

Shape-based Alphabet for Off-line Arabic Handwriting Recognition

F. MENASRI N. VINCENT
SIP-CRIP5 Universite de Paris 5 (France)
{menasri,vincent}@math-info.univ-paris5.fr

E. AUGUSTIN
A2iA SA - Paris (France)
ea@a2ia.com

M. CHERIET
ETS - Montreal (Canada)
cheriet@etsmtl.ca

Abstract

This article describes an off-line handwritten Arabic words recognition system. Both explicit graphem segmentation and feature extraction are originally designed for Latin cursive handwriting. The recognizer itself is a Hybrid HMM/NN. We introduce a new shape-based alphabet for handwriting Arabic recognition which is intended to benefit from ofsome specificities of Arabic writing.

We performed several experiments using IFN/ENIT benchmark database to validate our approach. Our recognizer performs as close as the state of the art recognition rate with 87%. Those results are very encouraging as many perspectives and improvements may be considered. Especially, the explicit processing of dots and diacritics, therefore making use of more prior knowledge of Arabic writing specificities.

1 Introduction

Although Arabic is spoken by more than 250 million people in the world, a professional automated system has yet to be developed to process offline handwritten Arabic words. The fields of application are numerous : postal automation, check processing, forms processing, automatic reading of ancient Arabic manuscripts, etc... A lot of work has been carried out [6], but automatic processing of handwritten Arabic is still a wide open field of research.

Many articles have highlighted the specific difficulties of Arabic handwriting recognition. First, the cursive nature of Arabic, which leads to a lot of variability between curve angles, shape and size. Second, the shapes of the descenders could also lead to specific problems. Third, Arabic includes many dots and other diacritical marks. Those patterns vary considerably between writers. Fourth, the words are split into subwords, also called Piece of Arabic Words (PAWs). Those PAWs also raise many problems [6]. Fifth, vertical ligatures are not easily segmentable.

Some authors have also rightly quoted how to exploit some of the Arabic writing specificities. Morphological

techniques were proposed, and some of them even tried to recognize words with very large vocabularies [5]. From our point of view, there is more work to be done in this direction. The idea is to try to use more prior knowledge of specificities and constraints of Arabic writing to improve the recognition rate of handwritten Arabic. For this reason, we introduce a shape based alphabet for arabic recognition in the next section.

All experiments and discussions are carried out on IFN/ENIT database. It is further described in section 2.5.

The best two systems of last ICDAR Arabic handwriting competition were very close : 'ARAB-IFN' and 'UOB'. Both are based on HMMs with sliding windows and baseline-dependant feature. 'UOB' has been recently improved to 85% using multiple recognizers combination trained on sliding windows at various angles [4]. 'ARAB-IFN' has been improved up to 89.1% with new feature extraction and improved baseline detection [10]. Recently, Semi-Continuous HMMs with explicit state duration [2] have also given very good results of 89.79% recognition rate. Microsoft Research also provided a recognition system [1], based on PAWs, which achieves 88,94% recognition rate.

Our approach is based on explicit graphem segmentation, and hybrid HMM/NN recognition scheme [3]. We also introduce a shape-based alphabet which is intended to take advantage of arabic writing specificities, especially the redundancy in the shapes of Arabic letters.

First, we will present in details the new alphabet of shapes we use in our recognition system. Then, we will comment upon image preprocessing and then we will briefly describe the recognizer itself. Finally, some comparison will be performed.

2 Letter body alphabet

Alphabets of graphemes designed for printed recognition are presented in [5]. Their authors explain how to build letters or more complex patterns (prefixes and suffixes) explicitly from a set of graphemes. Our goal is not really to define a set of graphemes, as handwriting variability would

make this problem intractable. We still believe, however, that building a new shape alphabet for handwritten Arabic is a good idea for the following reasons :

First, we do not want to train multiple classes to compete against each other when they are supposed to model the same information. Experiments carried out in section 4 justify this approach. Second, if we want to benefit from prior knowledge of Arabic writing, we do not especially need to go straight down to the letter level. For example for the segmentation into PAWs, an intermediate level which would regroup { ر ز } and also regroup { د ذ } would already carry all the information needed.

2.1 A root shape plus a tail

While studying cursive handwriting in Latin alphabet, we never consider that a lowercase ‘c’ takes two different shapes regarding its position (first or not) in a word. When the ‘c’ is not the first letter of a word, we can expect it to have some sort of ligature prior to it. But it’s just the same shape. Saying that the letter ‘c’ has two different shapes depending on the presence or not of the prior ligature is not a usual way to deal with the problem of Latin cursive handwriting recognition.

Saying that Arabic letters can take four different shapes depending on their position in the word (first, middle, last and isolated) is simply not true in general. This statement makes the problem look much more complicated than it actually is. Only three letters take four different shapes :

{ ع ع ع ع } , { غ غ غ غ } , and { ه ه ه ه } .

In addition, the first two sets share the same four shapes, only a dot makes a difference between them. This shape sharing of Arabic letters will also be discussed further in this section.

Arabic letters others than those described above take only two shapes : first/middle and isolated/last. One might also notice that for the majority of Arabic letters, the isolated/last shape is the same as the first/middle shape, followed by some sort of leg (or tail) attached to it. There are roughly three main kinds of tails in Arabic. Only one of those three applies for a given letter (see Table 1).

Tail 1 : ㄥ	Tail2 : ㄣ	Tail 3 : ㄚ
ب ت ث → ㄥ	ل → ㄣ	ح → ㄚ
و → ㄥ	س → ㄣ	ع → ㄚ
	ص → ㄣ	ع → ㄚ
	ز → ㄣ	

Table 1. Arabic letters shape from first/middle to isolated/last

To sum up, most Arabic letters take only one shape, possibly preceded by a ligature (as in Latin cursive handwriting), and possibly followed by one of three kinds of tails (if the letter is last/isolated).

2.2 Shapes which differ only by dots

Another common statement about Arabic writing recognition is that it is a hard task because many letters only differ by the number and position of dots. From this point of view, the task of recognizing Arabic writing may look very difficult. Another way to present things could be : in Arabic writing, many couples of letters share the same shapes and morphological properties (variation in the shape depending on the position of the letter in the word, or the fact that this shape will introduce a break in the word and therefore split it into two PAWs). As a result, we can expect the number of shapes to recognized to be smaller than the number of letters of Arabic alphabet (see Table 2). From this point of view the task seems easier to achieve, as it should be possible to first recognize the sequence of shapes, and then use the dots and diacritic marks to build a sequence of letters from this previously recognized sequence of shapes [7].

{ د ذ } → د	{ ر ب ت ث ز ي } → ر	{ س ش } → س
{ ص ض } → ص	{ ر ز } → ر	{ ح ج ح خ } → ح
{ و ف و } → و	{ ط ظ } → ط	{ ع غ } → ع
{ ع غ } → ع	{ ي ي } → ي	{ ه ه } → ه

Table 2. A few examples of Arabic letters and their corresponding letter-body class

2.3 Vertical ligatures

In Arabic writing, some couples or triplets of letters can be chunked with vertical ligatures (one letter on top of another). The use of those vertical ligatures is up to the writer habit. Segmentation of those complex symbols is not an easy task. Ultimately this question is of little practical significance, as each one of those symbols can be recognized as is. The number of types of vertical ligatures commonly used in everyday’s writing is less than ten (see Figure 1).

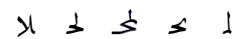


Figure 1. Ligatures shapes (without dots) used in our alphabet

2.4 Our alphabet

With those considerations in mind, we designed a new alphabet of symbols for Arabic writing recognition. We called it *letter-body alphabet*. Dots and diacritics were removed, and letter which share the same shapes were regrouped into one letter-body class. We also added each common vertical ligature as a letter-body class. At this stage, we only took advantage of Tail-1 class 1. Using Tail-2 and Tail-3 would require additionnal segmentation tuning, in order to split the letter from its tail. The list of symbols in our alphabet is given in table 3

ا	ر	ه	ه	ح	ح	د	ر	س	س
ص	ص	ط	ع	ع	ع	و	ك	ل	
ل	م	م	ن	ه	ه	و	ي	Tail 1 : ل	
لا	لا	لا	لا	لا					

Table 3. Complete list of symbols

2.5 Validation on IFN/ENIT database

This section is a theoretical analysis of our approach on IFN/ENIT Database. This database consists of 26,459 images of 937 cities and names of Tunisian towns, written by 411 different writers [9]. It is widely spread as the major database for evaluating Arabic handwriting recognition. It brings not only word level annotation, but also contains information about the shape of each letter in a word.

We can remove the dots and diacritic marks, and then translate each town name from a sequence of letters into our alphabet (a sequence of letter-bodies). We will build and train letter-body HMMs instead of letter HMMs. This translation from a letter sequences vocabulary to a letter-body sequences vocabulary is not a bijection. First, the same Arabic word (sequence of Arabic letters) can be written differently by two writers (use of vertical ligatures or not, or for example the use of *ة* instead of *ت*). As a result, the same Arabic word can be represented by multiple letter-body sequences. But this is not an issue, since the only thing to do is to accumulate the probability of a town name over all the sequences of shapes it can take. Second, the following question may be raised : does a sequence of letter-bodies (word in letter-body vocabulary) belong to one and only one sequence of letters (word in Arabic vocabulary) ? Indeed, if a sequence of letter-bodies matches with two or more distinct words (distinct sequences of characters), there is an ambiguity and we can't draw a final conclusion without examining the dots and diacritic marks. The answer depends on the vocabulary of the given application. On IFN/ENIT database, we found three couples of confusing classes :

classes 7027 (مثلين) and 8140 (متلين) : 45 images

classes 1240 (فريانة) and 8140 (فرنانة) : 15 images

classes 1220 (فوشانة) and 2082 (فوسانة) : 16 images

Those confusions represent a total of 76 images, over a complete database of 26459. Those confusing classes represent 0.3% of the whole database, which is negligible with regard to the current 13% error rate. As a result, for this application, we consider it is safe to use our alphabet to simplify the problem. In order to be solved, some other application with a different vocabulary would probably require the information carried by the dots. In addition, examining the dots and diacritic marks even for this specific problem where it is not mandatory, would probably improve the recognition rate, as it will help to reduce the confusion rate between various classes. There will be further improvement of our system to recognize the dots and diacritics and to redistribute them over the shapes. But for the time being, we can only detect and remove those signs.

In the next section, we will comment upon the pre-processings, and then we will present the recognition system we used, coming along with the letter-body alphabet we just defined in this section.

3 Our recognition system

In this section, we briefly describe our recognition system. It is an HMM/NN hybrid system with explicit grapheme segmentation. We believe that explicit grapheme segmentation is well adapted for Arabic writing. One of the reasons is that some letters such as *ر* or *و* have tails that go almost horizontally under the baseline. If the tail is long, which often happens in Arabic handwriting, the next letter of the word is likely to be vertically overlapping the previous tail. Building a sequence of graphemes intrinsically solves this problem, while, on the other hand, vertical frames or sliding windows will be forced to process a piece of image that contains parts of two different letters at the same time.

3.1 Preprocessing and Segmentation

3.1.1 Baseline extraction

Images from IFN/ENIT Arabic database are already extracted and binarized. The aim of the preprocessing we made was to clean most of the dots and diacritic marks, without damaging the bodies of letters, and then extract the baseline (the baseline is used for feature extraction). We used a slightly modified version of the algorithm proposed by Miled & al [7] to extract the baseline, based on the horizontal projection histogram. The main peak of the histogram is in the baseline, and thresholds are used to extract the upper baseline and the lower baseline around this maximum. horitonztal projection histogram will be disturbed

by many kinds of noises. Among them are the dots and diacritic marks, especially ‘chedda’ or couple of dots represented as a straight horizontal lines. Another problem is the succession of descenders, or long tails under baseline that could lead to a high peak in the histogram under the baseline.

