Improved Arabic Characters Recognition by Combining Multiple Machine Learning Classifiers

Maytham Alabbas
Department of Computer Science
University of Basrah
Basrah, Iraq
ma@uobasrah.edu.iq

Raidah S. Khudeyer
Department of Computer Information Systems
University of Basrah
Basrah, Iraq
raidah.khudayer@uobasrah.edu.iq

Sardar Jaf
School of Engineering and Computing Sciences
University of Durham
Durham, UK
sardar.jaf@durham.ac.uk

Abstract—**In this paper, we investigate a range of strategies for combining multiple machine learning techniques for recognizing Arabic characters, where we are faced with imperfect and dimensionally variable input characters. Experimental results show that combined confidence-based backoff strategies can produce more accurate results than each technique produces by itself and even the ones exhibited by the majority voting combination.**

Keywords- optical character recognition (OCR); systems combination; kNN; SVM; PNN;

I. INTRODUCTION

Systems combination is a popular way of improving accuracy in different tasks. This is concerned with combining various systems, which perform the same task to exploit the unique advantage of each system and reduce some of the random errors. Systems combination has been widely applied in different fields such as natural language processing (NLP) [1], pattern recognition [2], and image processing. In this paper, a range of strategies for combining machine learning techniques for Arabic character recognition is evaluated. Arabic characters are used in various languages, like Arabic, Persian, Urdu, and other; more than half of billion people use the Arabic characters. Optical character recognition (OCR) is one of the most successful applications of automatic pattern recognition. OCR is the process of translating the graphical document into a textual one. The objective of characters recognition is to recognize handwritten accurately or typist characters to facilitate man-machine interaction. The motivation behind developing characters recognition systems is inspired by their wide range of applications including archiving documents, automatic reading of checks, and number plate reading. Although a large number of researches have been done in this field, there is no obvious mathematical function that could perform this translation. Considerable attention has been paid to Latin and Chinese characters recognition, while Arabic characters recognition still limited in spite of the challenge due to the difficulties of these characters, which arises from the characteristics of Arabic characters and the way of these characters are connected and written [3].

In the current work, several combining strategies for the recognition of isolated printed Arabic characters are investigated. These techniques based on combining three machine learning classifiers (k-nearest neighbors (kNN), support vector machine (SVM), and probabilistic neural network (PNN)) and then using different decision-making strategies such as majority vote, confidence systems, and others to produce the final decision. Simulation results prove that the combination strategies always provides a higher recognition rate compared with the results of each technique in isolation.

The rest of the paper is organized as follows: in Section 2 the Arabic characters are analyzed, and the main problems of these characters are presented. A brief explanation of the three recognition techniques is given in Section 3. Section 4 explains the combination strategies and the experiments performed using these strategies. Finally, Section 5 presents the conclusions.

II. FEATURES OF THE ARABIC CHARACTERS

Arabic writing and Arabic characters have many features that make the Arabic characters recognition system differs from the characters recognition systems for other languages such as Latin and Chinese. The Arabic language is written from right to left in a cursive script in both handwritten and typewritten. Arabic characters also have many characteristics that complicate the recognition of such characters. Some of these features are listed below:

1. The Arabic alphabet consists of 29 characters. Each Arabic character might have up to four forms depending on its relative position in the word (i.e., begin, middle, end, and isolated) that increase the number of patterns from 29 to about 100 patterns. Table I shows the Arabic character patterns (each table cell contains one character with its possible forms).
2. Most of the Arabic characters (around 16 of 29) have a character complementary that is associated with the body of the character. This complementary may be a dot, two dots, three dots, or zigzag (hamza). It can be above the character as in (ـف), below as in (ـب), or inside the character as in (ج).
3. There are different groups of characters that have the same body, but they are distinguished by the dots number (ن،ـت،ـث), the dots position (ج،خ), or the shape of a character complementary whether it is a dot or zigzag (ـئ ، ـنـ).

4. Both widths and heights of Arabic characters are variable (e.g. ا and ب).

TABLE I. ARABIC CHARACTER PATTERNS

Meem	Ayn	Seen	HHa	Hamza
مـ ـمـ ـم م	عـ ـعـ ـع ع	سـ ـسـ ـس س	حـ ـحـ ـح ح	ء ئ ئـ
Noon	Ghayn	Sheen	Khaa	Alif
نـ ـنـ ـن ن	غـ ـغـ ـغ غ	شـ ـشـ ـش ش	خـ ـخـ ـخ خ	ا ـا
Ha	Faa	Saad	Daal	Baa
هـ ـهـ ـه ه	فـ ـفـ ـف ف	صـ ـصـ ـص ص	د ـد	بـ ـبـ ـب ب
Waaw	Qaaf	Dhad	Dhaa	Taa
و ـو	قـ ـقـ ـق ق	ضـ ـضـ ـض ض	ذ ـذ	تـ ـتـ ـت ت
Yaa	Kaaf	Taa	Raa	Thaa
يـ ـيـ ـي ي	كـ ـكـ ـك ك	ط ـط	ر ـر	ثـ ـثـ ـث ث
	Laam	Dhaa	Zaay	Jeem
	لـ ـلـ ـل ل	ظ ـظ	ز ـز	جـ ـجـ ـج ج

