Arabic online word extraction from handwritten text using SVM-RBF classifiers decision fusion

Randa Ibrahim Elanwar\(^{(a)}\), M.D.; Mohsen Rashwan\(^{(b)}\), Professor; Samia Mashali\(^{(a)}\), Professor
(a) Electronic Research Institute, (b) Faculty of Engineering, Cairo University.
Cairo, Egypt

eng_r_i_elanwar@yahoo.com, Mohsen_Rashwan@rdi-eg.com, samia@eri.sci.eg

Abstract: - In this paper, we propose a system for Arabic online word extraction from handwritten text lines, a problem addressed for the first time for Arabic language as there is no public dataset of Arabic online handwritten texts available so far. We collected a dataset of unconstrained online handwritten sentences and used it to design and evaluate our system.

First, our system classifies the white gaps between words connected components into either intra-word or inter-word gap according to some local and global online features extracted from each gap together with the groups of strokes encompassing the gap. The classifier is a polynomial kernel support vector machine (SVM) which decisions are used for initial word extraction.

A post stage is added to the system to test the extracted words for under-segmentation and resolve this under-segmentation by reconsidering the gap type decisions for the stuck word. Classifiers decision fusion takes place by consulting five different classifiers (four SVM and a radial basis function neural network ‘RBF NN’) and feeding their decisions to a separate pre-trained SVM to make the final decision. Most stuck words are correctly detected and a lot of them have been correctly resolved. The post stage leads to remarkable error reduction compared to single classifiers performance. Promising results are achieved regarding the fact that the unconstrained Arabic handwriting nature adds more difficulties to the problem.

Key-Words: - Arabic, Decision fusion, Online handwriting, RBF NN, SVM, Text segmentation, Word Extraction

1 Introduction

With the development and the wide spreading of fast computers and digital data entry devices, it became a very crucial need to convert any piece of hand sketching or machine printed paper into its corresponding electronic format version to, spread information, facilitate data access to save effort, and provide more services in much less time duration.

Handwriting recognition is the transcription of handwritten data into a digital format. It is a classical pattern recognition problem. The task lies in assigning a pattern to one class out of a set of classes [1], for example, assigning a handwritten sample to a specific character.

The handwritten data may be present in online or offline format. In the case of online recognition, a time ordered sequence of coordinates, representing the pen movement on an electronic sensing device like a tablet, is available. In the case of offline recognition, only the scanned or photographed image of the text is present.

Although automated handwriting recognition has been a research topic for more than forty years, there are still many open challenges in this field, especially in the domain of unconstrained handwritten text line recognition.

In an unconstrained handwritten text line recognition system, the first unit is word extraction. The output of this unit is a group of separate word images (offline case) or strokes (online case) extracted from the input text line. In online recognition systems the segmentation is usually based on the horizontal extend of the pen-movements to the right when the pen is lifted up. If this length exceeds a certain threshold it is assumed that a new word starts [6]. In off-line recognition systems more refined procedures are typically used.

Some researches investigate the determination of a threshold to distinguish between gaps lying within one word gaps and gaps lying between two words [7], [8], [5]. Other researches view the word extraction problem as a recognition problem as well, where the recognized entity in this case is the gap or white space encompassed between the words connected characters. The gap has to be classified to either inter-word gap if it lies between two different
words or intra-word gap if it lies within one word parts (a two class problem) [11], [9], [10], [4]. Different classifiers are used: neural networks (NN), support vector machines (SVM), Gaussian mixture models (GMM).

Researches of the former trend are like Liwicki et al. [6] who used different types of thresholds: fixed thresholds, Median white run-length (MWR), and Average white run-length (AWR) to split a line of text into sequences of connected components. Same thresholds were used by Varga et al. [11] where the decision about a gap is not only made in terms of a threshold, but also depends on the context of that gap, i.e. the relative sizes of the surrounding gaps are taken into consideration. Another research by Kurniawan et al. [5] used the MWR and the AWR but with contour following to extract words with high confidence.

Researches of the latter trend are like Quiniou et al. [9] who used RBF neural networks with an output associated confidence index to limit the total number of segmentation hypotheses. Sun et al. [10] employed and compared the performance of five different classifiers which are: Logistic discrimination analysis (LDA), K-nearest neighbor classification (KNN), Guassian mixture model (GMM), Multi-layer perceptron (MLP), and finally and the best of all was Support vector machine (SVM) which is also used by V. Papavassiliou et al. [8] but with different type of kernel. ICDAR competitions for handwriting segmentation presented systems of both trends [3], [4] showing the success of SVM based systems over other competent systems.