To circumvent those problems, we first coarsely remove the dots and diacritic marks based on the size of the connected components. Loops are often in the baseline. So we detect the loops to prelocate a horizontal band where the projection histogram will be computed. This avoids the problem of considering high peaks under the baseline. After diacritics removal and prelocalization based on loops, we compute the projection histogram and evaluate the baseline (Figure 2). We use this baseline to further clean the remaining diacritics, which are usually located outside the baseline.

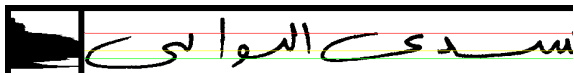


Figure 2. Horizontal histogram to roughly extract the upper and lower baselines

In the final step, we evaluate more precisely the lower baseline, using support points. Those support points are local minimums located in the baseline, and singular points of the skeleton for which one stroke starts inside the baseline and finishes under the baseline (Figure 3).

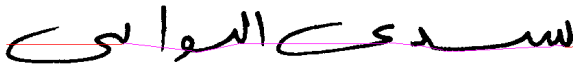


Figure 3. Precise lower baseline using local minimums and specific points of the skeleton

3.1.2 Segmentation

We use a plain generic graphem segmentation designed for Latin cursive writing [3]. We want to avoid under-segmentations : a graphem should be a “piece of ink” that belongs to one and only one letter. In addition, we look forward to split the body of a letter from its tail, accordingly to the letter-body alphabet described in the previous section.

3.2 The recognizer

Seventy-four baseline-dependant features vectors are extracted from the graphems. This feature extraction procedure is not described in this article, but is also the same as the one used for Latin cursive writing [3].

The recognition system is an Hybrid Neural Networks and Hidden Markov Models system (extensively described in [3]). Each letter-body class is represented by an HMM model. The Neural Network computes the observations probability distribution.

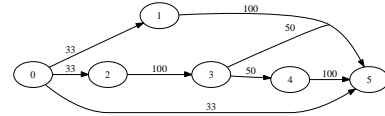


Figure 4. Topology of a Letter-Body HMM. Initial transition probabilities are uniformly distributed

The neural network is a Multi-Layer Perceptron (MLP) with 500 hidden neurons with softmax outputs. To initialize the Neural Network, we first compute a Kmeans over all the feature vectors (observations). Then, we use this Kmeans to annotate the same database of graphems, and we train a Neural Network to produce roughly the same transfert function as the Kmeans does. This Kmeans sets up a first initialization of the Neural Network used in the hybrid system. The Neural Network will then be trained iteratively with the HMMs using the standard Baum/Welch algorithm (see Figure 5). We conducted various experiments where the number of observations classes (each output of the Neural Network corresponds to one observation class) were set to 35, 50, 100, 150, 200 and 250. 100 observation classes with 500 neurons on the hidden layer provided the best results.

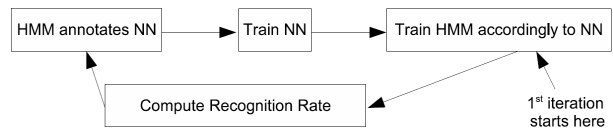


Figure 5. The training of the hybrid system is an iterative procedure [3].

4 Experimental Results

Experiments have been carried out on IFN/ENIT database. This database has a vocabulary of 937 city names. Besides, this represents a vocabulary of 1287 letter-body sequences. Our recognizer builds a letter-body sequence by a concatenation of letter-body HMMs (each HMM describe one class of shape). As we said earlier, each letter-body sequence belongs to one and only one city name.

As described in [9], we train our system on sets {a,b,c} and test it on set {d}.

4.1 Validation of the alphabet

The results I vs II and I vs III of table 4 validate the new alphabet over standard 29 letter arabic alphabet. One advantage is : when the dots and diacritics are removed, the new alphabet allows the models to be trained on more samples, because the Arabic letters that differentiate only by dots will be regrouped and trained together, thus making the models more robust. The results also show that redistributing the dots and diacritics directly into the sequence of graphem is not a good solution, it clearly makes the recognition rate worse. First, a new feature extraction procedure should be considered. Second, it seems to add more noise than information. The sliding windows have a clear advantage here. They naturally take into account the dots and diacritics, whereas in explicit graphem segmentation, a hard decision has to be made about where the dot should be added in the sequence. Nevertheless, an interesting perspective would be to recognize the sequence of dots itself, and then combine it with the sequence of shapes.

4.2 Comparison to other systems

We compare our recognizer to the 4 best systems of state of the art. Three of them (ARAB-IFN [10], UOB [4], SCHMMs [2]) are based on HMMs with sliding windows or frames. We also use HMMs, but as we said in section 3, we believe that explicit graphem segmentation based on skeleton strokes is well adapted for Arabic, especially because of long descenders that go almost horizontally under the baseline. Microsoft Research presents an alternative work based on recognition of PAWs using Neural Networks [1], that also provides state of the art results. Our system achieves decent 87.4% recognition rate, to be compared with 89% to 90% of the best systems.

The main source of errors is undersegmentation, which causes half of the recognition errors. The segmentation algorithm requires further tuning on Arabic writing.

5 Conclusion and Perspectives

We have presented an Arabic offline recognition system based on explicit graphem segmentation. The recognizer is a Hybrid Neural Networks and Hidden Markov Models which gives nearly state of the art results. We introduced a new shape alphabet (the letter-body alphabet) which reduces the number of classes by exploiting some prior knowledge of the Arabic writing specificities and redundancies.

The perspectives are numerous. The graphem segmentation should be looked into in the first place, as it is currently the main source of recognition errors. The next step will be

	1st pos	2nd pos	10th pos
Train {a,b,c} I	91.1	94.7	98.1
Test {d} I	87.4	92.4	96.9
Test {d} II	81.6	87.6	96.0
Test {d} III	73.2	80.2	91.6
UOB	85.02	91.29	93.14
ARAB-IFN	89.1	91.7	95.9
SCHMMs	89.79	92.25	96.78
Microsoft Research	88.94		95.01

Table 4. Results in 1st, 2nd and 10th position
I : our novel shape alphabet
II : Arabic alphabet (29 letters) without dots
III : Arabic alphabet (29 letters) with dots

the design of the dots/diacritics recognizer, and its combination with the letter-body recognizer, using Weighted Finite State Machines (WFSM) formalism. This combination between letter-body and a dot recognizers should increase the recognition rate, as some wrong candidates will be discarded during the composition. Moreover, this will allow the use of this system over a larger vocabulary, wherein the processing of dots is mandatory to resolve ambiguities (see paragraph 2.5).

References

- [1] A. AdbulKader. Two-tier approach for arabic offline handwriting recognition. In *IWFHR*, 2006.
- [2] A. Benouareth, A. Ennaji, and M. Sellami. Semi-continuous hmms with explicit state duration applied to arabic handwritten word recognition. In *IWFHR*, 2006.
- [3] X. Dupre. *Reconnaissance de l'écriture manuscrite*. PhD thesis, Univ Rene Descartes - Paris V, 2003.
- [4] R. El-Hajj, C. Mokbel, and L. Likforman-Sulem. Reconnaissance de l'écriture arabe cursive : combinaison de classifieurs mmcs fenêtres orientées. In *CIFED*, 2006.
- [5] W. Kammoun and A. Ennaji. Reconnaissance de textes arabes vocabulaire ouvert. In *CIFED*, 2004.
- [6] L. M. Lorigo and V. Govindaraju. Offline arabic handwriting recognition: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(5):712–724, 2006.
- [7] H. Miled. *Reconnaissance de l'écriture semi-cursive : Application aux mots manuscrits arabes*. PhD thesis, PSI-La3i Univ Rouen, LIVIA ETS Montreal, 1998.
- [8] H. Miled, C. Olivier, and M. Cheriet. Modélisation de la notion de pseudo-mot en reconnaissance de mots manuscrits arabes. In *CIFED*, 2000.
- [9] M. Pechwitz, S. S. Maddouri, V. Maergner, N. Ellouze, and H. Amiri. Ifn/enit-database of handwritten arabic words. In *CIFED*, 2002.
- [10] M. Pechwitz, W. Maergner, and H. ElAbed. Comparison of two different feature sets for offline recognition of handwritten arabic words. In *IWFHR*, 2006.