية الهاشمية
سميت بالأردن نسبة الى نهر الأردن الذي يمر على حدودها الغربية
Line splitting
الأردن رسميا المملكة الأردنية الهاشمية
Word splitting
الهاشمية
الأردنية
المملكة
رسميا
الأردن
Character splitting
ن
د
ر
لأ
ا

Figure 8: Demonstration of the whole system steps

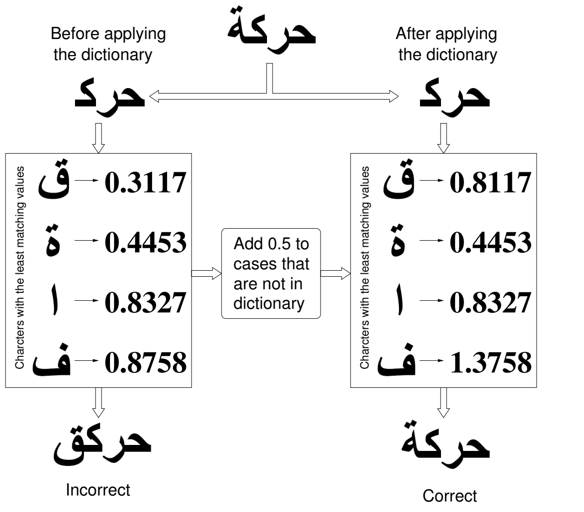
حركة
Before applying the dictionary
After applying the dictionary
حرك
حرك
Charcters with the least matching values
ق → 0.3117
ة → 0.4453
ا → 0.8327
ف → 0.8758
Add 0.5 to cases that are not in dictionary
ق → 0.8117
ة → 0.4453
ا → 0.8327
ف → 1.3758
حركق
Incorrect
حركة
Correct

Figure 9: Dictionary employment in recognizing the word حركة

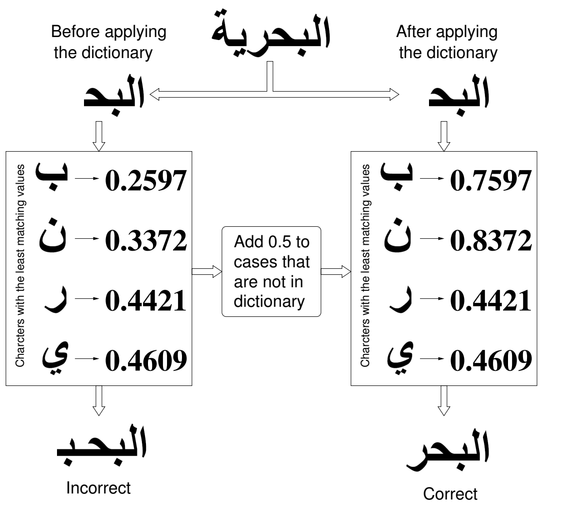
البحرية
Before applying the dictionary
After applying the dictionary
البح
البح
Charcters with the least matching values
ب → 0.2597
ن → 0.3372
ر → 0.4421
ي → 0.4609
Add 0.5 to cases that are not in dictionary
ب → 0.7597
ن → 0.8372
ر → 0.4421
ي → 0.4609
البحب
Incorrect
البحر
Correct

Figure 10: Dictionary employment in recognizing the word البحرية

in Figures 9, 10, and 11. These examples were extracted for the samples that have been used to test the system. Two of these examples shown how the dictionary helps in improving the recognition rate. On the other hand, the third example shows a case where dictionary does not fix the incorrectly recognized character. In Figure 9, the word to be recognized is حركة. At the time of recognizing the last character in the word which is ة, the character ق has the least matching value if the dictionary is not considered. Therefore, the word حركة is recognized incorrectly as حركق. By applying dictionary for this case, the system adds the penalty value (0.5) to the cases that are not part of the dictionary. For example, حركق is not part of the dictionary and the matching value for the character ق is increased by 0.5. On the other hand, the word حركة is part of the dictionary and the matching value of the character ة is not changed. This would reorder the matching values of all characters such that the character ة has the least matching value. As a result, the word حركة is correctly recognized as حركة.

In Figure 10, the word البحرية is to be recognized. If the dictionary is not considered, the fifth character is recognized incorrectly as ب which has the least matching value. After applying the dictionary two matching values are adjusted because they are not part of the dictionary. These are البحب and البحن. The other two cases البحر and البحي are part of the dictionary and the matching values are not affected. Hence, the least matching value after applying the dictionary is the one for the character ر which is the correct character.

In the third example presented in Figure 11, all cases are part of the dictionary. Therefore the matching values are not changed and in both cases the third character which is ر is incorrectly recognized as ن.

V. Experimental Results

The proposed approach has been implemented and tested using Matlab. Different Arabic text samples have been scanned and saved as images for testing purposes. For each sample, the number of correctly recognized character are counted and the the recognition rate is computed.

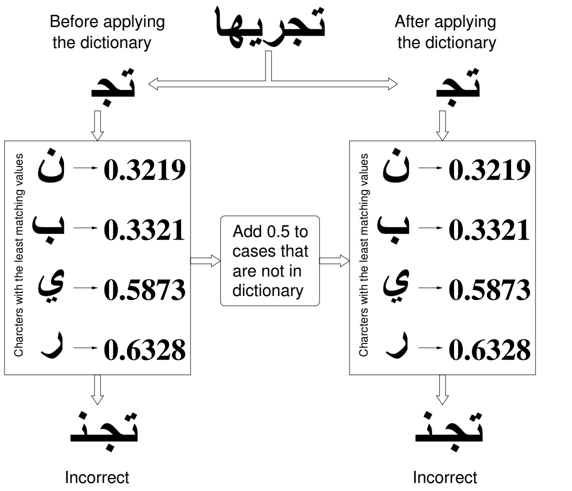
تجريها
Before applying the dictionary
After applying the dictionary
تج
تج
Charcters with the least matching values
ن → 0.3219
ب → 0.3321
ي → 0.5873
ر → 0.6328
Add 0.5 to cases that are not in dictionary
ن → 0.3219
ب → 0.3321
ي → 0.5873
ر → 0.6328
تجن
Incorrect
تجن
Incorrect

Figure 11: Dictionary employment in recognizing the word تجريها

Table I reports the results for some of these samples. As can be figured out from Table I, the estimated overall recognition rate for these samples is 93.5%. This recognition rate outperforms the accuracy of 90% that is reported in [5].

The same samples that has been used to obtain the results of Table I were tested using the system after applying the dictionary. The new results are reported in Table II. It is clear from these results that applying the dictionary improves the recognition rate for different samples in the table. Therefore, the overall recognition rate is improved from 93.5% to 96.1% as shown in the table. It should be pointed out that a thorough study of the results shows that the majority of the errors are cases of

Table I: Results for a scanned arabic text document samples using the proposed system

Sample No.	No. of characters per sample	No. of correctly recognized characters
1	54	46
2	57	52
3	60	59
4	57	54
5	60	51
6	29	29
7	60	59
8	49	47
9	55	52
10	54	49
11	62	58
12	48	47
13	55	52
14	15	12
15	51	47
16	47	42
17	49	43
18	51	49
19	40	37
20	61	61
21	52	51
Total	**1066**	**997**
Recognition ratio=93.5%		

Table II: Results for the samples in Table I after applying the dictionary

Sample No.	No. of characters per sample	No. of correctly recognized characters
1	54	47
2	57	53
3	60	60
4	57	55
5	60	53
6	29	29
7	60	60
8	49	49
9	55	54
10	54	51
11	62	59
12	48	48
13	55	52
14	15	14
15	51	48
16	47	44
17	49	45
18	51	51
19	40	39
20	61	61
21	52	52
Total	**1066**	**1024**
Recognition ratio=96.1%		

either miss-segmentation or due to the similarity between some of the Arabic language characters.