The recorded results for word extraction systems (online/offline) are generally ranging from 92% to 96% word extraction rate (WER) and 89% to 94% gap classification rate (GCR). For threshold-based systems the results achieved are in average of 86% WER and 95% GCR by Liwicki’s system [6], and an average between 92%-95% WER by Varga’s [11], 96% WER by Kurniawan’s [5]. Whereas the results achieved by gap classification systems are 96% WER by Quiniou’s system [9] and 92% WER by Papavassiliou’s system [8]. Sun’s system [10] achieved GCR 89.5% for LDA, 90% for KNN, 89.8% for GMM, 93.2% for MLP and 93.7% for SVM. ICDAR competitions winners’ systems achieved 92% [3] and 94% [4] WER.

All the previously reviewed researches are done for English handwritten text lines. Most researchers worked on small size private datasets of their own especially for online handwritten word extraction [9], [10], [8]. Those who used public datasets [6], [5], [11] have worked only on a small portion of the dataset.

The Arabic language differs greatly from other Latin languages not only in its characters cursive but also in the language structure as well. For Arabic language, we don’t yet have public text line data sets for online Arabic handwriting. Thus, we collected a dataset of our own. In a previous work [2], we presented the OHASD dataset: the first online handwritten Arabic sentence dataset. The dataset is natural, simple and clear. Texts are sampled from daily newspapers and are dictated to writers using tablet PCs for data collection. The current version (β version) includes 154 paragraphs written by 48 writers. It contains 670 text lines, more than 3800 words and more than 19,400 characters.

In our system we classify white gaps to either inter- or intra- word gap based on a set of local and global online features. A typical recognition system is build followed by a post stage composed of a hybrid of radial basis function neural networks (RBF-NN) and support vector machines (SVM) with decision fusion to reconsider decisions of the pre stage classifier. The fusion results obtained are remarkably better than single classifiers. Our system results are promising regarding the difficulty of the Arabic language unconstrained handwriting.

The rest of the paper is organized as follows. Section 2 gives a description of the word extraction system. The features used are detailed in Section 3. Experiments and results are presented in Section 4, and Section 5 draws some conclusions and proposes future work.

2 System Description

Our system units are preprocessing, feature extraction, classification and post-processing respectively.

The preprocessing unit in our system performs two essential functions: (1) removing dots and keeping the main strokes only. (2) Building groups of overlapping components/strokes (OCS) preceding and following each gap. The one preceding the gap including the gap start stroke is defined as (PreOCS) and the one following the gap including the gap end stroke is defined as (PostOCS).

The second unit is feature extraction. Features from pre and post OCS together with the gap are
extracted. Global (strokes level) features, local (gap) features are detailed in section 3.

The third unit in our system is the classifier. We used a hybrid combination of both classifiers RBF NN and SVM employed sequentially and in parallel with decision fusion.

In Fig. 1 features vectors extracted from the input text line are fed to parallel functioning SVM and RBF NN. Four SVMs are employed, each with different kernel function (Gaussian ‘rbf’, polynomial, linear and sigmoid). The SVM with polynomial kernel output, which has been experimentally found to have the best performance, is used for initial word extraction.

The extracted words undergo stick detection tests based on the likelihood probability of having extracted words with such output structure parameter values (number of ocs, word width, number of strokes per ocs). If one or more words are suspected of being stuck, the gap type decisions are reconsidered by passing all five classifiers outputs for each gap to a separate pre-trained SVM (stick resolution stage) which give the final decision whether or not to break the word up. Finally, dots are restored to the words having their corresponding main strokes.

Apparently stuck words are defined by wrong inter-word gap decision causing two or more words to be viewed as one. Consequently the faulty word should have one or more of the following characteristics: (1) abnormal word width, (2) excessive number of OCSs, and (3) excessive number of strokes.

These parameters are computed for the training data words and a Gaussian pdf is fit for each to represent the obtained values. These parameters and their distributions are computed for single OCS, single word, two successive words and three successive words. The test words parameters are examined using their likelihood probabilities to follow these four distributions.

