Journal of Al Azhar University–Gaza (Natural Sciences),(2006),Vol. 8, 85-108 ISSN 1810-6366

Segmentation of Discrete Arabic Script Document Images

Ibrahim S. I. Abuhaiba*

Department of Electrical and Computer Engineering, Islamic University of Gaza, Gaza, Palestine.

Abstract: *In this paper, we address the problems of line segmentation and character segmentation for discrete Arabic script documents. A robust algorithm to efficiently segment lines in very general textual documents is presented. Also, we present an approach to character segmentation problem without any restriction on the style of text, hence; more realistic documents are dealt with. Vertical white cuts are supported with connected component analysis to aid in segmentation. A dataset that contains natural Arabic text with diacritics was used as the basis of our dataset. The line segmentation algorithm was tested on the whole dataset consisting of the 157 images achieving a line success rate that exceeds 97%. Some extra lines were generated. These extra lines can be appended to adjacent actual lines in a succeeding OCR stage. Our approach for character segmentation was tested on two pages of discrete Arabic script. The overall character success rate was 94.4%. The algorithms were implemented and run on a Pentium III 866MHz PC with 128 MB RAM. The average time required to segment one character was 62 msec. Programming optimisations and more powerful computers can be used to speed up the segmentation process.*

Keywords: *OCR, Discrete Arabic Script, Line Segmentation, Character Segmentation*

Introduction

Two important problems in OCR research are line segmentation and character segmentation. In this paper, we address these problems in detail and present robust algorithms to efficiently segment lines and characters in very general textual documents.

In [1], a sub-word segmentation scheme is proposed which is independent of font size and font type. In [2], an Arabic multi-font character recognition approach is presented based on both Hough transform and wavelets transform for features selection and Hidden Markov Models for classification. In [3], three techniques that can be used to discriminate between text written in Arabic script and text written in English script are presented and evaluated. These techniques address the language identification problem on the word level and on text line level. In [4], an omnifont and unlimited-vocabulary OCR system for English and Arabic is presented. The system is based on Hidden Markov Models.

Although line segmentation is a nontrivial problem, it isn't touched in a detailed way in the above mentioned references and other similar works such

* To receive any correspondence: E-mail: ihaiba@mail.iugaza.edu

as
[5-15] and no figures are reported about the success rate of line segmentation algorithms.

In [16], the ground was laid for the development of new fonts to produce discrete Arabic script, for the first time, instead of cursive Arabic script. These fonts help in automatic document understanding and can be used to print books, newspapers, periodicals, and all other printed materials. Of course, all other properties of Arabic writing system are preserved when producing such fonts. The new fonts can be used for discrete Arabic typography such that the characters can be segmented with simple vertical white cuts. Two parameters were investigated to help in character segmentation: left and right white spaces. Quantified analysis and experiments were conducted to conclude a sufficient amount of these spaces. A font with left and right spaces of 160 FUnits each achieved a high segmentation success rate using vertical white cuts.

However, in [16], the document was assumed to contain text setup only in regular style so that it was possible to segment characters using only vertical white cuts. Using Italic fonts dramatically decreases the segmentation success rate since Italic characters usually overlap. This makes it difficult to depend only on vertical white cuts to segment very general documents. In this paper, we address character segmentation in addition to line segmentation. No restriction is put on the style of text; hence, more realistic documents are dealt with. Vertical white cuts are supported with connected component analysis to aid in segmentation. We expose our character segmentation approach presented in this paper to tests difficult more than real documents. To summarize, the two goals of this paper are:

1. Present a robust algorithm for line segmentation which is a document understanding problem that is rarely addressed in great detail in the literature.
2. Segment lines into characters using vertical white cuts in addition to connected component analysis.

In this work, two fonts are used as the basis of our experimentation: Simplified Arabic and Traditional Arabic which come as part of the Microsoft Office 2000 installation. From these two fonts, two discrete versions, Discrete Simplified Arabic and Discrete Traditional Arabic, are developed using the principles addressed in [16]. Figures 1 and 2 show two pages printed using these discrete fonts. The new fonts are used to setup and print all the documents used to test the algorithms reported in this paper.

The rest of the paper is organized as follows. Line segmentation is addressed in Section 2. Character segmentation is presented in Section 3. In Section 4, experimental results are reported. The paper is concluded in Section 5. A function that supports line segmentation is given in the appendix.

Line Segmentation

To segment lines of a document image, *I*, it is first skew-corrected. Then, pepper noise is removed. We assume that black (foreground) and white (background) pixels equal 1 and 0, respectively. Initially, the image, *I*, is considered as one region that is horizontally and recursively split into smaller regions. The initial number of regions to be segmented equals 1. To segment a region, the main line segmentation algorithm calls a function, *segmentRegion*, with the image *I* and the region to be segmented passed as parameters. The function returns the new regions to which the passed region is segmented. If only one region is returned then it is considered to constitute a single line that is added to the list of segmented lines. Otherwise, the returned new regions are added to the list of regions still requiring more splitting. This process is repeated until no more regions remain. Figures 3 and 4 show the result of the line segmentation algorithm when applied to the images of Figures 1 and 2, respectively. Horizontal solid lines are artificially added at the start and end rows of lines for demonstration purposes. A formal description of the algorithm is given in the appendix.

The details of the function *segmentRegion* in the line segmentation algorithm follow. First, if there is no single column, in the region to be segmented, having at least one black pixel, then the passed region is completely white, i.e., it hasn't any regions; hence, the function returns. Otherwise, let *numBlackColumns* be equal to the number of columns, in the passed region, each of which has at least one black pixel. Sometimes, horizontal white cuts can't be found between adjacent lines of text due to very small interline spacing. This results in the horizontal projections of these lines to overlap. However, the small area of overlap exhibits a clear valley. We force white cuts to be formed in such areas by defining the threshold

$$minHorizontalBlackPixels = \beta \times numBlackColumns \quad (1)$$

If the horizontal projection of any row of the passed region is less than *minHorizontalBlackPixels* then it is set to 0.

Black runs in the modified horizontal projection are found to define the average line height by

$$averageLineHeight = \frac{1}{numTotalBlackPixels} \sum_i numRunPixels[i] \times runLength[i] \quad (2)$$

where *numTotalBlackPixels* is the accumulated sum of the modified horizontal projection values, *numRunPixels*[*i*] and *runLength*[*i*] are the sum of projection values and the length of black run *i*. Black runs of length less than $\alpha \times averageLineHeight$ are cleared. This is done to prevent strips containing diacritics and dots from forming independent lines. New black

runs in the modified horizontal projection are found. The region passed to function *segmentRegion* is segmented into new regions that correspond to these runs. The topmost new region is expanded to the north of the passed region. Also, the bottommost new region is expanded to the south of the passed region. Every two adjacent regions are expanded to the spacing between them. Details of such expansion can be found in the appendix.

If the height of the region, r, between the starting row of the passed region and the starting row of the topmost new region $\geq \alpha \times$ height of topmost region, then if r contains a black blob with height exceeding a certain threshold, *dotHeight*, then r is added to the set of new regions. The check is repeated for the region between the ending row of the bottommost new region and the ending row of the passed region. A similar check is performed for every two adjacent new regions, r_i and r_{i+1}. Let *averageRegionHeight* be the average height of r_i and r_{i+1}. If the height of the space between these two regions is not less than $\alpha \times$ *averageRegionHeight* then add this space as a new region. The function *segmentRegion* is formally presented in the appendix.

Character Segmentation

After line boundaries are determined using the line segmentation algorithm, every line is segmented into characters in a process of many steps:

1. The baseline of the line is determined by finding the row in the line under consideration that exhibits the maximum horizontal projection value. The y coordinate of the baseline is set equal to that row. Figure 5(a) shows a portion of a line of discrete Arabic text. The calculated baseline is shown in Figure 5(b).
2. The line is segmented from right to left into what we call white cut components using vertical white cuts. Generally, a white cut component consists of more than one connected component. Figure 5(c) shows seven white cut components of the line of Figure 5(a). Notice that we artificially increased the white space between these components for demonstration purposes.
3. White cut components are segmented into connected components using a sequential labeling algorithm [17]. Connected components of every white cut component are rearranged from right to left. The white cut components of Figure 5(c), from right to left, consist of 2, 1, 2, 6, 2, 1, and 4 connected components, respectively.
4. Basic and secondary components are determined. A basic component is a connected component that constitutes the main body of a character. A secondary component is a connected component that associates a basic component to form a character. Sometimes a secondary component

touches a basic component due to blotting. The resulting hybrid component will be counted as a basic component. An example of this case is the second component from the right in Figure 5(d).