III. RECOGNITION METHODOLOGIES

Three machine learning techniques were chosen to classify printed Arabic characters that are shown in Table I. In the following subsections, a functional description of these methods is introduced [4].

A. *k-Nearest Neighbors*

K-nearest neighbors (kNN) is a simplest machine learning algorithm, which is used for classifying objects based on the nearest training sets in the feature space. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its *k* nearest neighbor, where k is a small positive integer.

B. *Support Vector Machine*

The support vector machine (SVM) is based on statistical learning theory. The standard SVM takes a set of input data and predicts, for each given input, which possible classes from the input. The process of rearranging the objects is known as mapping. After learning by quadratic programming, the samples of non-zero weights are called support vectors (SVs).

C. *Probabilistic Neural Network*

Probabilistic neural network (PNN) is an implementation of a statistical algorithm called kernel discriminant analysis. The PNN architecture is composed of many interconnected processing units, or neurons, organized in four successive layers: input layer, pattern layer, summation layer, and output layer. The input layer does not perform any computation and simply distributes the input to the neurons in the pattern layer. The pattern layer contains one neuron for each case in the training data set. It stores the values of the predictor variables for the case along with the target value. The summation layer performs an average operation of the outputs from the pattern layer for each class. The output layer performs a weighted vote, selecting the largest value and uses the most significant vote to predict the target category.

IV. EXPERIMENT RESULTS

As we mentioned before, the aim of the current work is to investigate the evaluation of a range of strategies to combining three most popular and efficient machine learning techniques for Arabic characters recognition system, where the input characters are imperfect and dimensionally variable, and compared their results with each method in isolation.

Each recognition system, here, is conducted through three main modules. The first module is responsible for preparing the input images by acquisition and digitizing of the image, remove noise, binarized and thinning. The second module extracts the main features of the preprocessed images. The third module processes the main features to recognize the input characters. Several systems are used in this module; isolated systems such as kNN, SVM, PNN and combination among them. A brief description of each module is in the following lines.

A. *Preprocessing*

The preprocessing attempts to eliminate some variability related to the writing process, such as the variability due to the writing environment, writing style, acquisition and digitizing of the image. The main steps of preprocessing module are as below:

1) *Noise reduction*

Images usually still contain noise. One approach is applying adaptive median filter [5] to achieve noise reduction. The advantage of the median filter is to keep the edges of the image as well as to eliminate some of the noise.

2) *Binarizing*

This part is responsible for converting the input image to a binary image by replacing all pixels in the input image with luminance greater than a particular level with the value 1 (White); otherwise with the value 0 (Black).

3) *Thinning*

Thinning is done to make the characters around one pixel wide.

B. *Feature Extraction*

The extracted characters from the input image have different dimensions (e.g., the width of the Arabic character ب is distinct from the width of the Arabic character ا, and the same for the height). To deal with this challenge, the discrete cosine transform (DCT) is adopted to extract the features of the characters. DCT is a technique to convert the image data into its elementary frequency components where high-value coefficients are clustered in the upper left corner and low-value coefficients in the bottom right of the resulted matrix. To improve the performance and efficiency of the recognition systems, three various feature extractors have been investigated, e.g., 10 coefficients, 20 coefficients, and 64 coefficients. Indeed, each extractor kind range is different from those of the other extractor kinds. Each feature extractor, therefore, extracts vector which is not uniform with the other vectors extracted from others. We do not have enough space to include all the details and the results of these feature extractors. Instead, for simplicity, we focus here on the one that has been the most useful in practice, which is 64 DCT coefficients feature extractor: apply the DCT on a character and selecting the first 64 higher value DCT coefficients, which are extracted in a zigzag fashion as a feature vector for recognizing this character.

C. *Recognition technique*

Different systems have been investigated here, as follows:

- **System$_1$**: this system uses the kNN in isolation with the number of nearest neighbors (k=1). The output is an integer number that represents a character, e.g., 1=all patterns of Alif, …, 28=all patterns of Yaa.
- **System$_2$**: this system uses the SVM in isolation, which relies on multi-class SVM (28 SVMs, one-rest method) with the order of polynomial kernel equal to 2.
- **System$_3$**: this system uses the PNN in isolation with the spread of radial basis functions equal to 0.2. The output is an integer number that represents a character, e.g., 1=all patterns of Alif, …, 28=all patterns of Yaa.
- **System$_4$**: this system accepts the result of System$_1$ and System$_2$ if they agree and backoff to System$_3$ if they do not (whether or not the backoff system agrees with either of the chosen pair).
- **System$_5$**: this system accepts the result of System$_1$ and System$_3$ if they agree and backoff to System$_2$ if they do not (whether or not the backoff system agrees with either of the chosen pair).
- **System$_6$**: this system accepts the result of System$_2$ and System$_3$ if they agree and backoff to System$_1$ if they do not (whether or not the backoff system agrees with either of the chosen pair).
- **System$_7$**: this system accepts the result of System$_1$ and System$_2$ if they agree and backoff to a most confident system (1 or 2) if they do not.
- **System$_8$**: this system accepts the result of System$_1$ and System$_3$ if they agree and backoff to a most confident system (1 or 3) if they do not.
- **System$_9$**: this system accepts the result of System$_2$ and System$_3$ if they agree and backoff to a most confident system (2 or 3) if they do not.
- **System$_{10}$**: this system accepts the result of at least two systems if they agree and backoff to a most confident system (1-3) if they do not.
- **System$_{11}$**: this system accepts the result of System$_1$, System$_2$, and System$_3$ if they agree and backoff to a most confident system (1-3) if they do not.
- **System$_{12}$**: this system accepts the result of a most confident system only.

In the systems (7-12), we used a technique that depends on using the system which is known to be most reliable for each Arabic character. After testing the individual systems on the testing set with different levels of noise, we found the most reliable system for each Arabic character and then used these confidence levels to decide how much each system should be trusted for each character. We find that, for instance, System$_1$ should be trusted when the character is 'ر', whereas System$_2$ should be trusted when the character is 'ا'. Consider the three systems are entirely disagreed to recognize a character. For example, System$_1$ decides the character is 'ر' with confidence level for classifying this character equal to 90%, System$_2$ decides the character is 'د' with confidence level for classifying this character equal to 80%, and System$_3$ decides the character is 'ذ' with confidence level for classifying this character equal to 75%. In this case, the final decision will be the character 'ر' because the System$_1$ has high confidence level for classifying such character.

- *Results*

We carried out experiments using the isolated systems (1-3) above by training them on the training set, which is all Arabic characters in Table I (except Hamza) with Arial 14-point font. To make our experiments more realistic, all systems are tested on the testing set, which is the same training characters set after corrupted by three levels of "Salt & Pepper" noise (i.e. 10%, 30%, and 50%).

The results of these experiments, regarding the recognition rate, are illustrated in Table II. As it can be seen in the table below, System$_1$ obtains the best result.

TABLE II. ISOLATED SYSTEMS RECOGNITION RATES.

System	Recognition rate for the noise level		
	10%	*30%*	*50%*
System$_1$	**96%**	**92%**	64%
System$_2$	90%	82%	**69%**
System$_3$	94%	86%	48%

If you have multiple classifier systems, and they all suggest a particular character, then the only thing you can do is to accept that suggestion. The key issue is what to do when they disagree, but before investigating this, it is worth looking at what happens when they do agree.

We, therefore, looked at the precision (P), recall (R) and F-score (F) for various combinations of systems on cases where they agreed. Table III shows the precision, recall, and F-score for the merger of the systems output where they agree, either pairwise or unanimously.

TABLE III. PRECISION (P), RECALL (R) AND F-SCORE (F) FOR AGREEMENT OUTPUT FOR TWO SYSTEMS

Systems	Noise Level (10%)			Noise Level (30%)			Noise Level (50%)		
	P%	*R%*	*F*	*P%*	*R%*	*F*	*P%*	*R%*	*F*
System$_1$+ System$_2$	**100**	86	0.93	97.4	76	0.85	85	51	0.64
System$_1$+ System$_3$	96.9	93	0.95	94.4	85	0.90	71.8	51	0.60
System$_2$+ System$_3$	98.8	85	0.91	97.3	73	0.83	65.5	36	0.47
Three systems agree	**100**	84	0.93	**98.6**	72	0.83	**97.2**	35	0.52
At least two systems agree	97	**96**	**0.97**	94.7	**90**	**0.92**	73.5	**61**	**0.67**

Unsurprisingly, the precision on combining systems is considerably higher than the accuracy of any individual system. More importantly, when we combine only two systems, we find that the combination of $System_2$ with either of $System_1$ and $System_3$ gives mostly better precision and lower recall than combining $System_1$ and $System_2$. This is slightly surprising: $System_2$ uses a different technique from the systems (1 and 3), and hence when it agrees with one of them it is likely that they have arrived at the same conclusion by different routes, and hence that this conclusion has good supporting evidence.

A system, however, is required to give a complete recognition result, so we have to recommend a backoff strategy for cases where the systems do not all agree. Here, we briefly consider two promising strategies for dealing with this challenge–taking the output if all three systems agree (highest precision in Table III) and taking the output if any pair agrees (highest F-score in Table III)–and investigate a range of backoff strategies.

These strategies are divided into two groups: (i) voting-based backoff group which is a set of voting strategies; and (ii) confidence-based backoff group which is a set of techniques based on identifying which system is best at dealing with particular kinds of characters.

The latter group has proved highly effective for combining POS taggers [6] and parsers [7], and it seemed *prima facie* plausible that it would also work for characters recognition.

Table IV shows the results obtained from applying the combining systems for different noise levels of the testing set.

TABLE IV. RECOGNITION RATES FOR COMBINING SYSTEMS, DEFERENT NOISE LEVELS.

System	Recognition rate for the noise level		
	10%	*30%*	*50%*
$System_4$	96%	90%	61%
$System_5$	96%	90%	61%
$System_6$	96%	90%	61%
$System_7$	**100%**	**98%**	**81%**
$System_8$	97%	93%	64%
$System_9$	99%	95%	78%
$System_{10}$	96%	90%	61%
$System_{11}$	**100%**	**98%**	**81%**
$System_{12}$	**100%**	**98%**	**81%**

As noted above, we find that the confidence-based backoff systems (7-12) outperform each of the individual systems (1-3), and they also achieve better recognition rates than the simple voting-based backoff systems (4-6).

V. CONCLUSIONS AND FUTURE WORK

If you have multiple systems that perform the same task, it seems sensible to suppose that you can obtain better performance by using some judicious combination of them than can be achieved by any of them alone. A lot of combining strategies have been proposed, with majority voting being particularly popular. We have investigated here a range of strategies for combining machine learning techniques for Arabic characters recognition: the best strategy we have found for recognizing involves asking each of the systems how confident it is, and accepting the answer given by the most confident one. We hypothesize that the reason for the effectiveness of this strategy for characters recognition arises from the fact that the individual systems work in essentially different ways (e.g., different underlying algorithms), and hence if they make systematic errors, these will tend to be different. This means, in turn, that the places where they do not make mistakes will be different.

Based on the encouraging findings in the current work, two further research tasks have been identified. First, to improve the recognition efficiency and generality of the presented systems, these systems will be evaluated on multi-font and multi-size training and testing sets. Second, the current systems will be extended to deals with Arabic text rather than isolated characters by adding segmentation module to split an input text into words and then into characters.

REFERENCES

[1] M. Alabbas and A. Ramsay, "Combining strategies for tagging and parsing Arabic," in *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Langauge Processing (ANLP)*, Doha, Qatar, 2014, pp. 73–77.

[2] G. Giacinto, F. Roli, and G. Fumera, "Selection of Classifiers Based on Multiple Classifier Behaviour," in *Advances in Pattern Recognition: Joint IAPR International Workshops SSPR 2000 and SPR 2000 Alicante, Spain, August 30 – September 1, 2000 Proceedings*, F. J. Ferri, J. M. Iñesta, A. Amin, and P. Pudil, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 87-93.

[3] I. Supriana and A. Nasution, "Arabic Character Recognition System Development," *Procedia Technology,* vol. 11, pp. 334-341, 2013.

[4] W. Martinez and A. Martinez, *Computational Statistics Handbook with MATLAB*, 3rd ed.: Chapman and Hall/CRC, 2015.

[5] Y. Zhao, D. Li, and Z. Li, "Performance enhancement and analysis of an adaptive median filter," in *International Conference on Communications and Networking (CHINACOM '07)*, China, Shanghai, 2007, pp. 651-653.

[6] M. Alabbas and A. Ramsay, "Improved POS-Tagging for Arabic by Combining Diverse Taggers," in *Artificial Intelligence Applications and Innovations: 8th IFIP WG 12.5 International Conference, AIAI 2012, Halkidiki, Greece, September 27-30, 2012, Proceedings, Part I*, L. Iliadis, I. Maglogiannis, and H. Papadopoulos, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 107-116.

[7] M. Alabbas and A. Ramsay, "Improved Parsing for Arabic by Combining Diverse Dependency Parsers," in *Human Language Technology Challenges for Computer Science and Linguistics: 5th Language and Technology Conference, LTC 2011, Poznań, Poland, Revised Selected Papers*, Z. Vetulani and J. Mariani, Eds., ed Cham: Springer International Publishing, 2014, pp. 43-54.