The stick detection stage is a simple rule-based stage based on the log likelihood probability comparisons as shown in the flow chart in Fig. 2 where the parameters utilized are defined as:

- A0, A1, A2 and A3 are the absolute values of the test word width log likelihood probability following single OCS, single word, two successive words and three successive words distributions respectively.
- B1, B2 and B3 are the absolute values of the test word OCS number log likelihood probability following single word, two successive words and three successive words distributions respectively.
- C0, C1, C2 and C3 are the absolute values of the test word stroke number log likelihood probability following single OCS, single word, two successive words and three successive words distributions respectively.

G is the Gap Weight Penalty determined as the logarithm of the ratio between the sum of intra-gap widths to the sum of the test word OCS widths. If no gaps exists (i.e. total overlap cases) it will be set as the logarithm of the ratio between the average word width (computed from training data) to the sum of the test word OCS widths.
D is the Word Width Penalty determined according to the test word width (table 1).

Table 1 Word Width Penalty values

<table>
<thead>
<tr>
<th>Width penalty</th>
<th>Word width with respect to 1-word width</th>
<th>Word width with respect to 2-words width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smaller</td>
<td>log(WordWidth/1-wordWidth)</td>
<td>log(WordWidth/2-wordWidth)</td>
</tr>
<tr>
<td>Slightly Larger</td>
<td>log(WordWidth/1-wordWidth)</td>
<td>log(WordWidth/2-wordWidth)</td>
</tr>
<tr>
<td>Far Slightly Larger</td>
<td>log(WordWidth/1-wordWidth)</td>
<td>log(WordWidth/2-wordWidth)</td>
</tr>
<tr>
<td>Far Larger</td>
<td>log(WordWidth/1-wordWidth)</td>
<td>log(WordWidth/2-wordWidth)</td>
</tr>
</tbody>
</table>

Once detecting a word as being stuck, all the gap type decisions of this word have to be reconsidered by the stick resolution stage.

Training data is applied to the five classifiers and undergoes the stick detection test. Some gaps are wrongly classified. These cases act as solved examples because we know the correct decisions. The five classifiers decision values for each gap are now considered as the new features together with the known correct decision (gap type) are used to train the stick resolution SVM.

Thus, when a test work detected to be stuck, its five classifiers decision values for each gap are fed to the pre-trained SVM to approve or decline the gap type decision (Fig. 3).

Fig. 3 The stick resolving stage design

3 Features
We have two groups of extracted features: global features extracted on the text line level, and local features extracted at each gap.

Global features are those computed using contributions from all OCSs strokes (PreOCS and PostOCS) for each gap. Most features are investigated in [9, 10] and were proven to be significant using mutual information:

a. The gap bounding box width [9, 10].

b. Cosine of the angle between start stroke last sample and end stroke first sample with horizontal (i.e. acquiring end-to-end distance) [10].

c. The average overlap ratio of strokes within the OCS bounding box in both PreOCS and PostOCS.

d. The average angle difference between every two successive samples on a stroke referenced to the BB bottom-left corner in both PreOCS and PostOCS.

e. The angle difference between start stroke last sample and end stroke first sample referenced to the BB bottom-left corner in PreOCS [10].

f. The average centroid difference between every two successive strokes within PreOCS and PostOCS [9].

g. The centroid difference between start stroke and end stroke [10].

h. The total no of strokes contained in PreOCS and PostOCS.

i. The total no of strokes samples contained in PreOCS and PostOCS.

We have invented some local features relating the main gap feature (gap bounding box width) to the PreOCS (as a unit) and PostOCS (as a unit):

a. Gap bounding box width with respect to average main stroke width on the text line level.

b. Gap bounding box width with respect to PreOCS width.

c. Gap bounding box width with respect to PostOCS width.

d. Gap bounding box width with respect to number of samples in PreOCS.

e. Gap bounding box width with respect to number of samples in PostOCS.

f. River distance [10].

4. Experimental results
The paper addresses word extraction from online handwritten text lines for Arabic language. As illustrated in the introduction section, the Arabic language researches are holding back due to lack of language resources. For this reason, we collected a dataset of our own, OHASD dataset previously presented in [2].
The OHASD dataset is the first sentence dataset collected for Arabic online handwriting research purposes. 48 volunteers’ handwritings of different subjects are collected on tablet PCs. The handwriting is natural and unconstrained. The dataset is composed of 154 documents with total 670 text lines, more than 3800 words and more than 19,400 characters.

We divide our dataset into 3 parts: training, validation and test sets: 110 documents for training, 14 for validation and 30 for test. The number of inter-word gaps is 2264, 277 and 616 for training, validation and test respectively. The number of intra-word gaps is 3988, 437 and 1117 for training, validation and test respectively. The system is designed based on trials using the validation dataset (examples can be seen in Fig. 4).