5. A character consists of one basic component and zero or more secondary components. If the baseline intersects a connected component, *CC*, or the distance between them doesn't exceed a certain threshold, d_{basic}, then *CC* is counted as a basic component, otherwise, it is a secondary component. If no connected component could be classified as a basic component, then all connected components are considered basic components. From Figure 5(b), it is clear that the white cut components, from right to left, contain 2, 1, 1, 2, 1, 1, and 3 basic components, respectively, and 0, 0, 1, 4, 1, 0, and 1 secondary components, respectively.
6. Now, for every white cut component, characters are formed. Every basic component is considered to form the main body of an independent character. So, initially, the number of characters equals the number of basic components. The degree of horizontal overlap between every secondary component and all basic components is found. The secondary component is assigned to the basic component that exhibits the maximum overlap. If there is no overlap between a secondary component, *SC*, and any basic component, then, *SC* is assigned to the nearest basic component in the same white cut component. In Figure 5(c), the first and second white cut components contain two basic components and one basic component, respectively. They don't contain any secondary components; so, they form two characters and one character, respectively. The third white cut component consists of a single basic component and a single secondary component; so, these two constitute a single character. The forth white cut component contains two basic components and four secondary components. The rightmost basic component is assigned the nearest two secondary components and the next basic component is assigned the remaining two secondary components. The remaining white components are similarly manipulated. Finally, the eleven characters shown in Figure 5(d) are obtained with artificial white space added between characters for demonstration purposes.

Results

Dataset:

The dataset contains natural Arabic text with diacritics. An MS Word copy of the document was obtained with the following statistics: 2125 paragraphs, 81235 words, 398880 characters, and 482563 characters with spaces. The file was formatted with the following difficult-to-segment settings: 8 points font size, single-line spacing, and bold Italic. We selected these settings to expose our algorithms to hard tests such that if they

succeeded under such conditions, then success is more guaranteed under more relaxed conditions. The 8 pt font size was selected since this is almost the smallest font size in which Arabic publications are usually printed. This size gives very small values of inter-character and inter-word spacing. The single-line spacing was selected to decrease the vertical white gap between adjacent lines especially when diacritics are used. Actually, such a small value of line spacing results in joined horizontal projections of adjacent lines which makes it impossible to segment lines using direct horizontal white cuts. The bold setting was mainly used to decrease the inter-character spacing to make it more difficult to segment the characters of documents printed using our designed discrete Arabic script fonts. The whole document was set in Italic to make it very difficult to depend only on vertical white cuts to segment the characters. Overall, we wanted to expose our algorithms to difficult tests. The file was first set using Simplified Arabic Discrete font and then printed in black and white and normal mode using an hp deskjet 845c printer generating 88 A4 pages for a total of 4495 lines. Another copy of the file was produced using Traditional Arabic Discrete font and printed using the same printer settings yielding 69 A4 pages of 3897 lines.

A Xerox copy of the printed Simplified Arabic Discrete font copy was obtained. Then, a second Xerox copy of the first Xerox copy was prepared to add natural random degradation to the images to test the robustness of our algorithms. This second Xerox copy of the Simplified Arabic version was scanned, using an hp ScanJet 3400C scanner at 300 dpi to obtain a set of 88 images. The Traditional Arabic printed copy was not Xerox-copied; it was directly scanned using the same scanner settings to obtain a set of 69 images. Thus, a total of 157 images were prepared. The dataset obtained was initially gray-level images that were converted to binary images using Paint Shop Pro 6. Samples of the dataset are shown in Figures 1 and 2 for Simplified Arabic and Traditional Arabic fonts, respectively.

The algorithms were implemented and run on a Pentium III 866MHz PC with 128 MB RAM. In the line segmentation algorithm, the parameters β and α were empirically determined to be 0.01 and 0.70, respectively, and *dotHeight* was set to 4 pixels. The line segmentation algorithm was tested on the whole dataset consisting of the 88 Simplified Arabic images and 69 Traditional Arabic images. All actual lines in these pages were successfully segmented. The line success rate, *LS*, defined as:

$$LS = \text{Actual No. of Lines / Computed No. of Lines} \quad (3)$$

is 97.3% and 98.4% for the Simplified and Traditional Arabic fonts, respectively. Figures 3 and 4 show the result when the algorithm was applied to the images of Figures 1 and 2, respectively. For demonstrative purposes, two horizontal solid lines are added to delimit each line. The original images suffer from some skew that was eliminated by the skew

correction component used in our algorithm. Skew correction is a necessary step for the line segmentation algorithm to succeed. Notice that lines containing only a single word were successfully segmented.

Sometimes, diacritics were missed and couldn't be grouped to any line as exemplified in Figure 6, where the under-line diacritics between the second and third solid lines and between the forth and fifth solid lines couldn't be grouped to any line. This happens because: (1) the height of such diacritics is small enough for our algorithm to clear them, and (2) they are delimited by horizontal white cuts. This shouldn't constitute problems in a succeeding OCR stage since such diacritics are auxiliary marks added to only adjust the pronunciation of characters and their absence doesn't change adjusted characters.

The cases in which extra lines were obtained could be classified into four categories:

1. Extra lines generated due to pepper noise, black extraneous dots, or blobs, see Figure 7.
2. Extra lines consisting of under-line diacritics, see Figure 8.
3. Extra lines consisting of above-line diacritics, see Figure 9.
4. Extra lines consisting of original dots, see Figure 10.

These extra lines can be appended to adjacent actual lines in a succeeding OCR stage. Details about extra lines are shown in Table I. As it was mentioned above, the images of Simplified Arabic font are scanned images of the second Xerox copy of the printed pages, while the images of the Traditional Arabic font are scanned images of the original printed pages. This resulted in an average number of extra lines per page of 1.42 for the case of the Simplified Arabic font compared to 0.91 extra lines for the Traditional Arabic font. This is explained more if we notice that 64.0% of extra lines belong to category of type (1), i.e., the pepper noise, black extraneous dots, or blobs category, which all are known bad side effects of Xerox photocopying. The Traditional Arabic pages were not Xerox photocopied; hence, weren't polluted with such noise, which resulted in only 1.6% of extra lines belonging to category (1). An example of category (1) extra lines is shown in Figure 7 where an extra line is located between the third and forth horizontal solid lines starting from the bottom of the image and consists of a single black dot.

For the first font, the percentages of categories (2, 3) extra lines, which are 15.2% and 20.0%, respectively, are small compared to that of category (1) indicating that Xerox photocopying is the main reason of extra lines. Category (4) extra lines, which corresponds to under-line original dots, exhibit a very low percentage, 0.8%. It seems to us that errors of categories (2, 3, 4) can be grouped into one category for which the last column of the table corresponds.

For the Traditional Arabic font, the dominant errors are those of category (2), 77.8%. This is due to the large distance between underline diacritics and the corresponding characters compared to the Simplified Arabic font. Category (4) extra lines are absent in this font. Overall, 1.20 extra lines are added to each page which is easily tolerable and can be manipulated with little overhead in succeeding OCR stages.

Our approach for character segmentation was tested on two pages set in Discrete Simplified Arabic and Discrete Traditional Arabic consisting of 4428 and 5672 characters, respectively. The parameter, d_{basic}, was set to 2 pixels. Reasons of segmentation errors were:

1. The basic component is assigned a secondary component of a previous or following basic component, Figure 11(a).
2. The basic component is assigned a secondary component from a preceding or following line, Figure 11(b).
3. A secondary component is classified as a basic component, Figure 11(c).
4. Two connected basic components couldn't be segmented into their individual components, Figure 11(d).

Table II details the percentage of occurrence of the above four reasons. The numbers in the leftmost column refer to the aforementioned reasons. Every occurrence of these errors results in almost two adjacent characters being in error resulting in 2 × 136 and 2 × 149 characters for these two fonts. Therefore, the character segmentation success rate is 93.9% and 94.7% for the Simplified Arabic and Traditional Arabic fonts, respectively. The overall success rate is 94.4%. The most prominent reason of error is that secondary components are assigned to adjacent basic components none of which are the proper ones. For our dataset, this is mainly due to the Italic setting of the whole dataset, which results in some secondary components going further from the mother basic component. However, since Italic setting is rarely used in normal textual pages we expect that this cause of error to be too much smaller than what is reported here. Or, the fonts can be redesigned such that with Italic setting the secondary components don't fly away from their actual basic components. The next prominent cause of error is that a secondary component close to the baseline is classified as a basic component. Reason no. 2 is a side effect of our line segmentation algorithm since in the presence of diacritics and small interline spacing it is very difficult to segment every line with all its components, which results in few secondary components being grouped with adjacent lines. However, Arabic documents are usually setup with bigger interline spacing alleviating from this problem. Reason no. 4 is the result of blotting and other reasons of noise, but it occurs very rarely.

The average time required to segment one character is 62 msec. This

includes the time for skew correction, noise removal, and line segmentation. It is true that this is slow; however, we concentrated on the functional correctness of our algorithms. Many programming optimisations and hints can be used to speed up the segmentation process. In addition, using more powerful computers helps too much in this respect.

Conclusion

Two important problems in OCR research are line segmentation and character segmentation. In this paper, we addressed these two problems in detail and presented a robust algorithm to efficiently segment lines in very general textual documents. No restriction was put on the style of text; hence, more realistic documents were dealt with. Vertical white cuts were supported with connected component analysis to aid in segmentation. We exposed our character segmentation approach presented in this paper to tests difficult more than real documents.

The dataset we used to test our system contains natural Arabic text with diacritics. The line segmentation algorithm was tested on the whole dataset consisting of the 88 Simplified Arabic images and the 69 Traditional Arabic images. All actual lines in these pages were successfully segmented. The line success rate was 97.3% and 98.4% for the Simplified and Traditional Arabic fonts, respectively. Sometimes, diacritics were missed and couldn't be grouped to any line. Some extra lines were generated. These extra lines can be appended to adjacent actual lines in a succeeding OCR stage.

The overall success rate of the character segmentation algorithm was 94.4%. The most prominent reason of error was that secondary components were assigned to adjacent basic components none of which are the proper ones. For our dataset this was mainly due to the Italic setting of the whole dataset which resulted in some secondary components going further from the mother basic component. However, since Italic setting is rarely used in normal textual pages, we expect that this cause of error to be too much smaller than what is reported here. Or, the fonts can be redesigned such that with Italic setting the secondary components don't fly away from their actual basic components.

The algorithms were implemented and run on a Pentium III 866MHz PC with 128 MB RAM. The average time required to segment one character was 62 msec. This included the time for skew correction, noise removal, and line segmentation. Many programming optimisations and hints can be used to speed up the segmentation process. In addition, using more powerful computers helps too much in this respect.

References:

1. A. Zidouri, M. Sarfraz, S . A. Shahab, and S . M. Jafri, "Adaptive

Dissection Based Subword Segmentation of Printed Arabic Text", Ninth International Conference on Information Visualisation (IV'05), **2005**, 239-243.

2. Nadia Ben Amor and Najoua Essoukri Ben Amara, "Combining a hybrid Approach for Features Selection and Hidden Markov Models in Multifont Arabic Characters Recognition", Second International Conference on Document Image Analysis for Libraries (DIAL'06), **2006**, 103-107.
3. Ahmed M. Elgammal and Mohamed A. Ismail, "Techniques for Language Identification for Hybrid Arabic-English Document Images", Sixth International Conference on Document Analysis and Recognition (ICDAR'01), **2001**, 1100.
4. Issam Bazzi, Richard Schwartz, and John Makhoul, IEEE Transactions on Pattern Analysis and Machine Intelligence, **1999**, 21(6),495-504.
5. S. Mori, C. Y. Suen, and K. Yamamoto, Proceedings of The IEEE, **1992**, 80(7),1029-1058.
6. S. N. Srihari, Proceedings of The IEEE, **1992**, 80(7),1120-1132.
7. S. Mori, K. Yamamoto, and M. Yasuda, IEEE Transactions on Pattern Analysis and Machine Intelligence, **1984**, 6(4), 386-405.
8. G. Nagy, Proceedings of The IEEE, **1992**, 80(7),1093-1100.
9. H. S. Baird, Proceedings of The IEEE, **1992**, 80(7),1059-1065.
10. S. Tsujimoto and H. Asada, Proceedings of The IEEE, **1992**, 80(7),1133-1149.
11. M. Bokser, Proceedings of The IEEE, **1992**, 80(7),1066-1078.
12. J. Schurmann, N. Bartneck, T. Bayer, J. Franke, E. Mandler, and M. Oberlander, Proceedings of The IEEE, **1992**, 80(7), 1101-1119.
13. L. O'Gorman, IEEE Transactions on Pattern Analysis and Machine Intelligence, **1993**, 15(11),1162-1173.
14. K. Y. Wong, R. G. Casey, and F. M. Wahl, IBM J. Res. Develop., **1982**, 26(6), 647-656.
15. H. Fujisawa, Y. Nakano, and K. Kurino, Proceedings of The IEEE, **1992**, 80(7), 1079-1092.
16. I. S. I. Abuhaiba, The Arabian Journal for Science and Engineering (AJSE), **2003**, 28(1B), 77-94.

17. G. X. Ritter and J. N. Wilson, Handbook of Computer Vision Algorithms in Image Algebra, 2nd edition, CRC Press LLC, **2001**, 177.

APPENDIX

Algotithm. Line Segmentation
Input: Skew corrected and pepper-noise-filtered image, *I*
Returned Values:
1. Total number of lines, *numTotalLines*, in the image
2. Starting row, *lineStartRow*[*i*], and ending row, *lineEndRow*[*i*], of line no. *i*, *i* = 0, 1, …, *numTotalLines* – 1

Procedure:

Step 1. Initialization

(a) Assuming foreground and background pixels in *I* equal 1 and 0, respectively, the image *I* is surrounded with a one-pixel-thickness boundary of 0 pixels.

(b) Let *numTotalLines* = 0. Define *numTotalRegions* as the number of regions in the image, *I*. As the image is horizontally and recursively split into smaller regions, *numTotalRegions* is updated. Initially, *numTotalRegions* = 1, i.e., there is only a single region, region no. 0, in *I*.

(c) The limits of region no. 0 extend from row 1 to row *height* of the image *I*, where *height* is the height of *I*. Therefore, let *regionStartRow*[0] = 1 and *regionEndRow*[0] = *height*. Let *numProcessedRegions* = 0.

Step 2. Recursive Region Segmentation

While (*numProcessedRegions* < *numTotalRegions*)
{
segmentRegion(*I*, *regionStartRow*[*numProcessedRegions*],
regionEndRow[*numProcessedRegions*],
numNewRegions, *newRegionStartRow*[],
newRegionEndRow[]);
where *segmentRegion* is a function that segments a region located between *regionStartRow*[*numProcessedRegions*], *regionEndRow*[*numProcessedRegions*] in image *I* into a number, *numNewRegions*, of new regions each defined by *newRegionStartRow*[*i*] and *newRegionEndRow*[*i*], *i* = 0, 1, …, *numNewRegions* – 1.
Increment *numProcessedRegions*;
If(*numNewRegions* = 1)

```
    {
        The region constitutes a single line.
        lineStartRow[numTotalLines] = newRegionStartRow[0];
        lineEndRow[numTotalLines] = newRegionEndRow[0];
        Increment numTotalLines;
    }
    else
       for(i = 0; i < numNewRegions; i++)
       {
        regionStartRow[numTotalRegions] = newRegionStartRow[i];
        regionEndRow[numTotalRegions] = newRegionEndRow[i];
        Increment numTotalRegions;
        }
}
```

End of Algorithm.

Function segmentRegion

segmentRegion(*I*, *regionStartRow*, *regionEndRow*,
numNewRegions, *newRegionStartRow*[], *newRegionEndRow*[])
{

Let *numBlackColumns* be equal to the number of columns, in the region located between *regionStartRow* and *regionEndRow* in the image *I*, each of which has at least one black pixel.

If(*numBlackColumns* = 0)
{
numNewRegions = 0;
return;
}

Let the required minimum number of back pixels in a row of the region under consideration be equal to *minHorizontalBlackPixel*s = $\beta \times$ *numBlackColumns*.

Let *originalHorizontalProjection*[*i*] be equal to the horizontal projection of the input region, *i* = *regionStartRow*, *regionStartRow* + 1, …, *regionEndRow*.
Set *originalHorizontalProjection*[*regionStartRow* – 1] = 0,
originalHorizontalProjection[*regionEndRow* + 1] = 0.
Let *workingHorizontalProjection* = *originalHorizontalProjection*.

For(*i* = *regionStartRow*; *i* ≤ *regionEndRow*; *i*++)

If(*workingHorizontalProjection*[*i*] < *minHorizontalBlackPixels*)
workingHorizontalProjection[*i*] = 0;

Find the number, *numNonZeroRuns*, of nonzero runs in *workingHorizontalProjection*. Let *numRunPixels*[*i*] and *runLength*[*i*] be equal to the number of pixels and length of run *i*, respectively, *i* = 0, 1, *numNonZeroRuns* – 1.

Let *numTotalBlackPixels* be equal to the total number of black pixels in the input region.

Define the average line height, *averageLineHeight*, as in Equation (2).

In projection *workingHorizontalProjection*, clear the runs the length of which is less than $\alpha \times$ *averageLineHeight*.

Find the new nonzero runs in *workingHorizontalProjection*. Run no. *i* is located between *runStartRow*[*i*] and *runEndRow*[*i*], *i* = 0, 1, …, *numRuns* – 1, where *numRuns* is the number of nonzero runs.

Divide the input region into *numNewRegions* = *numRuns*, where new region no. *i* is defined by

newRegionStartRow[*i*] = *runStartRow*[*i*],
newRegionEndRow[*i*] = *runEndRow*[*i*], *i* = 0, 1, … , *numNewRegions* – 1.

Expand the topmost new region to the north of the input region as follows:

```
expandedNewRegionStartRow = newRegionStartRow[0];
minHorizontalProjection =
originalHorizontalProjection[expandedNewRegionStartRow];
for(i = newRegionStartRow[0] – 1; i ≥ regionStartRow; i--)
  if(originalHorizontalProjection[i] < minHorizontalProjection)
  {
      minHorizontalProjection = originalHorizontalProjection[i];
      expandedNewRegionStartRow = i;
  }
newRegionStartRow[0] = expandedNewRegionStartRow;
```

Expand the bottommost new region to the south of the input region as follows:

```
expandedNewRegionEndRow = newRegionEndRow[numNewRegions – 1];
minHorizontalProjection=
```

```
originalHorizontalProjection[expandedNewRegionEndRow];
for(i = newRegionEndRow[numNewRegions – 1]; i ≤ regionEndRow; i++)
  if(originalHorizontalProjection[i] < minHorizontalProjection)
  {
      minHorizontalProjection = originalHorizontalProjection[i];
      expandedNewRegionEndRow = i;
  }
newRegionEndRow[numNewRegions – 1] =
       expandedNewRegionEndRow;
```

Expand every two adjacent new regions to the spacing between them as follows:

```
For(i = 0; i< numNewRegions – 1; i++)
{
      expandedNewRegionEndRow = newRegionEndRow[i];
      minHorizontalProjection =
        originalHorizontalProjection[expandedNewRegionEndRow];
      for(j = newRegionEndRow[i] + 1; j < newRegionStartRow[i + 1]; j++)
        if(originalHorizontalProjection[j] < minHorizontalProjection)
          {
              expandedNewRegionEndRow = j;
                minHorizontalProjection =
                      originalHorizontalProjection[j];
          }

    expandedNewRegionStartRow = newRegionStartRow[i + 1];
     minHorizontalProjection =
       originalHorizontalProjection[expandedNewRegionStartRow];
       for(j = newRegionStartRow[i + 1] - 1; j > newRegionEndRow[i]; j--)
           if(originalHorizontalProjection[j] <
             minHorizontalProjection)
           {
                  expandedNewRegionStartRow = j;
                  minHorizontalProjection =
                        originalHorizontalProjection[j];
           }

     if(expandedNewRegionEndRow = expandedNewRegionStartRow)
         expandedNewRegionStartRow--;

       newRegionEndRow[i] = expandedNewRegionEndRow;
       newRegionStartRow[i + 1] = expandedNewRegionStartRow;
  }
```

If the height of the region, *r*, between the starting row of the input

region and the starting row of the topmost new region is relatively large compared to the height of the topmost new region, then if *r* contains a black blob with height exceeding a certain threshold, then add *r* to the set of new regions. This is implemented as follows:

Let *numRegions* = *numNewRegions*;
If(*newRegionStartRow*[0] – *regionStartRow* ≥ *α* × (*newRegionEndRow*[0] – *newRegionStartRow*[0] + 1))
{
    Check if there is a black run in the part of *originalHorizontalProjection* located between *regionStartRow* and *newRegionStartRow*[0].
    If there is such a run with a length not less than *dotHeight* then
    {
        newRegionStartRow[*numRegions*] = *regionStartRow*;
        newRegionEndRow[*numRegions*] = *newRegionStartRow*[0] – 1;
        Increment *numRegions*;
    }
}

Repeat the above check for the region between the ending row of the bottommost new region and the ending row of the input region as follows:

If(*regionEndRow* - *newRegionEndRow*[*numNewRegions* – 1] ≥
    α × (*newRegionEndRow*[*numNewRegions* – 1] –
        newRegionStartRow[*numNewRegions* – 1] + 1))
{
    Check if there is a black run in the part of *originalHorizontalProjection* located between *newRegionEndRow*[*numNewRegions* – 1] and *regionEndRow*.
    If there is such a run with a length not less than *dotHeight* then
    {
        newRegionStartRow[*numRegions*] =
          newRegionEndRow[*numNewRegions* – 1] + 1;
        newRegionEndRow[*numRegions*] = *regionEndRow*;
        Increment *numRegions*;
    }
}

For every adjacent two new regions r_i, r_{i+1}, i = 0, 1, …, *numNewRegions* – 2:
    Let *averageRegionHeight* be the average height of r_i and r_{i+1}.
    If the height of the space between these two regions is not less than *α* × *averageRegionHeight* then
    {

```
                Check if there is a black run in the part of
                originalHorizontalProjection located between
                newRegionEndRow[i] and newRegionStartRow[i+1].
                If there is such a run with a length not less than dotHeight
                then
                {
                        newRegionStartRow[numRegions] =
                                newRegionEndRow[i] + 1;
                        newRegionEndRow[numRegions] =
                                newRegionStartRow[i + 1] – 1;
                        Increment numRegions;
                }
        }
    Let numNewRegions = numRegions;
}
```

((تَدَاوَوْا، فإنَّ اللهَ عَزَّ وجَلَّ لم يضَعْ داءً إلا وَضَعَ لَهُ شِفاءً غيرَ داءٍ واحدٍ))، قالوا: ما هو؟ قال: ((الهَرَمُ)).

وفي لفظٍ: ((إنَّ اللهَ لم يُنْزِلْ داءً إلا أنزلَ له شِفَاءً، عَلِمَهُ مَنْ عَلِمَهُ وجَهِلَهُ مَنْ جَهِلَهُ)).

وفي ((المسند)): من حديث ابن مسعود برفعه: ((إنَّ اللهَ عَزَّ وجَلَّ لم يُنْزِلْ داءً إلا أنزلَ لَهُ شِفاءً، عَلِمَهُ مَنْ عَلِمَهُ، وَجَهِلَهُ مَنْ جَهِلَهُ)).

وفي ((المسند)) و((السنن)): عن أبي خِزَامةَ، قال: قلتُ: يا رسول الله، أرأيْتَ رُقًى نَسْتَرْقِيهَا، ودواءً نتداوى به، وتُقَاةً نَتَّقِيهَا، هل تَرُدُّ من قَدَرِ اللهِ شيئاً؟ فقال: ((هي من قَدَرِ اللهِ)).

فقد تضمَّنت هذه الأحاديثُ إثباتَ الأسباب والمسبَّبات، وإبطالَ قولِ مَن أنكرها، ويجوزُ أن يكون قوله((لكل داءٍ دواء))، على عمومه حتى يتداول الأدواءَ القاتلة، والأدواء التي لا يُمكن لطبيب أن يُبرئها، ويكون الله عَزَّ وجَلَّ قد جعل لها أدويةً تُبرئها، ولكن طَوَى علمَها عن البَشَر، ولم يجعل لهم إليه سبيلاً، لأنه لا علم للخلق إلا ما علَّمهم الله، ولهذا علَّق الذبيُّ صلى الله عليه وسلم الشِّفاءَ على مصادفة الدواء للداء، فإنه لا شيءَ من المخلوقات إلا له ضدٌّ، وكلُّ داء له ضد من الدواء يعالَج بضدِّه، فعلَّق الذبيُّ صلى الله عليه وسلم البُرءَ بموافقة الداء للدواء، وهذا قدرٌ زائدٌ على مجرد وجوده، فإنَّ الدواء متى جاوز درجة الداء في الكيفية، أو زاد في الكمية على ما ينبغي، نَقَلَه إلى داء آخر، ومتى قصر عنها لم يَفِ بمقاومته، وكان العلاج قاصراً، ومتى لم يقع المُداوِي على الدواء، أو لم يقع الدواء على الداء، لم يحصُل الشفاء، ومتى لم يكن الزمان صالحاً لذلك الدواء، لم ينفع، ومتى كان البدنُ غيرَ قابل له، أو القوةُ عاجزةً عن حمله، أو ثَمَّ مانعٌ يمنعُ من تأثيره، لم يحصل البُرء لعدم المصادفة، ومتى تمت المصادفة حصلَ البرءُ بإذن الله ولا بُدَّ، وهذا أحسنُ المحمَلَيْن في الحديث.

والثاني: أن يكون من العام المراد به الخاصُّ، لا سيما والداخل في اللَّفظ أضعاف أضعاف الخارج منه، وهذا يُستعمل في كل لسان، ويكونُ المراد أنَّ الله لم يضع داءً بَوْئِلَ الدواء إلا وضع له دواء، فلا يَدخل في هذا الأدواء التي لا تقبل الدواء، وهذا كقوله تعالى في الرِّيح التي سلَّطها على قوم عاد: ﴿ تُدَمِّرُ كُلَّ شَيْءٍ بِأَمْرِ رَبِّهَا﴾[الأحقاف : ٢٥] أي: كل شيء يقبلُ التدمير، ومِن شأن الرِّيح أن تدمِّره، وظائرُه كثيرة.

ومَن تأمَّل خلْقَ الأضداد في هذا العالَم، ومقاومةَ بعضها لبعض، ودفْعَ بعضها ببعض، وتسليطَ بعضها على بعض، تبيَّن له كمالُ قدرة الربِّ تعالى، وحكمتُه، وإتقانُه ما صنعه، وتفرُّده بالربوبية، والوحدانية، والقهر، وأنَّ كل ما سواه فله ما يُضاده ويُماثِلُه، كما أنه الغنيُّ بذاته، وكُلُّ ما سواه محتاجٌ بذاته.

وفي الأحاديث الصحيحة الأمرُ بالتداوي، وأنه لا يُنافي التوكل، كما لا يُنافيه دفْع داء الجوع، والعطش، والحرِّ، والبرد بأضدادها، بل لا تتم حقيقةُ التوحيد إلا بمباشرة الأسباب التي نَصَبها الله مقتضياتٍ لمسبَّباتها قدراً وشرعاً، وأن تعطيلها يقدَحُ في نفس التوكل، كما يَقْدَحُ في الأمر والحكمة، وبضعفه من حيث يظن مُعطِّلُها أنَّ تركها أقوى في التوكل، فإن تركها عجزاً يُنافي التوكلَ الذي حقيقتُه اعتمادُ القلب على الله في حصول ما ينفع العبدَ في دينه ودنياه، ودفْع ما يضرُّه في دينه ودنياه، ولا بد مع هذا الاعتماد من مباشرة الأسباب؛ وإلا كان معطِّلاً للحكمة والشرع، فلا يجعل العبدُ عجزه توكلاً، ولا توكُّلَه عجزاً.

وفيها ردٌّ على مَن أنكر التداوي، وقال: إن كان الشفاء قد قُدِّرَ، فالتداوي لا يفيد، وإن لم يكن قد قُدِّرَ، فكذلك. وأيضاً، فإنَّ المرض حصل بقَدَر الله، وقَدَرُ الله لا يُدفَع ولا يُرد، وهذا السؤال هو الذي أورده الأعراب على رسول الله صلى الله عليه وسلم. وأما أفاضلُ الصحابة، فأعلَمُ بالله وحكمته وصفاته من أن يُورِدوا مِثْلَ هذا، وقد أجابهم النبيُّ صلى الله عليه وسلم بما شفى وكفى، فقال: هذه الأدويةُ والرُّقَى والتُّقَى هي من قَدَر الله، فما خرج شيءٌ عن قَدَره، بل يُرَدُّ قَدَرُه بقَدَره، وهذا الرَّدُّ من قَدَره. فلا سبيلَ إلى الخروج عن قَدَره بوجه ما، وهذا كردِّ قَدَرِ الجوع، والعطش، والحرِّ، والبرد بأضدادها، وكردِّ قَدَرِ العدوِّ بالجهاد، وكلٌّ من قَدَرِ الله: الدَّافِعُ، والمدفوعُ، والدَّفْعُ.

وبقال لمُورِدِ هذا السؤال: هذا يُوجبُ عليك أن لا تُباشر سبباً من الأسباب التي تَجلِبُ بها منفعة، أو تَدفَعُ بها مضرَّة، لأن المنفعة والمضرَّة إن قُدِّرَتا، لم يكن بدٌّ من وقوعهما، وإن لم تُقدَّر لم يكن سبيلٌ إلى وقوعهما، وفي ذلك خرابُ الدِّين والدنيا، وفسادُ العالَم، وهذا لا يقوله إلا دافعُ الحق، معاندٌ له، فيُذكر القَدَرَ ليدفعَ حُجةَ المُحقِّ عليه، كالمشركين الذين قالوا: ﴿لَوْ شَاءَ اللهُ مَا أَشْرَكْنَا وَلاَ آبَاؤُنَا﴾[الأنعام: ١٤٨]، و﴿ لَوْ شَاءَ اللهُ مَا عَبَدْنَا مِن دُونِهِ مِن شَيْءٍ نَّحْنُ وَلاَ آبَاؤُنَا﴾[النحل : ٣٥]، فهذا قالوه دفعاً لحُجَّة الله عليهم بالرُّسُل.

وجوابُ هذا السائل أن يُقال: بقي قسمٌ ثالث لم تذكره، وهو أنَّ الله قَدَّر كذا وكذا بهذا السبب؛ فإن أتيتَ بالسَّبب حَصَلَ المسبَّبُ، وإلا فلا.

فإن قال: إن كان قَدَّر لي السَّببَ، فعلتُه، وإن لم يُقدِّره لي لم أتمكن من فعله.

قيل: فهل تقبل هذا الاحتجاجَ من عبدك، وولدك، وأجيرك إذا احتجَّ به عليك فيما أمرتَه به، ونهيتَه عنه فخالَفَك؟، فإن قبلتَه، فلا تَلُمْ مَنْ عصاك، وأخذ مالك، وقَذَفَ عِرْضَك، وضيَّع حقوقَك، وإن لم تَقبلْه، فكيف يكونُ مقبولاً منك في دفع حُقوق الله عليك .. وقد روى في أثر إسرائيلي: ((أنَّ إبراهيمَ الخليلَ قال: يا ربِّ! ممَّن الدَّاء؟ قال: منِّي. قال: فممَّنْ الدَّوَاءُ؟ قال: منِّي. قال: فَمَا بَالُ الطَّبِيبِ؟ قال:رَجُلٌ أَرْسَلَ الدَّوَاءَ عَلَى يَدَيْهِ))

وفي قوله صلى الله عليه وسلم: ((لكلِّ داءٍ دواء))، تقويةُ نفس المريض والطبيب، وحثٌّ على طلب ذلك الدواء والتفتيش عليه، فإنَّ المريض إذا استشعرت نفسُه أن لدائه دواءً يُزيله، تعلَّق قلبُه بروح الرجاء، وبَرَدت عنده حرارة اليأس، وانفتَحَ له بابُ الرجاء، ومتى قويتْ نفسُه انبعثتْ حرارتُه الغريزية، وكان ذلك سبباً لقوة الأرواح الحيوانية والنفسانية والطبيعية، ومتى قويتْ هذه الأرواح، قويت القُوَى التي هي حاملةٌ لها، فقهرت المرضَ ودفعتْه.وكذلك الطبيبُ إذا علم أنَّ لهذا الداء دواءً أمكنه طلبُه والتفتيشُ عليه. وأمراض الأبدان على وِزَانِ أمراض القلوب، وما جعل الله للقلب مرضاً إلا جعل له شفاءً بضده، فإنَّ علمه صاحبُ الداء واستعمله، وصادف داءَ قلبه، أبرأه بإذن الله تعالى.

فصل

في هَدْيه صلى الله عليه وسلم في الاحتماء من التخم، والزيادة في الأكل على قدر الحاجة، والقانون الذي ينبغي مراعاتُه في الأكل والشرب

في ((المسند)) وغيره: عنه صلى الله عليه وسلم أنه قال: ((ما مَلأ آدميٌّ وِعاءً شرّاً مِنْ بطنٍ، بِحَسْبِ ابنِ آدمَ أُكَيْماتٌ يُقِمْنَ صُلْبَه، فإنْ كان لا بُدَّ فاعلاً، فَثُلُثٌ لِطَعَامِهِ، وثُلُثٌ لِشَرَابِهِ، وثُلُثٌ لِنَفَسِهِ)).

الأمراض نوعان: أمراضٌ مادية تكون عن زيادة مادة أفرطتْ في البدن حتى أضرَّتْ بأفعاله الطبيعية، وهي الأمراض الأكثرية، وسببها إدخالُ الطعام على البدن قبل هضم الأوَّل، والزيادةُ في القدر الذي تحتاج إليه البدن، وتناولُ الأغذية القليلة الدفع، البطيئة الهضم، والإكثارُ من الأغذية المختلفة التراكيب المتنوعة، فإذا ملأ الآدميُّ بطنه من هذه الأغذية، واعتاد ذلك، أورثته أمراضاً متنوعة، منها بطيءُ الزوال وسريعُه، فإذا توسَّط في الغذاء، وتناول منه قدرَ الحاجة، وكان معتدلاً في كميته وكيفيته، كان انتفاعُ البدن به أكثرَ من انتفاعه بالغذاء الكثير

Figure 1. One image of the Discrete Simplified Arabic dataset.

Figure 2. One image of the Discrete Traditional Arabic dataset.

Figure 3. Line segmentation of the image of Figure 1.

فكان من هَدْيه صلى الله عليه وسلم فعلُ التداوي في نفسه، والأمرُ به لمن أصابه مرض من أهله وأصحابه، ولكن لم يكن من هَدْيه ولا هَدْي أصحابه استعمالُ هذه الأدوية المركَّبة التي تسمى ((أقرباذين))، بل كان غالبُ أدويتهم بالمفردات، وربما أضافوا إلى المفرد ما يعاونه، أو يُكسر سَوْرَته، وهذا غالبُ طبِّ الأمم على اختلاف أجناسها من العرب والتُّرك، وأهل البوادي قاطبةً، وإنما عُني بالمركبات الرومُ واليونانيون، وأكثرُ طبِّ الهند بالمفردات

وقد اتفق الأطباء على أنه متى أمكن التداوي بالغذاء لا يُعْدَل عنه إلى الدواء، ومتى أمكن بالبسيط لا يُعْدَل عنه إلى المركَّب.

قالوا: وكل داء قدر على دفعه بالأغذية والحمية، لم يُحاوَلْ دفعه بالأدوية.قالوا: ولا ينبغي للطبيب أن يولع بسقي الأدوية، فإنَّ الدواء إذا لم يجد في البدن داءً يُحلِّله، أو وجد داءً لا يُوافقه، أو وجد ما يُوافقه فرادت كميتهُ عليه، أو كيفيته، تشبَّث بالصحة، وعبث بها، وأرادتْ التجارب من الأطباء طبُّهم بالمفردات غالبًا، وهم أحد فرَقِ الطبِّ الثلاث.

والتحقيقُ في ذلك أن الأدوية من جنس الأغذية، فالأمة والطائفة التي غالبُ أغذيتها المفردات، أمراضُها قليلة جدًا، وطبُّها بالمفردات، وأهلُ المدن الذين غلبتْ عليهم الأغذيةُ المركَّبة يحتاجون إلى الأدوية المركَّبة، وسببُ ذلك أنَّ أمراضهم في الغالب مركَّبةٌ، فالأدويةُ المركَّبة أنفعُ لها، وأمراضُ أهل البوادي والصحاري مفردة، ويكفي في مداواتها الأدويةُ المفردة. فهذا برهانٌ بحسب الصناعة الطبية.

ونحن نقول: إن ههنا أمرًا آخرَ، نسبةُ طب الأطبَّاء إليه كنسبة طبِّ الطُّرُقية والعجائز إلى طبهم، وقد اعترف به حُذَّاقهم وأئمتهم، فإنَّ ما عندهم من العلم بالطِّب منهم مَن يقول: هو قياس. ومنهم مَن يقول: هو تجربة. ومنهم مَن يقول: هو إلهامات، ومنامات، وحَدْسٌ صائب. ومنهم مَن يقول: أخذ كثير منه من الحيوانات البهيمية، كما نشاهد السنانير إذا أكلت دوات السموم تَعْمِدُ إلى البَرَاح، فتَلغ في الزيت تتداوى به، وكما رؤيت الحيَّات إذا خرجت من بطون الأرض، وقد عَشِيت أبصارُها تأتي إلى وري الرازيانج، فتُمرِّغ عيونها عليها. وكما شُهد من الطير الذي يحتقن بماء البحر عند انحباس طبعه، وأمثال ذلك مما ذُكِرَ في مبادئ الطب.

وأين يقع هذا وأمثالُه من الوحي الذي أُوحيه الله إلى رسوله بما ينفعه ويضره، فنسبة ما عندهم من الطب إلى هذا الوحي كنسبة ما عندهم من العلوم إلى ما جاءت به الأنبياء، بل ههنا من الأدوية التي تُشفي من الأمراض ما لم يهتد إليها عقولُ أكابر الأطباء، ولم تصل إليها عُلومُهم وتجاربهم وأقيستهم، من الأدوية القلبية، والروحانية، وقوة القلب، واعتماده على الله، والتوكل عليه، والالتجاء إليه، والانطراح والانكسار بين يديه، والتذلُّل له، والصدقة، والدعاء، والتوبة، والاستغفار، والإحسان إلى الخلق، وإغاثة الملهوف، والتفريج عن المكروب، فإنَّ هذه الأدوية قد جَرَّبتها الأممُ على اختلاف أديانها ومللها، فوجدوا لها من التأثير في الشفاء ما لا يصل إليه علمُ أعلم الأطباء، ولا تجربته، ولا قياسُه.

وقد حرَّرنا نحن وعرفنا من هذا أمورًا كثيرةً، ورأيناها تفعل ما لا تفعل الأدويةُ الحسِّيَّة، بل تُصيِّر الأدوية الحسِّيَّة عندها بمنزلة الأدوية الطُّرُقية عند الأطباء، وهذا جارٍ على قانون الحكمة الإلهية ليس خارجًا عنها، ولكن الأسباب متنوعة، فإن القلبَ متى اتصل برب العالمين، وخالق الداء والدواء، ومدبِّر الطبيعة ومُصرِّفها على ما يشاء كانت له أدويةٌ أخرى عن الأدوية التي يُعالجها الغالبُ البعيدُ منه المُعْرِضُ عنه، وقد عُلِمَ أنَّ الأرواحَ متى قويت، وقويتْ النفسُ والطبيعةُ تعاونا على دفع الداء وقهره، فكيف يُنكر لمن قويت طبيعتُه ونفسُه، وقويت بقُربها من بارئها، وأنسها به، وحبِّها له، وتنعُّمها بذكره، وانصراف قواها كُلِّها إليه، وجَمْعها عليه، واستعانتها به، وتوكلها عليه، أن يكونَ ذلك لها من أكبر الأدوية، وأن توجب لها هذه القوةُ دفعَ الألم بالكلية، ولا يُنكرُ هذا إلا أجهلُ الناس، وأغلظهم حجابًا، وأكثفُهم نفسًا، وأبعدُهم عن الله وعن حقيقة الإنسانية، وسنذكر إن شاء الله السببَ الذي به أزالتْ قراءةُ الفاتحة داءَ اللَّدِيغ التي رُقي لها، فقام حتى كأنَّ ما به قَلَبة.

فهذان نوعان من الطب النبوي، نحن نَجْزِل الله نتكلم عليهما بحسب الجهد والطاقة، ومبلغ علومنا القاصرة، ومعارفنا المتلاشية جدًا، وبضاعتنا المُزْجاة، ولكنَّا نستوهبُ مَن بيده الخزائن كلَّها، ونستمد من فضله، فإنه العزيز الوهَّاب.

فصل

في الأحاديث التي تحث على التداوي وربط الأسباب بالمسببات

روى مسلم في ((صحيحه)): من حديث أبي الزُّبَيْر، عن جابر بن عبد الله، عن النبيِّ صلى الله عليه وسلم أنه قال: ((لكلِّ داء دواء، فإذا أُصيبَ دَوَاءُ الدَّاءِ، برأ بإذن الله عَزَّ وجَلَّ)).

وفي ((الصحيحين)): عن عطاء، عن أبي هريرة قال: قال رسول الله صلى الله عليه وسلم: ((ما أنزل الله مِنْ داء إلا أَنْزَلَ لَهُ شِفَاءً)).

وفي ((مسند الإمام أحمد)): من حديث زياد بن علاقة عن أسامةَ ابن شريك، قال: ((كنتُ عندَ النبيِّ صلى الله عليه وسلم، وجاءت الأعرابُ، فقالوا: يا رسول الله؛ أَنَتَدَاوَى ؟ فقال:

((نَعَمْ يا عبادَ الله تَدَاوَوْا، فإنَّ الله عَزَّ وجَلَّ لم يضَعْ داءً إلا وَضَعَ لَهُ شِفَاءً غيرَ داء واحد))، قالوا: ما هو ؟ قال: ((الهَرَمُ)).

وفي لفظ: ((إنَّ الله لم يُنْزِلْ دَاءً إلا أنزل له شِفَاءً، عَلِمَهُ مَنْ عَلِمَهُ وجَهِلَهُ مَنْ جَهِلَهُ)).

وفي ((المسند)): من حديث ابن مسعود يرفعه: ((إنَّ الله عَزَّ وجَلَّ لم يُنْزِلْ داءً إلا أنزَلَ لَهُ شِفاءً، عَلِمَهُ مَنْ عَلِمَهُ، وَجَهِلَهُ مَنْ جَهِلَهُ)).

وفي ((المسند)) و((السنن)): عن أبي خِزَامة، قال: قلتُ: يا رسول الله؛ أرأيتَ رُقًى نَسْتَرْقِيهَا، ودواء نتداوى به، وتُقَاةً نَتَّقِيهَا، هل تَرُدُّ من قَدَرِ الله شيئًا ؟ فقال: ((هي من قَدَرِ الله)).

فقد تضمَّنت هذه الأحاديثُ إثبات الأسباب والمسبَّبات، وإبطالَ قولِ مَن أنكرها، ويجوزُ أن يكون قوله((لكلِّ داء دواء))، على عمومه حتى يتناول الأدواءَ القاتلة، والأدواء التي لا يُمكن الطبيب أن يُبرئها، ويكون الله عَزَّ وجَلَّ قد جعل لها أدوية تُبرئها، ولكن طَوَى علمَها عن البَشر، ولم يجعل لهم إليه سبيلًا، لأنه لا علم للحاق إلا ما علَّمهم الله، وهذا علَّق النبيُّ صلى الله عليه وسلم الشِّفاءَ على مصادفة الدواء للداء، فإنه لا شيء من المخلوقات إلا له ضدٌّ، وكلُّ داء له ضد من الدواء يعالَج بضدِّه، فعلَّق النبيُّ صلى الله عليه وسلم البُرءَ بموافقة الداء للدواء، وهذا قدرٌ زائدٌ على مجرد وجوده، فإنَّ الدواء متى جاوز درجة الداء في الكيفية، أو زاد في الكمية على ما ينبغي، نَفَلَه إلى داء آخر، ومتى قصر عنها لم يَفِ بمقاومته، وكان العلاج قاصرًا، ومتى لم يقع الدواء موقع الداء، أو لم يقع الدواء على الداء، لم يحصِّل الشفاء، ومتى لم يكن الزمان صالحًا لذلك الدواء، لم ينفع، ومتى كان البدنُ غيرَ قابل له، أو القوةُ عاجزةً عن حمله، أو ثَمَّ مانعٌ يمنع من تأثيره، لم يحصل البُرء لعدم المصادفة، ومتى تمت المصادفة حصلَ البرءُ بإذن الله ولا بُدَّ، وهذا أحسنُ المجملَيْن في الحديث.

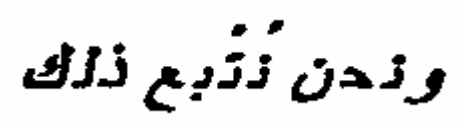
والثاني: أن يكون من العام المراد به الخاصُّ، لا سيما والداخل في اللَّفظ أضعاف أضعاف الخارج منه، وهذا يُستعمل في كل لسان، ويكونُ المراد أنَّ الله لم يضع داءً يَقْبَلُ الدواء إلا وضع له دواء، فلا يَدخل في هذا الأدواء التي لا تقبل الدواء، وهذا كقوله تعالى في الرِّيح التي سلَّطها على قوم عاد: ﴿تُدَمِّرُ كُلَّ شَيْءٍ بِأَمْرِ رَبِّهَا﴾[الأحقاف: ٢٥] أي: كل شيء يقبلُ التدمير، ومن شأن الرِّيح أن تدمِّره، ونظائرُه كثيرة.

ومَن تأمَّل خلقَ الأضداد في هذا العالَم، ومقاومةَ بعضها لبعض، ودفعَ بعضها ببعض، وتسليطَ بعضها على بعض، تبيَّن له كمالُ قدرة الرب تعالى، وحكمتُه، وإتقانُه ما صنعه، وتفرُّدُه بالربوبية، والوحدانية، والقهر، وأنَّ كل ما سواه فله ما يُضاده ويُنافره، كما أنه الغنيُّ بذاته، وكلُّ ما سواه محتاجٌ بذاته.

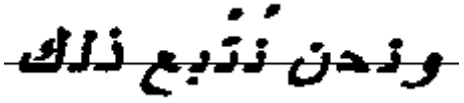
وفي الأحاديث الصحيحة الأمرُ بالتداوي، وأنه لا يُنافي التوكل، كما لا يُنافيه دفعُ داء الجوع، والعطش، والحرِّ، والبرد بأضدادها، بل لا تتم حقيقةُ التوحيد إلا بمباشرة الأسباب التي نَصَبَها الله مقتضياتٍ لمسبَّباتها قدرًا وشرعًا، وأن تعطيلها يقدَحُ في نفس التوكل، كما يَقْدَحُ في الأمر والحكمة، ويضعفه من حيث يظن مُعطِّلُها أنَّ تركها أقوى في التوكل، فإن تركها عجزًا يُنافي التوكلَ الذي حقيقتُه اعتمادُ القلب على الله في حصول ما ينفع العبد في دينه ودنياه، ودفعِ ما يضرُّه في دينه ودنياه، ولا بد مع هذا الاعتماد من مباشرة الأسباب؛ وإلا كان معطِّلًا للحكمة والشرع، فلا يجعل العبدُ عجزه توكلًا، ولا توكُّله عجزًا.

وفيها رد على مَن أنكر التداوي، وقال: إن كان الشفاء قد قُدِّر، فالتداوي لا يفيد، وإن لم يكن قد قُدِّر، فكذلك. وأيضًا، فإنَّ المرض حصل بقَدَر الله، وقَدَرُ الله لا يُدْفَع ولا يُرد، وهذا السؤال هو الذي أورده الأعراب على رسول الله صلى الله عليه وسلم. وأما أفاضلُ الصحابة، فأعلَمُ بالله وحكمته وصفاته من أن يُوردوا مثلَ هذا، وقد أجابهم

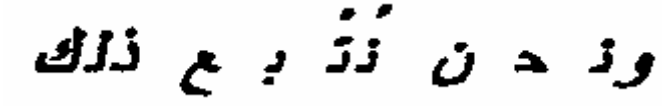
Figure 4. Line segmentation of the image of Figure 2.



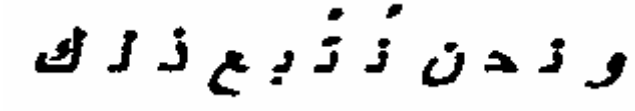
(a)



(b)



(c)



(d)

Figure 5. Character formation: (a) A portion of a line of discrete Arabic script, (b) the baseline determined, (c) the line segmented into seven white cut components, and (d) white cut components segmented into connected components that are combined to form final eleven characters.

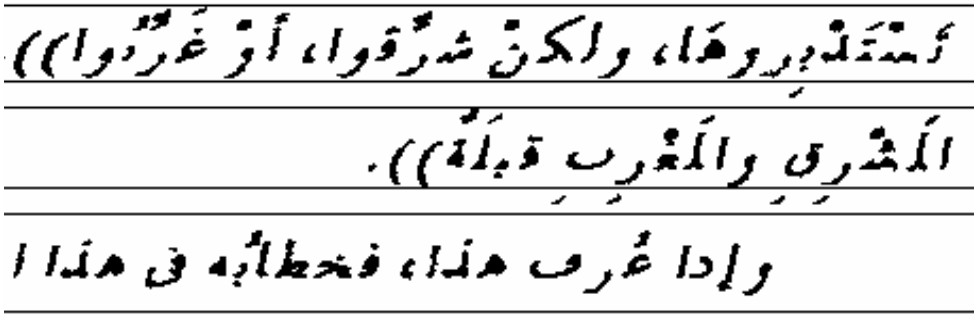


Figure 6. Missed diacritics: The under-line diacritics between the second and third solid lines and between the forth and fifth solid lines couldn't be grouped to any line.

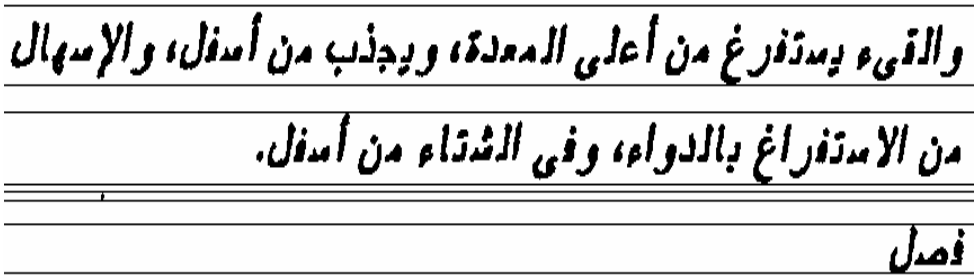


Figure 7. Part of an image, where an extra line is generated due to blobs. The extra line is located between the third and forth horizontal solid lines, starting from the bottom of the image, and consists of a single black dot.

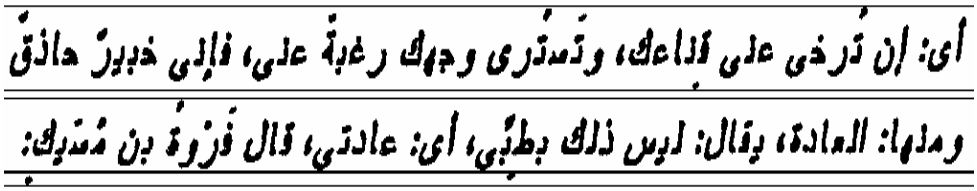


Figure 8. Part of an image, where an extra line is generated. The extra line is located between the first and second horizontal solid lines, starting from the bottom of the image, and consists only of under-line diacritics.

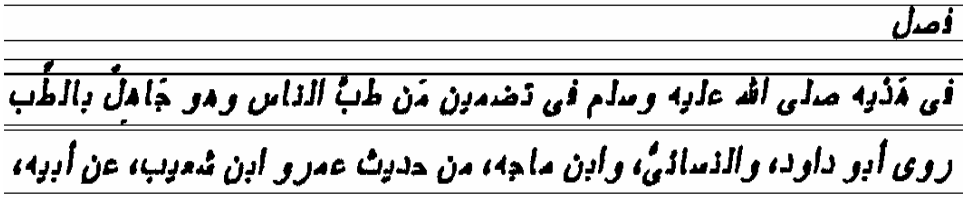


Figure 9. Part of an image, where an extra line is generated. The extra line is located between the third and forth horizontal solid lines, starting from the top of the image, and consists only of above-line diacritics.

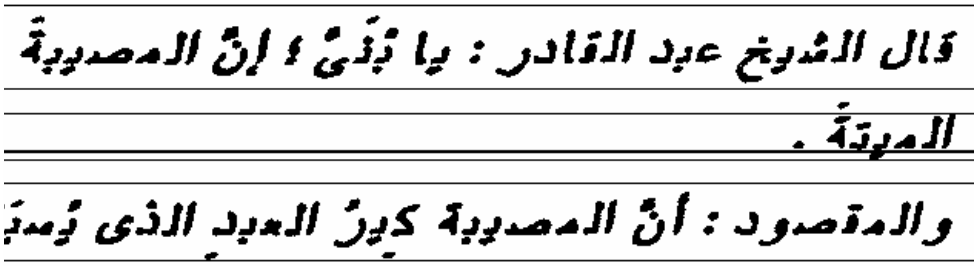


Figure 10. Part of an image, where an extra line is generated. The extra line is located between the third and forth horizontal solid lines, starting from the bottom of the image, and consists of under-line two connected dots.

(a) (b) (c) (d)

Figure 11. Examples of character segmentation errors: (a) A basic component assigned a secondary component of a previous basic component, (b) A basic component assigned a secondary component from a following line, (c) A secondary component classified as a basic component, and (d) Two connected basic components.

Table (I): Results of the Line Segmentation Algorithm.
A: Actual no. of lines, C: Computed no. of lines, *LS*: Line Success rate, E: Average no. of extra lines per page, 1: Percentage of category (1) extra lines containing pepper noise, black extraneous dots, or blobs, 2: percentage of category (2) extra lines containing underline diacritics, 3: percentage of category (3) extra lines containing above-line diacritics, 4: percentage of category (4) extra lines containing underline original dots, and 2 & 3 & 4: percentage of categories (2, 3, 4) extra lines.

Font	A	C	*LS* %	E	Percentage of Error Category				
					1	2	3	4	2 & 3 & 4
Simplified	4495	4620	97.3	1.42	64.0	15.2	20.0	0.8	36.0
Traditional	3897	3960	98.4	0.91	1.6	77.8	20.6	0.0	98.4
Both	8392	8580	97.8	1.20	43.1	36.2	20.2	0.5	56.9

Table (II). Errors of Character Segmentation.

Reason	Simplified		Traditional		Both	
	No. of Errors	Percentage of Errors	No. of Errors	Percentage of Errors	No. of Errors	Percentage of Errors
1	103	75.7	122	81.9	225	78.9
2	8	5.9	0	0.0	8	2.8
3	20	14.7	27	18.1	47	16.5
4	5	3.7	0	0.0	5	1.8
All	136	100	149	100	285	100