VI. Conclusion and future work

An Arabic Optical Character Recognition system is implemented. The system takes a scanned image of an Arabic text as an input and generates an editable text out of it. Promising results are achieved regardless that Arabic Optical Character Recognition is considered many times harder to handle than its counterparts in other languages like English due to the continuity between the letters in the same word. Challenges that have been faced during the implementation of the AOCR system comprises connectivity between letters, position dependency of letters shape, variable length of letters based on their neighbors from the left and right, and similarity between some Arabic letters like غ and ع. To improve the accuracy of the system, a lookup dictionary is employed to correct some of the misclassified characters. This step helped improve the recognition rate from 93.5% to 96.1%. As a future work, different aspects can be considered as an improvement or extension to this work. These may include: Using size invariant features and Arabic handwritten text recognition.

References

[1] D. Motawa, A. Amin, and R. Sabourin, "Segmentation of arabic cursive script," in *Proceedings of the 4th International Conference on Document Analysis and Recognition*, ser. ICDAR '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 625–628. [Online]. Available: http://dl.acm.org/citation.cfm?id=646270.685484

[2] B. H. Al-Badr, "A segmentation-free approach to text recognition with application to arabic text," Ph.D. dissertation, Seattle, WA, USA, 1995, uMI Order No. GAX95-37297.

[3] B. Al-badr and R. M. Haralick, "A segmentation-free approach to text recognition with application to arabic text," *International Journal on Document Analysis and Recognition*, pp. 147–166, 1998.

[4] T. Sari, L. Souici, and M. Sellami, "Off-line handwritten arabic character segmentation algorithm: Acsa," in *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, ser. IWFHR '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 452–457. [Online]. Available: http://dl.acm.org/citation.cfm?id=851040.856901

[5] A. Cheung, M. Bennamoun, and N. Bergmann, "An arabic optical character recognition system using recognition-based segmentation," *Pattern Recognition*, vol. 34, no. 2, pp. 215 – 233, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320399002277

[6] K. Romeo-Pakker, H. Miled, and Y. Lecourtier, "A new approach for latin/arabic character segmentation," in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 2, aug 1995, pp. 874 –877 vol.2.

[7] H. Al-Yousefi and S. S. Udpa, "Recognition of arabic characters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 8, pp. 853–857, Aug. 1992. [Online]. Available: http://dx.doi.org/10.1109/34.149585

[8] M. M. Fahmy and S. Al Ali, "Automatic recognition of handwritten arabic characters using their geometrical features," *Studies in Informatics and Control*, vol. 10, no. 2, 2001.

[9] O. Al-Jarrah, S. Al-Kiswany, B. Al-Gharaibeh, M. Fraiwan, and H. Khasawneh, "A new algorithm for arabic optical character recognition," in *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, ser. AIKED'06. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society

(WSEAS), 2006, pp. 211–224. [Online]. Available: http://dl.acm.org/citation.cfm?id=1364262.1364299

[10] R. Haraty and C. Ghaddar, "Neuro-classification for handwritten arabic text," in *ACSIEEE International Conference on Computer Systems and Applications*, july 2003, p. 109.

[11] R. A. Haraty and C. Ghaddar, "Arabic text recognition," *International Arab Journal of Information Technology*, vol. 1, no. 2, pp. 156–163, 2004.

[12] R. El-Hajj, L. Likforman-Sulem, and C. Mokbel, "Arabic handwriting recognition using baseline dependant features and hidden markov modeling," in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, aug.-1 sept. 2005, pp. 893 – 897 Vol. 2.

[13] M. Bokser, "Omnidocument technologies," *Proceedings of the IEEE*, vol. 80, no. 7, pp. 1066 –1078, jul 1992.

[14] Y. Bassil and M. Alwani, "Ocr post-processing error correction algorithm using google online spelling suggestion," *CoRR*, vol. abs/1204.0191, 2012.

[15] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. New Jersey, USA: Prentice Hall, 2007.