A Graph-Based Segmentation and Feature Extraction Framework for Arabic Text Recognition

Ahmed M. Elgammal
Department of Computer Science
University of Maryland
College Park, MD 20742, USA

Mohamed A. Ismail
Department of Computer Science
Faculty of Engineering
University of Alexandria, Egypt

Abstract

This paper presents a graph-based framework for the segmentation of Arabic text. The same framework is used to extract font independent structural features from the text that are used in the recognition. The major contribution of this paper is a new graph-based structural segmentation approach based on the topological relation between the baseline and the line adjacency graph (LAG) representation of the text. The text is segmented to sub-character units that we call "scripts". A structure analysis approach is used for recognition of these units. A different classifier is used to recognize dots and diacritic signs. The final character recognition is achieved by using a regular grammar that describes how characters are composed from scripts.

1. Introduction

The cursive nature of the Arabic text is the main obstacle for any Arabic OCR system. Mostly, characters are connected to each other in what is called "the base line" to form subwords. Some characters might connect to each other above the base line as in ligatures. Researchers have paid special attention to this problem and developed many algorithms to segment Arabic text into characters. Three different approaches can be traced in the literature to handle the cursive nature of Arabic text: explicit segmentation, implicit segmentation and segmentation free approaches. In [5, 4, 2, 12, 16, 10, 15, 8, 3] explicit segmentation was used where a separate phase in the text recognition process is dedicated to segment subwords into smaller units that can be individually recognized. Implicit segmentation was used by [7, 6] where character are segmented while being recognized. Segmentation free approach for Arabic word recognition was used by [1]

This paper presents a graph-based framework for the segmentation of Arabic text. The same framework is used to extract font independent structural features from the text that are used in the recognition. The approach is based on representing the text using *line adjacency graph* (LAG) [14]. The major contribution of this paper is the use of this representation for the *segmentation* of cursive text. The segmentation is achieved by considering the relation between the text baseline and this graph. The approach can successfully handle situations where characters are overlapping vertically that cause problems to many existing segmentation techniques.

The text is segmented to what we call scripts. A script is considered to be the unit of recognition and it does not necessarily correspond to characters. A script may be a complete character, it may be a part of a character or it may be more than one character (as in ligatures.) The graph representing the text is used to extract structural shape features such as strokes, loops and feature points that is used in the recognition. Two different classifiers are used. A classifier for the scripts and a classifier for the dots and diacritic signs. The results of the two classifiers are combined together using some linguistic rules and the final recognition result are obtained using a regular grammar describing the formation of the characters from the basic scripts.

The layout of the paper is as follows: Section 2 describes the process of script extraction (segmentation.) Section 3 describes the script classifier and the dot and diacritic classifier. Final character recognition process is described in section 3.2. Experiments and Error analysis are summarized in section 4.

2. Script Segmentation

The segmentation process extracts the basic scripts from the textline. By a script we do not mean a character. A script is considered to be the unit of recognition. A script may be a complete character, it may be a part of a character or it may be more than one character (as in ligatures). The segmentation of a textline into its scripts is based on the topological relation between the textline, represented as a graph, and the baseline of the text. This process is described in more details in the next two subsections.

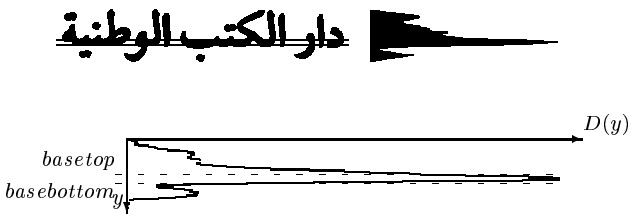

Figure 1. Top: A text line and its horizontal projection profile. Bottom: Baseline detection

2.1. Baseline Detection

Baseline plays an essential role on Arabic writing. Most characters connect to each other on the baseline. The baseline detection is done by detecting the peak in the the horizontal density histogram of the text line. Fig. 1-top, shows an Arabic text line and its histogram of horizontal densities, called the *horizontal projection profile*. As can be noticed from the figure, the peak in the histogram corresponds to the baseline in the text. The global peak of the histogram is detected. The ratio of this peak to the width of the text line should exceed certain threshold α to be considered as a baseline. The base line is parameterized by two values, *basetop* and *basebottom*, which represent the top row and the bottom row of the base line. These two values are set such that a certain percentage, t, of the black pixels in the textline is included between these two rows and the height of the baseline is minimized. The ratio t as well as the threshold α are pre-trained based on sample data and their values may vary slightly without significant effect on the next phases. Fig. 1 illustrates baseline detection.

2.2. Subword and Script Extraction

Each text line is represented by a *line adjacency graph* (LAG) [14]. The LAG is a graph consisting of nodes representing horizontal runlengths. Any two runlengths lying on adjacent scan lines and overlapping with each other have an edge connecting their corresponding nodes. The process of building a LAG representation for a text line yields a set of isolated subgraphs (connected components), in which every subgraph of is the LAG representing a subword of the text line. Therefore the process of word isolation is done at the same time the LAG is constructed. A connected component of the LAG may corresponds to a dot combination, a diacritic, an isolated character or a subword consisting of several characters. Each LAG component is classified into one of three categories; components that intersect with the base line, i.e., subwords, and components that are either above or below the baseline. The last two categories are candidates to be dot combinations or diacritics and are handled as will be described in section 3.1.

The LAG for each subword is then transformed into another graph which is homomorphic to the LAG and has minimum number of nodes. The new graph is called *compressed* LAG, or *c*-LAG [14]. The nodes of the new graph are labeled as "path" or "junction". Obviously path nodes are never adjacent to each other but junction nodes may be adjacent to each other. Fig. 2 illustrate a subword with its corresponding LAG and *c*-LAG.

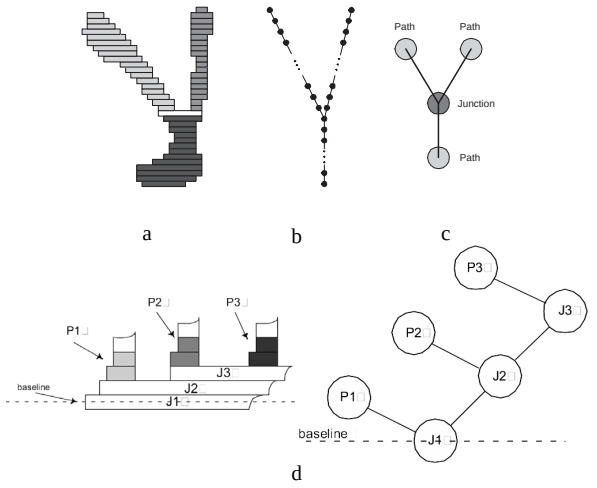

Figure 2. (a) a subword with each black run-length is presented by a rectangle. (b) The LAG representation. (c) The compressed LAG. (d) a configuration that needs the application of **rule 1** and **rule 2** to get the correct baseline relation.

The relation between the *c*-LAG nodes and the baseline is very important in feature extraction phase and in script extraction. Each node of the graph is labeled with one of the following labels to reflect its relation with the baseline.

1. Above the baseline.
2. Below the baseline.
3. Inside the baseline.
4. Above the baseline and connected with it.
5. Below the baseline and connected with it.
6. Crossing the baseline.

Any path node may be assigned any of these labels. Junction nodes can only be assigned any of the first three labels (*inside, above or below* the baseline) since they have a height of one. The following two rules are applied to the graph node labeling. By applying these two rules we can overcome the problem of the uncertainty in baseline detection due to font variation or due to slanting in the image or due to any other factors that may affect the baseline detection.

Rule 1 : A junction node that is adjacent to another junction node inside the baseline have to be labeled as *inside* the baseline even though it may not be physically located between *basetop,basebottom*. This rule should be applied recursively. In Fig. 2-d J_2, J_3 will both be labeled as *inside*

since they are adjacent to J_1 although they are not actually located inside the baseline.

Rule 2 : If a path node is adjacent to a junction node that has been labeled as *inside*, this path node is to be labeled as *connected* to the baseline. This rule is illustrated in Fig. 2-d. P_1, P_2 and P_3 are labeled as *above and connected* even though they are not actually connected with the baseline. This rule is not recursive but must be applied after applying rule 1.

By a script we mean a subgraph that is a unit for recognition. A script may be a complete character, it may be a part of a character or it may be more than one character (as in ligatures). To find subgraphs corresponding to scripts, first the *c*-LAG is traversed starting from any path node that is above the baseline until we reach a junction node that is labeled as inside the baseline. The traversed nodes constitute a subgraph that we call a script. The dimension of a script is the maximum dimension of its nodes. As we can see from the stopping criteria in this graph traversal algorithm, a Junction node labeled as inside the baseline is the break point between scripts. This criteria matches with the basic characteristic of connecting characters in Arabic writing where characters are connected in the baseline area. Such nodes will not be considered as part of any of the resulting scripts. The process of *c*-LAG traversal is continued until all scripts are found.

Another rule that is applied when extracting scripts from the *c*-LAG is: a loop must be contained inside a script and should not be segmented between different scripts. So, given a loop, all its nodes are considered to be a part of a single script. The *c*-LAG is traversed starting from its path nodes until a break point is reached, as mentioned above, to find the entire script.

The resulting scripts are classified to two sets: basic scripts B and non-basic scripts N. A script is called a basic script if the whole script or a part of it is located above the baseline. The majority of Arabic characters contains a part of them above the baseline and so the basic scripts are considered as the seeds of the characters. Any other script that does not satisfy the above condition is considered to be a non-basic script. Every non-basic script is either a part of a character below the baseline that has another part of it above the baseline or a character that is located completely below the detected baseline. In the first case, we have to associate the non-basic script with its corresponding basic one to form a new, larger basic script. The process goes like that: for every non-basic script, n, the basic script set, B, is searched to find the best one to associate n with. If no such basic script is found, n itself is considered to be a basic script and it is included in B instead of N. After this association process, we only have one set of script each of them is a candidate to be a character. Fig. 3-a illustrates the scripts of a given text line.

3. Script Classification

To develop a font and size independent character recognition system, a set of shape features should be identified that could describe the different alphabet symbols independently from the font and the size. The selected set of shape features should have discriminating capability and also should be robust to changes in font and size. The following set of structural descriptors are used for feature extraction:

- Strokes representing the skeleton of the script.
- Holes in the script. Some Arabic characters have one hole while some other characters have two holes.
- Feature points: Local minimum and maximum in the counter of the script and endpoints in the vertical direction (*top endpoint* and *bottom end point*).
- Script bounding box: The location of the script bounding box with respect to the baseline and its size compared to the height of the text line are powerful features. line and ends at the bottom of the text line.

These structural descriptors can be extracted by traversing the *c*-LAG of each subword. Strokes representing the skeleton of the script can be found by analyzing path nodes of the *c*-LAG [14]. Fig. 3-b illustrates a text line and the strokes, feature points and loops of its scripts.

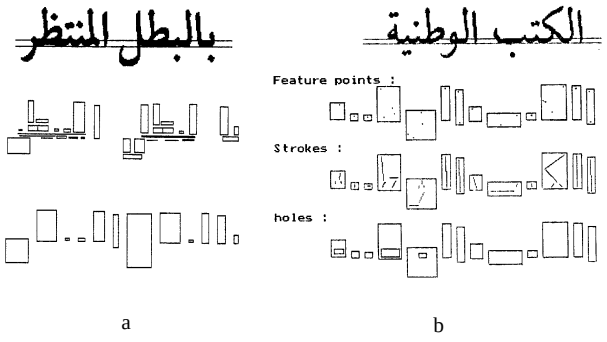

Figure 3. (a) Top: the original text line. middle: path nodes with rectangles representing their spatial dimension. Bottom: extracted scripts. (b) Structure Descriptors

For feature identification, we use an approach similar to that proposed by Kahan and Pavalidis [11]. The objective of feature identification process is to map the structural descriptors representing each script into a set of binary features. The first step in this method is to transform the original parameters describing each shape into new ones in such a way that similarity between shapes can be modelled by a metric in a new parameter space. Once this is done, a binary feature is associated with each feature defining region (cluster) in the parameter space as follows: the feature will take

the value one when at least one shape lies in the associated region in the parameter space and zero otherwise. These "feature-defining" regions can be selected by an automatic clustering algorithm.

The details of shape parameterization are generally different for each shape-type. The transformation from the original shape space into the parameter space is performed by evaluating the shape dimensions locally with respect to the bounding box of the script containing this shape. The diagonal of the script bounding box is considered as reference for length normalization. Also the location and dimension of the script bounding box is considered as a shape feature in this model.

Clustering is used to find clusters of each shape in each parameter space. About 7000 strokes, 9000 feature points and 500 loops are extracted from a training set of pages and are used in clustering experiments. The result of the clustering is a set of clusters' mean and variance. Each cluster is approximated by a hyper-rectangular region centered at the cluster mean. As mentioned before, each cluster (region) is assigned a binary feature variable which is set to one when any input shape falls within its neighborhood. As a result of this process, each script is represented by a binary feature vector. Statistical Bayesian classifier using binary features [13] is used to classify the scripts. Figure 5-a illustrates a block diagram for the whole script classification process.

3.1. Dot and Diacritic Classifier

Dots and diacritics play a very important role in reading Arabic text. Scripts recognized in the previous phase cannot be transformed into real characters without associating them with dots and diacritics above and below them. Although diacritics have no effect on the recognition decision, they must be treated carefully in order not to be misclassified as dot combinations. Recognizing diacritics themselves is important in understanding the text. A word in Arabic can have different meaning with different diacritics above or below its characters.

Figure 4. Recognized Dot a Diacritic Classes

The appearance of dots and diacritics does not vary significantly with font but their appearances are affected very much with printing quality and sampling rates since their size are relatively small. We used a template matching technique to recognize 14 different classes of dots, diacritics and their combinations. First the patterns are normalized to fit into 8x8 gray-scale grid. The resulting patterns are classified using minimum Euclidean distance classifier where class prototype are learned using a sample training set of about 800 dots and diacritics from different font and size environments. Figure 4 shows the 14 classes that can be recognized by the system. Figure 5-b illustrates a block diagram for dot and diacritic classifier.

3.2. Dot Association and Character Recognition

Each classified dot and diacritic pattern is associated with one of the classified scripts based on the topology and linguistic rules. For example, some classes of scripts do not have dots above or below them. Others can only have dots above and/or below them. Since dots and diacritics are sensitive to noise, the best two dot classifier matchings are considered for the association. A classified script associated with its classified dot combination is called a "character candidate".

The mapping from scripts associated with dots (character candidates) to actual Arabic character is not one to one since a script may be a part of a character, a complete character or more than one character. A regular grammar is used in this phase to identify Arabic characters from the stream of character candidates resulting from the association phase. The input to this phase is a string of character candidates. This grammar is a regular grammar, i.e., it can be represented as a finite state automaton and back tracking is not required in parsing character candidate string. Fig 5-c illustrates an example of a parse tree for a word. The number of script classes that are recognized by the system is 38 basic script besides 11 ligatures. After dot association process, 33 character candidates can be identified. The details of the grammar used as well as dot and diacritic association rules can be found in [9]

4. Experimental Results

Experiments have been performed to evaluate the suggested algorithms. In one of the experiments, two sets of pages were used. The first set included 31 pages which did not contain any diacritics while the second set included 15 pages which contained diacritic signs extensively. All the pages were taken from 5 Arabic magazines and contains some Naskh fonts (more than 10 fonts) that are regularly used in printed Arabic documents with size varying from 10 to 16 point size. Non of these pages were used in the training process. All the pages were scanned at resolution 300 dot per inch. The first set gave an average correct script classification rate of 95.2% while the second set gave 94.1%. The performance of dot and diacritic classifier was also tested. Two classifiers were tested: The first one used only for the basic dot combination classification (4 classes: single dot, double dots, triple dots, hamza). The other classifier is used for all the 14 dots and diacritics classes as shown in figure 4. The first set of pages gave an average correct dot classification rate of 97.3% in the 4-class case, and 94.7% in the 14-class case while the second set, which contained diacritics, gave 91.7% using the 14-class classi-

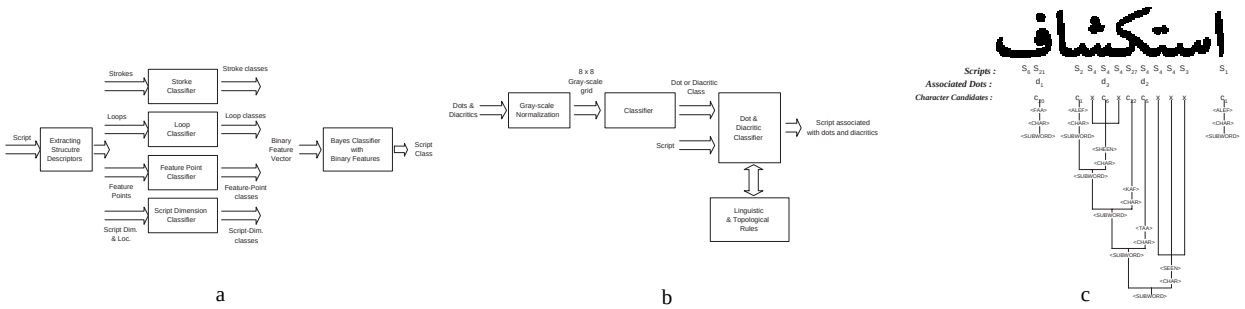

Figure 5. (a) Block Diagram For Script Classifier. (b) Block Diagram For Dot and Diacritic Classification and Association. (c) A word and its parse tree

fier. The final recognition rates were obtained using the 4-class dot classifier for the first set and the 14-class classifier for the second set. The average correct recognition rates were 94.8% for the first set and 88.9% for the second set with average substitution rate of 2.7% and 4.1% for the two sets respectively.

Another experiment was performed to evaluate the algorithms on pages printed using laser printers. For this experiment a data set of pages were prepared using an Arabic word processor using some of the fonts that are available commercially. The pages were printed using a laser printer with resolution 300 dpi (lower quality printing than the previous case but with the same scanning resolution.) The average correct script classification rate was 94.6% and the average correct dot classification rate was 96.6%. The final correct recognition rate was 93.4% with substitution rate of 2.9%.

5. Conclusion

In this paper, a graph-based approach was presented for segmentation and recognition of Arabic text. A novel algorithm for segmentation was presented based on the topological relation between the text, represented by a line adjacency graphs, and the baseline. The resulting segments are classified using font independent structural features that are extracted from the same graph. Another classifier is used for dots and diacritic signs. A regular grammar is used in the final character recognition phase. The system was tested on different groups of pages with different properties. The testing data set contained many Naskh fonts that is used regularly in Arabic printing.

The main cause of errors in script classification was touching characters in irregular positions due to poor quality printing and scanning. We can distinguish between four different types of touching that occur in Arabic text: 1) Dot touching, 2) Ascender touching: character touching above the baseline, 3) Descender touching: touching below baseline and 4) Baseline touching. The algorithm was robust in case of baseline touching while other types of touching caused failure for the segmentation. Future work includes extending the segmentation algorithm to handel these types of touching errors.

References

[1] B. Al-Badr and R. M. Haralick. Segmentation-free word recognition with application to arabic. In *(ICDAR*, 1995.

[2] H. Al-Yousefi and S. Udpa. Recognition of arabic characters. *IEEE PAMI*, 14(8):853–857, Aug. 1992.

[3] H. Almuallim and S. Yamaguchi. A method of recognition of arabic cursive handwiting. *PAMI*, 9(5):715–722, Sept. 1987.

[4] A. Amin. Arabic character recognition. In H. Bunke and P. Wang, editors, *Handbook of Character Recognition and Document Image Analysis*, pages 397–420. World Sceintific Publishing Company, 1997.

[5] A. Amin and J. F. Mari. Machine recognition and correction of printed arabic text. *IEEE Trans. Syst. Man Cybern.*, 19(5):1300–1306, Sept. 1989.

[6] I. Bazzi, R. Schwartz, and J. Makhoul. An omnifont open-vocabulary ocr system for english and arabic. *IEEE PAMI*, 21(6):495–504, June 1999.

[7] S. S. El-Dabi, R. Ramsis, and A. Kamel. Arabic character recognition system: A statistical approach for recognizing cursive typewritten text. *Pattern Recognition*, 23(5):485–495, 1990.

[8] T. El-Sheikh and R. Guindi. Computer recognition of arabic cursive script. *Pattern Recognition*, 21(4):293–302, 1988.

[9] A. Elgammal. Bilingual (arabic/english) document image analysis system with font independent arabic text recognition. Master's thesis, Computer Science Department - Alexandria University, 1996.

[10] A. M. Emam. *Designing a reading machine for the blind*. PhD thesis, University of Alexandria, 1995.

[11] S. Kahan and T. Pavalidis. On the recognition of printed characters of any font and size. *PAMI*, 9(2):274–288, 3 1987.

[12] B. A. Najoua and E. Noureddine. A robust approach for arabic printed character segmentation. In *ICDAR*, pages 865–868, 1995.

[13] R. O.Duda and P. E. Hart. *Patter Classification and Scene Analysis*. Wiley-interscience, 1973.

[14] T. Pavalidis. A vectorizor and feature extractor for document recognition. *Comput. Vision, Graphics, and Image Processing*, 35:111–127, 1986.

[15] K. Romeo-Pakker, H. Miled, and Y.Lecourtier. A new approach for latin/arabic character segmentation. In *ICDAR*, pages 874–877, 1995.

[16] A. Shoukry. A sequential algorithm for the segmentation of typewritten arabic digitized text. *The Arabian Journal for Science and Engineering*, 16(4b):543–556, Oct. 1990.