Single classifiers are employed initially using our features to point out the best performing classifiers. The most popular and successful classifiers in literature are used: SVM and RBF neural network.

Analyzing the wrongly extracted words we observed the following:

1. Some inter-word distance is equal (Fig. 5.a) or much smaller (Fig. 5.b) than intra-word distances thus these distances are usually misclassified or in best cases if detected it leads to misclassifying wide intra-word distances.

2. Overlapping words has no inter-word gaps detected (Fig. 5.c).

Therefore, we have stuck (under-segmented) and split (over-segmented) words beside the correctly extracted ones. Fortunately, the split words are very few (represent about 2% of total words). Thus, the system focused only on solving the stuck words problem.

To solve the problem a stick detection stage based on probability density function estimation and log likelihood probability of word parameters is employed to investigate the words extracted by the best classifier. Applying stick detection rules, the system is capable to detect about 96% of stuck words (91 out of 95 words) but also some lengthy words (having too many strokes) are mistakenly detected as stuck words (represent 37.34% of correctly extracted words). Thus the stick resolution stage has to resolve stuck words carefully with minimal damage to mistakenly-stuck detected words.

Stick resolution is performed using a separate SVM classifier pre-trained on all five classifiers decisions at the gaps corresponding to possible segmentation locations. 62% of stuck words are resolved correctly, 8% are wrongly resolved, 16% of the mistakenly detected stuck words are damaged. Resolving stuck words by decision fusion achieved a remarkable rise in the word extraction rate (WER) compared to single classifier performance as shown in details in tables 3 and 4. Comparing to the best case single classifiers performance, stick resolution lead to error reduction in GCR equal to 31.19% and error reduction in WER equal to 43.89%.
Applying the test data to our system resulted at GCR of 88.4% and WER of 71.5%. Unfortunately, complete overlaps and obviously wide intra-word gaps in the test set are very frequent. To detect and solve this type of problem, we need language resources help rather than spatial features. However, our results seem promising regarding the fact of the Arabic language nature difficulty compared to Latin and the unconstrained handwriting writing challenge in the absence of contextual information help.

<table>
<thead>
<tr>
<th>Problem Type</th>
<th>Result</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolved</td>
<td>57</td>
<td>91</td>
<td>62.64%</td>
</tr>
<tr>
<td>wrongly resolved</td>
<td>8</td>
<td></td>
<td>8.8%</td>
</tr>
<tr>
<td>not yet resolved</td>
<td>20</td>
<td></td>
<td>21.98%</td>
</tr>
<tr>
<td>will never be resolved (complete overlap)</td>
<td>6</td>
<td></td>
<td>6.6%</td>
</tr>
<tr>
<td>Damaged (are mistakenly detected as stuck)</td>
<td>14</td>
<td>87</td>
<td>16.09%</td>
</tr>
</tbody>
</table>

Table 3: Decision Fusion stick resolving results

<table>
<thead>
<tr>
<th>SVM-RBF</th>
<th>SVM-Poly</th>
<th>SVM-Sig</th>
<th>SVM-Lin</th>
<th>RBF NN</th>
<th>Fusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCR</td>
<td>90.1</td>
<td>91.12</td>
<td>83.3</td>
<td>87.71</td>
<td>90.66</td>
</tr>
<tr>
<td>WER</td>
<td>69.19</td>
<td>69.15</td>
<td>66.69</td>
<td>61.12</td>
<td>67.79</td>
</tr>
</tbody>
</table>

Table 4: Comparing stick resolving by decision fusion to single classifiers performance

5. Conclusion and Future work

In this paper, we have presented an automatic word extraction system by gap type classification. A post stage is added to allow the detection and resolution of stuck words. In this stage multi-classifiers decisions fusion is used to reconsider the pre-stage gap type decisions obtained for stuck words. 5 classifiers are employed, 4 SVM and RBF. Thanks to it, we have been able to reduce the word extraction error significantly. Experimental results have shown remarkable improvements on the performances of our word extraction system over single classifiers results. Promising results are achieved for validation dataset. But unfortunately, smaller rates are obtained for test dataset due to excessive occurrence of overlapping and split word problems.

In the future, we may employ Hidden Markov models (HMM) with the help of natural language resources for split and stick detection-resolution on context base.

References: