21st International Conference on Pattern Recognition (ICPR 2012)
November 11-15, 2012. Tsukuba, Japan

Semi-Automated OCR Database Generation for Nabataean Scripts

Adnan Ul-Hasan[1], Syed Saqib Bukhari[1], Sheikh Faisal Rashid[1],
Faisal Shafait[2], Thomas M. Breuel[1]
[1]Technical University of Kaiserslautern, Germany.
{*adnan, bukhari, s_rashid09, tmb*}*@cs.uni-kl.de*
[2]German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany.
faisal.shafait@dfki.de

Abstract

A large amount of real-world data is required to train and benchmark any character recognition algorithm. Developing a page-level ground-truth database for this purpose is overwhelmingly laborious, as it involves a lot of manual efforts to produce a reasonable database that covers all possible words of a language. Moreover, generating such a database for historical (degraded) documents or for a cursive script like Urdu[1] is even more complex and grueling. The presented work attempts to solve this problem by proposing a semi-automated technique for generating ground-truth database. It is believed that the proposed automation will greatly reduce the manual efforts for developing any OCR database. The basic idea is to apply ligature-clustering prior to manual labeling. Two prototype datasets for Urdu script have been developed using the proposed technique and the results are also presented.

1 Introduction

Urdu belongs to the family of cursive scripts where words mainly consist of ligatures. Ligatures are formed by joining individual characters and shape of a character in a ligature depends on its position. Moreover, there are dots and diacritics that are associated with certain characters. Each ligature in Urdu is separated from other ligatures or its own diacritics by vertical, horizontal or diagonal (slanted) space (see Figure 1). In some cases, the diacritic or dot may be surrounded by the main ligature/character (see Figure 1-third character from right).

It is assumed in the context of the present work that the smallest unit is a ligature, which may be either a combination of many characters or a single non joinable character. There are around $18,000$ ligatures in Urdu script[2], and a reasonable database of Urdu script for recognition purposes must cover all ligatures. Other cursive scripts like Arabic, Persian and many Indic scripts share the same characteristics and this is a big hindrance in developing reliable OCR tools for these scripts.

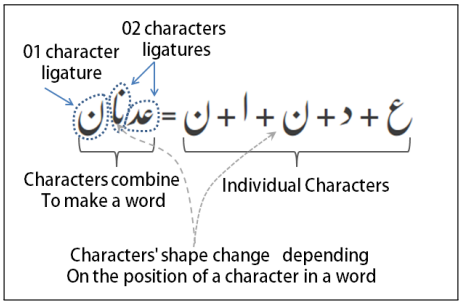


Figure 1: Ligatures are made up of individual characters, but shape of character may change depending upon its location in the ligature. Moreover, spacing between ligatures may be slanted (as between first and second ligature) or vertical (as between second and third ligature).

In literature, many methods have been proposed to overcome the problem of manual labeling process. One approach is to use degradation models [1] on a synthetically produced documents. Another approach is to find the alignment of the transcription of the text lines with the document image. Kanungo et al. [4] presented an automatic methodology for generating character ground truth for scanned document. A document is first created electronically using any typesetting system, printed and scanned. Corresponding feature points from both versions of the same documents were found and then

[1]http://en.wikipedia.org/wiki/Urdu

[2]http://www.crulp.org/software/ling_resources/UrduLigatures.htm

the parameters of the transformation were estimated. The *ideal* ground truth information was transformed accordingly using these estimates. Kim and Kanungo [5] improved this method by presenting a more robust attributed branch-and-bound algorithm. von Beusekom et al. [13] proposed a robust and pixel-accurate alignment method. In a first step, the global transformation parameters were estimated similar to [5], and in the second step, adaptation of smaller regions was carried out.

Pechwitz et al. [7] presented the IfN/ENIT database of handwritten Arabic names of cities along with their postal codes. A projection profile method was used to extract words and the postal codes automatically. Mozaffari et al. [6] developed a similar database (IfN/Farsi-database) for Farsi (Persian) handwritten city names. Sagheer et al. [8] also proposed a similar methodology for generating an Urdu database for handwriting recognition. Slimane et al. [11] developed an Arabic printed text images database. This database was synthetically generated which covers many fonts and font-sizes.

