International Journal of Software Engineering and Its Applications
Vol.8, No.2 (2014), pp.11-20
http://dx.doi.org/10.14257/ijseia.2014.8.2.02

Lexicon-Driven Word Recognition Based on Levenshtein Distance

Perdana Adhitama, Soo Hyung Kim and In Seop Na*

School of Electronics and Computer Engineering, Chonnam National University
Gwangju, 500-757, South Korea
perdana_adhitama@yahoo.com, shkim@jnu.ac.kr, ypencil@hanmail.net

Abstract

In this paper, we propose a word recognition method for printed Arabic word images using HMM and Levenshtein Distance. The existing algorithm has the difficulty for Arabic text recognition to treat various fonts and sizes. This is because Arabic characters are cursive and each character may have up to four different shapes based on its location in a word. Our work begins with segmentation of a word into characters. Then each character is recognized individually using HMM classifier. Since the recognition of HMM is not accurate enough, we apply Levenshtein distance to correct misclassification and miss segmentation of a character in a word. Levenshtein distance works by comparing between recognized word and every words in a dictionary. We tested our proposed system with APTI dataset, and the achieved average recognition rates in more than 95% for six different fonts.

Keywords: *Hidden Markov Model, Printed Arabic Word Recognition, OCR, Levenshtein distance, Segmentation*

1. Introduction

Among pattern recognition application, there is automatic reading of a text namely, text recognition. The objective is to imitate human ability to read printed text with human accuracy, but at a higher speed.

Over the past ten years, research related to Optical Character Recognition (OCR) have achieved considerable improvements. There are many research related to OCR in recognizing Korean, Latin, and Chinese character. However, Arabic text recognition has not gained much pay attention. This might be due to a challenging task which has to cope several difficulties. They consist of 4 different shape depends on the position in a word. In summary, characteristics of Arabic characters can be described as follows [1].

- Arabic text is cursive and written from right to left.
- Consist of 28 characters.
- Connected to each other on the baseline.
- Arabic characters might be increase from 28 to 108 characters, due to their position in the word.
- Some Arabic characters have same shape but they are distinguished from each other based on the number and position of dots.

* Corresponding author In Seop Na (ypencil@hanmail.net)

ISSN: 1738-9984 IJSEIA

There are two approaches to overcome the problem of Arabic cursive text: the global approach and the analytical approach. The global approach treats the world as a whole. Features are extracted from unsegmented word and compared to a model. In contrast, analytical approach segment the word into smaller units which may or may not correspond to characters. Finally, the characters are recognized individually into a classifier.

Hidden Markov Model (HMM) has been proved as one of classifier used by many researchers to recognize Arabic text. HMM offers several advantages. They are resistant to noise, they can tolerate variations in writing, and HMM tools are freely available. Previous research showed HMM has been applied in wide area of applications, especially in speech processing [2]. The advantages of HMM which is based in statistical model can be applied in character recognition either online [3] or offline [4].

The advantages of HMM in character recognition has motivates many researchers to implement in Arabic text recognition [5]. They use sliding window to extract 16 features. The achieved average recognition rates were between 98.08% and 99.89%. Manal et al [6] use HMM to recognize Arabic text using analytical approach where word are segmented into characters. As a result, the recognition rate was 81%.

It is worth mentioning that no generally accepted database for printed Arabic text recognition was freely available for researchers. Therefore, many researchers of Arabic text recognition use different data, and hence the results are varies and may not comparable. This raises the need for researchers to make their data available for other researchers as a first step and to work on producing a comprehensive database for printed Arabic text recognition. In this paper we will use public dataset which is freely available for academic purpose.

This paper presents segmentation based Arabic word recognition using HMM and Levenshtein distance. Our algorithm begins with segmentation of Arabic word image. The segmentation process use sliding window which moves from right to left to split Arabic word into characters. Features are extracted from segmented words. Moment invariants are extracted from text image and fed to recognizer. From training process we build HMM model for each character. The testing dataset is used against the model that we have obtained during training section to evaluate our system. Since the result of word recognition is not quite promising, we apply post processing using dictionary based error correction. In post processing, we compare the similarity between misspelled words and lexicon in our dictionary and select the lowest distance to fit as a correction.

2. Proposed Approach

Our research uses the concept of word segmentation where the words are segmented into characters then each characters are recognized individually. The proposed technique is based on sliding window to segment Arabic word image. Moment invariants and other feature vectors are used to train HMM models. Finally, post processing using Levenshtein distance will compare the weight between recognized word and a lexicon in dictionary. The smallest weight will be chosen as the best candidate of Arabic word.

2.1. Preprocessing

The preprocessing step attempts to obtain baseline of Arabic word. This is achieved by calculating horizontal projection of Arabic word. All image acquisition have different position of baseline, therefore there is no ideal situation in which baseline is presented. In Table 1, we will describe our preprocessing step.