Vamvakas et al. [12] proposed that a character database for historical documents may be constructed by choosing a small subset of images and then using character segmentation and clustering techniques. This work is similar to our approach; however, the main difference is the use of different segmentation technique for Urdu ligatures and use of different clustering algorithm. Due to large shape variations in Urdu ligatures, its not possible to assign a single threshold for all types during word segmentation step. Discussion on threshold selection is given in Section 2.2.

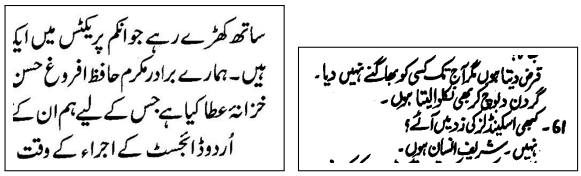
This paper is organized further as follow: Section 2 describes the process of generating database in detail; Section 3 presents the experimental evaluation of the proposed method and concluding remarks are given in Section 4.

2 Method

Starting with Binarization as the pre-processing step, Urdu ligatures are extracted from the text images. These ligatures are then clustered prior to manual labeling of correct ligatures.

2.1 Pre-processing

Binarization is the only pre-processing step in current work; however, skew detection and correction may be included as further pre-processing steps. Local thresholding technique [9] is used for the binarization purpose. Fast implementation of this algorithm proposed by Shafait et al. [10] has been used to speed-up



(a) An example of clean document

(b) A document with narrow line-spacing (written by a calligrapher)

Figure 2: Examples of Urdu scripts in our dataset.

the process. Two parameters, namely local window size and k-parameter, are needed to be set empirically according to the documents. The local window size is taken to be 70×70 and k-parameter is set to 0.3.

2.2 Ligature extraction

Ligature extraction may be carried out in two variations: one is to apply ligature extraction algorithm directly on the binarized image, and the second is to extract text-lines before applying ligature extraction. The former is suitable where documents are clean having well-defined text-line spacing (see Figure 2-(a)) and the latter is suitable when text-lines are not separated very well in the documents (Figure 2-(b)), and in case of degraded historical documents. Narrow line-separation results in poor connected component analysis, thereby leading to many merged ligatures. The decision to apply text-line segmentation is taken on the basis of line-spacing in a particular document.

In the present work, ligature extraction is started by applying connected component analysis. The list of connected components is first divided into two parts: base components and diacritics (including dots). This division is based on connected component's height, width, and their ratio. In the presented work, font variations are not considered and primary focus is to cover the typically used font in Urdu books and magazines. Therefore, thresholds for separating main ligatures and diacritics are set empirically on this font size and they remain same for all document images in our dataset. As mentioned earlier, its not possible to separate Urdu ligatures by a single threshold value. Therefore, different thresholds have been employed as per properties of a particular ligature. For ligature consisting of single اَ, the average height to width ratio is 4.0 and the average width of this ligature is around 6 pixels. For ligatures like ب, ت and ث the average height to width ratio is 0.4 and the average width is around 30 pixels. For all other ligatures, it is sufficient to check for a width greater than 10 pixels. It is not possible to separate

Urdu ligatures based on height to width ratio of connected components.

If there are no diacritics in a ligature, e.g. لو, then no further processing is needed; however, if one or more diacritics are present, e.g. نستعلیق, then these diacritics must be associated to the base component to completely extract a ligature. Diacritics are searched in the neighborhood of a base component by extending the bounding box of the base connected component. This window size depends on the font size; but since we have used only documents with the dominant font size, this window is set according to this font size. Presently, the bounding box of base component is extended by 15 pixels on top and bottom and by 10 pixels on right and left. Extracted ligatures in this manner are then saved to a database file for further clustering and labeling.

2.3 Clustering

As it is already mentioned that huge amount of ligatures in a cursive script makes the labeling of individual ligature highly impractical. It is proposed that the extracted ligatures may be clustered according to similar shapes. Two types of clustering methods are available under OCRopus framework [2], namely k-means and epsilon-net clustering. In the current work, the latter technique is employed. Simply changing the value of epsilon, we can control the number of clusters. The value for epsilon was set empirically to get moderate amount of clusters, so that it can be managed easily at manual step of validation. Feature used for epsilon clustering are bit maps of the ligatures. Moreover, this method is relatively faster than the k-means.

2.4 Ligature Labeling

The next step is to verify the clustering process and if needed, modify clusters manually. The OCRopus framework provides a nice graphical user interface (*ocropus-cedit*) to do this without much hassle. It is also possible that the clustering divides a single ligature in more than one cluster (see Figure 3-(a)), so, one needs to merge different clusters to save time at latter stage of labeling. Moreover, one can also modify the step of dividing a cluster in a way to retain only valid cluster members (same label as that of representative), assign null class to incorrect members and then apply further iterations of clustering on null class.

In the current work, merging of same ligature-clusters preceded the manual labeling and only single cluster iteration is employed. After this verification step, each cluster is examined individually to identify invalid clusters, which are then discarded. Again OCRopus framework can be used for this purpose (see Figure 3-(b)).

At the end of this labeling process, we have a database whose entries indicate the following information about a ligature:

1. Image file name from where this ligature was originally extracted.
2. Bounding box information regarding the location of a ligature in a document image.
3. Unicode string corresponding to the character forming this ligature.

3 Experiments and Evaluation of Results

This section describes the experimental setup and evaluation of results. Two prototype datasets for Urdu script have been developed using the proposed technique. One dataset consists of clean documents such as shown in Figure 2-(a). Presently, 20 such document images have been used. Here, this dataset is referred to as DB-I. The second dataset (referred to as DB-II) consists of 15 documents written by a calligrapher such as shown in Figure 2-(b). An important property of calligraphic documents is that shape of ligature does not remain identical in the documents and there remain minor differences in the ligatures' shapes throughout the document. Ground-truth information about the DB-II is available which is used to gauge the accuracy of line-segmentation algorithm. The importance of choosing these two datasets is to evaluate upper and lower bounds on the performance of the proposed algorithm.

Performance evaluation metric used in present work is ***ligature coverage*** which refers to the number of ligatures in the dataset that are correctly labeled by the clustering step followed by the manual validation step.

The ligature extraction algorithm extracted $16,857$ ligatures from DB-I database. The epsilon-net based clustering, then, clustered these ligatures into 778 clusters. Each individual cluster is then examined to verify the clustering and Unicode values are also assigned to new clusters at this stage. Invalid ligatures are then discarded. The ligature coverage achieved by this process is 82.3%. This high ligature coverage is due to sufficient line spacing and non-touching ligatures.

The inherent difficulty with any connected component analysis based method is the poor accuracy in case of overlapping lines and touching ligatures. To solve the problem of overlapping lines in DB-II dataset, a state-of-the-art line segmentation algorithm [3] has been employed. The segmentation accuracy of this algorithm is over 90%. The second problem of touching ligatures may be improved by using more sophisticated tech-

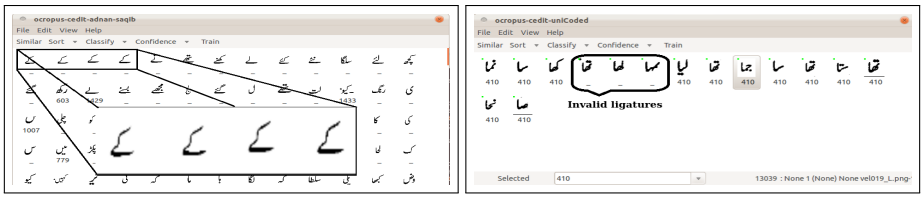


Figure 3: (a) Cluster representatives. A single ligature may be clustered into many clusters. (b)Validating an individual cluster. Invalid ligatures may be discarded by examining a single cluster (represented by "_").

niques. However, we are not interested in fine separation of individual ligatures as the errors may be corrected at latter manual labeling stage. Hence, we did not tackle the problem of touching ligatures in this work.

In the first step, a total of $18,914$ ligatures were extracted. Then the clustering of these ligatures resulted in $1,132$ clusters. After the labeling process, the total ligature coverage is around 62.7%. Inconsistency in ligatures' shape due to calligrapher writing resulted in poor clustering accuracy for DB-II dataset. In this case, simple shape-based clustering methods might not work sufficiently and other methods need to be explored.

As mentioned already, the proposed algorithm has been tested on a prototype database. It is planned to develop a large-scale Urdu dataset for Urdu script using this algorithm. This ground-truth database will be used extensively in future for benchmarking recognition algorithms for Urdu script. Moreover, the proposed methodology may be extended to develop databases for other cursive scripts, e.g. Telegu, Bangla, Arabic, Persian, etc.

4 Conclusions

This paper presented a semi-automated technique to generate ground-truth database for recognition purposes. The basic idea of this algorithm is to cluster similar ligatures prior to manual labeling. This method is quite useful in generating ground-truth databases for scripts having a lot of ligatures, e.g. Nabataean and Indic scripts. OCRopus framework has been used for clustering purpose. This method is tested on a prototype dataset of Urdu script and it resulted in 82.3% coverage for clean documents and 62.7% coverage for documents written by a calligrapher.

References

[1] H. S. Baird. State of the art of Document Image Degradation Modeling. In *DAS* , pages 261–279, Rio de Janeiro, Brazil, Dec. 2000.

[2] T. M. Breuel. The OCRopus open source OCR system. In B. A. Yanikoglu and K. Berkner, editors, *DRR XV*, page 68150, San Jose, California, USA, Jan. 2008.

[3] S. S. Bukhari, F. Shafait, and T. M. Breuel. High Performance Layout Analysis of Arabic and Urdu Document Images. In *ICDAR*, pages 1275–1279, Bejing, China, Sept. 2011.

[4] T. Kanungo and R. M. Haralick. An Auotmatic Closed-loop Methodology for Generating Character Groundtruth for Scanned Documents. *IEEE Trans. on Pattern Anal. Mach. Intell.*, 21(2):179–183, 1999.

[5] D.-W. Kim and T. Kanungo. Attributed Point Matching for Automatic Groundtruth Generation. *Int. Journal on Document Analysis and Recognition*, 5(1):47–66, 2002.

[6] S. Mozaffari, H. Abed, V. Märgner, K. Faez, and A. Amirshahi. IfN/Farsi-Database: A Database of Farsi Handwritten City Names. In *ICFHR*, pages 397–402, Montreal, Canada, Aug. 2008.

[7] M. Pechwitz, S. S. Maddouri, V. Märgner, N. Ellouze, and H. Amiri. IFN/ENIT-Database of Handwritten Arabic Words. In *CIFED* , pages 129–136, Hammamet, Tunis, Oct. 2002.

[8] M. Sagheer, C. He, N. Nobile, and C. Suen. A New Large Urdu Database for Off-Line Handwriting Recognition. In *ICIAP*, pages 538–546, Vietri sul Mare, Italy, Sept. 2009.

[9] J. Sauvola and M. Pietikäinen. Adaptive Document Image Binarization. *Pattern Recognition*, 33:225–236, 2000.

[10] F. Shafait, D. Keysers, and T. M. Breuel. Efficient Implementation of Local Adaptive Thresholding Techniques Using Integral Images. In B. A. Yanikoglu and K. Berkner, editors, *DRR XV*, page 681510, San Jose, California, USA, Jan. 2008.

[11] F. Slimane, R. Ingold, S. Kanoun, A. M. Alimi, and J. Hennebert. A New Arabic Printed Text Image Database and Evaluation Protocols. In *ICDAR*, pages 946–950, Washington, DC, USA, July 2009.

[12] G. Vamvakas, B. Gatos, N. Stamatopoulos, and S. Perantonis. A Complete Optical Character Recognition Methodology for Historical Documents. In *DAS* , pages 525–532, Nara, Japan, Sept. 2008.

[13] J. van Beusekom, F. Shafait, and T. M. Breuel. Automated OCR Ground Truth Generation. In *DAS* , pages 111–117, Nara, Japan, Sept. 2008.