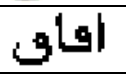
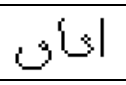
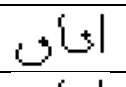
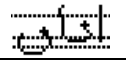
2.1.1. Dots Removal: When locating the prospective segmentation point using vertical histogram technique, punctuation marks will cause incorrect segment point because most of the punctuation marks placed on another character or above or under ligature. In general, the density shape of any dot is smaller than any other character in Arabic text; therefore we can mark it and remove. The dots removal is calculated, if the density is less than the constant value, the shape will be marked as dot then removed by replacing foreground pixel by background pixel.

2.1.2. Skeletonization: Due to variability of word image, it was first essential to uniform the characteristic that varied from one word to another. One such important characteristic was reducing the stroke width. A simple algorithm was employed for performing uniform stroke width of each word. It is necessary to make the stroke width uniform for each word image preparing for the heuristics that will be employed in the segmentation algorithm. The benefits of skeletonization image are: (1) unification a characteristics of varied word image. (2) Locating prospective segmentation point more accurately by using vertical histogram. (3) Locate the holes and parallel strokes. (4) Speed up the algorithm and make it more efficient.

2.1.3. Holes Filling: Some Arabic characters have a hole either in ascender or descender for example Waw (و) Fa (ڤ), Gha (ڧ), Ha' (ھ). Since this algorithm use a sliding window, it would be a problem during the scanning meet those characters. Therefore, we perform holes filling to fill the holes in Arabic characters. As a result the holes did not considered as a segmentation point.

2.1.4. Baseline Estimation: Some Arabic characters are contained on strokes above or under the center of Arabic word such as: ش ,ر ,ي ,س etc. These strokes are called ascenders and descenders, respectively. Ascenders and descenders of the main body of the wordmay overlap parts of the characters in the main body that do not contain strokes. To increase over segmentation of the word image, it is important to remove ascenders and descenders before the beginning of segmentation process.

Table 1. Preprocessing Processes

1	Grayscale input image	
2	Dots removal	
3	Skeletonization	
4	Holes filling	
5	Baseline, top line, bottom line estimation	

In this paper, we calculate baseline estimation using horizontal histogram by counting the total number of foreground pixels of word image. The baseline is a medium line between in the Arabic word in which all the connections between the successive characters take place. Besides baseline estimation, we also calculated top line and bottom line. This techniques is used to segment ascender and descender characters. The top line was estimated by half of the

distance between top border of image and baseline, while bottom line was calculated by half of the distance between baseline and height of the image.

2.2. Segmentation

In this step we will perform segmentation. Segmentation is done by using sliding window. First, horizontal projection in Figure 1 was used to segment words into sub-words. This technique will find the sum of the foreground pixel which has non-zero values.

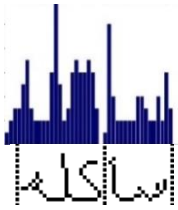


Figure 1. Horizontal Projection

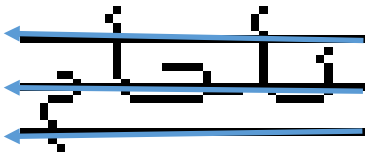


Figure 2. Three Points Sliding Window

Sliding window was performed from right to left. The segmented point was marked by detecting the value from zero to one. In addition, baseline, top line and bottom line were used to segment the characters. This method shows in Figure 2 where sliding window is calculated.

In general the Arabic characters can be distinguished based on the position according to baseline. The first point or top line used to segment the characters at the top of the baseline. The baseline is used to segment the characters which is located at the middle. While the bottom line is used to segment characters under the baseline. Because of those characteristics, we use three points in a sliding window to segment Arabic characters [7]. Some of Arabic characters based on the position on baseline can be described as follow.

- Upper characters: {ا, ـلـ, ك, ط , ظ } at the top line.
- Horizontal characters :{ب , ت, ث , ن , ف , ق , د , س , ش , ص , ض , ه , ي } at the baseline.
- Lower characters: {و , ر , ز , ع , غ , ج, ح, خ } at the bottom line.

Segmenting character to a small parts could be expensive due to the complex structure and various font style. However our segmentation algorithm has been able to segment most of Arabic words. The miss segmentation was mainly because of some characters which have three strokes such as characters س and ش.

2.3. Feature Extraction

All word images in our dataset are in gray level. Hence, the used of preprocessing to extract features is necessary. In this research, features are extracted from the skeletonized words by using moment invariants and other features. In summary, we can conclude the feature in Table 2 below.

Table 2. Feature Extraction

f1	Position (middle, isolated, end)
f2	Number of holes
f3	Position of character (top, middle, bottom)
f4,f5	Horizontal and Vertical Transition
f6,f7	Maxima and minima of vertical histogram

f8,f9	Maxima and minima of horizontal histogram
f10	Connected component
f11	Pixel Ratio
f12-f18	Moment Invariants

The horizontal and vertical transition is a technique used to detect curvature of each character and found to be effective for this purpose [8]. The idea is to compute the location and number of transition from background to foreground along horizontal and vertical lines. This transition calculation is performed from right to left, left to right, top to bottom, and bottom to up.

Hu [9] introduced the use of moment invariants as features for pattern recognition. The moment invariants are used to evaluate seven distributed parameters of a numeral image. In any character recognition system, the characters are processed to extract features that uniquely represent properties of the character. The moment invariants are well known method to be invariant under translation, rotation, scaling and reflection. They are measures of the pixel distribution around the center of gravity of the character and allow to capture the global character shape information.

The moment invariants have innately continues values. If they are considered as continues type, we will encounter an infinite number of possible observation vectors that may not be modeled by discrete HMM. Therefore, we apply vector quantization. The resulting feature vector is mapped against predefined codebook vectors, and replace with symbol representing the nearest codebook vector.

2.4. Classification

In this paper we use Hidden Markov Model (HMM) as a classifier. A HMM assumes that the sequence of observed feature vectors representing each printed text line is generated by a Markov Model. A Markov model is a finite state machine that can move to a new state of stay in its current state at each time unit. The probability of generating the text observation vector, O, by model λ through state sequence S is the product of the probabilities of the outputs and the probabilities of the transitions:

$$P(O, Q|\lambda) = \pi_1 \times b_1(o_1) \times a_{12} \times b_2(o_2) \ldots \quad (1)$$

The model parameters are estimated in the training phase using Baum-Welch algorithm to maximize the likelihood probabilities of the training data given the model. The sequence of state transition that gives the highest probability is determined by Viterbi algorithm.

This study uses a left-to-right HMM for our printed Arabic text recognition. This model allows relatively large variations in the horizontal position of the Arabic text. The sequence of state transition in the training and testing of the model is related to each text feature observation. Although each printed letter model may have a different number of states, we decided to use the same number of states for all letters. In our system, we use seven states of HMM. It can be seen on Figure 3 below.

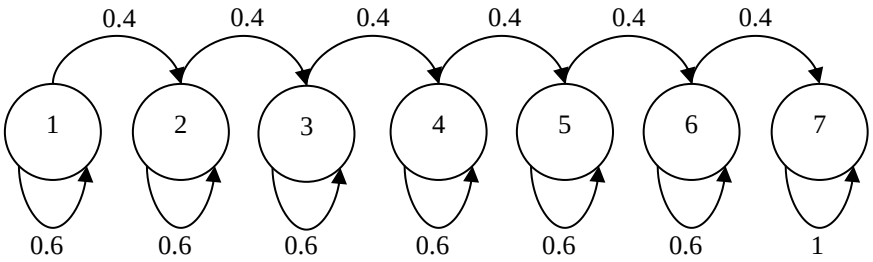


Figure 3. Seven States of HMM

2.5. Post Processing

During evaluating proposed algorithm with testing image, there might be miss classification and miss segmentation. Once a character miss classification in a word, then the word is misspelled. And hence we applied post processing using Lavenshtein distance.

We use dictionary based word correction methodology, also known as lexical error correction. In this approach, a lexicon or a lookup dictionary is used to spell check OCR recognized words and correct them if they are misspelled [10].

Levenshtein distance is a measure of the similarity between two strings, the source string (s) and the target string (t). The distance is the number of deletions, insertions, or substitutions required to transform s into t. The greater the Levenshtein distance, the more different the strings are. In our case, the source string is a recognized Arabic word and the target string is one of a word in a dictionary. In this paper, we will use Levenshtein distance in the post processing correction to increase the performance of word recognition. A brief detail of Levenshtein distance algorithm is given below as an algorithmic expression by essentially steps:

1. ***Initialization***
   - ***a.*** *Set n to be the length of s, set m to be the length of t.*
   - ***b.*** *Construct a matrix containing 0..m rows and 0…n columns.*
   - ***c.*** *Initialize the first row to 0…n.*
   - ***d.*** *Initialize the first column to 0…m.*
2. ***Processing***
   - ***a.*** *Examine s (I from 1 to n).*
   - ***b.*** *Examine t (j from 1 to m).*
   - ***c.*** *If s[i] equals t[j], the cost is 0.*
   - ***d.*** *If s[i] does not equal t[j], the cost is 1.*
   - ***e.*** *Set cell d[i,j] of the matrix equal to the minimum of:*
     - ***i.*** *The cell immediately above plus 1:d[i-1,j] + 1.*
     - ***ii.*** *The cell immediately to the left plus 1:d[I,j-1] + 1.*
     - ***iii.*** *The cell diagonally above and to the left plus cost: d[i-1,j-1] + cost.*
3. ***Repeat until d[n,m] value is found.***

Suppose we have two strings below:

X= مكر هة (TaaaClosed_E Haa_B Raa_E Kaaf_M Miim_B)

Y= مـ كر هة (TaaaClosed_E Haa_B Raa_E Kaaf_M Miim_M)

The rows represent X as a misspelled word while the columns represent Y as an original word. From those two strings we can see that م (Miim) character is miss recognized according to the position between middle and beginning. By applying Levenshtein distance above we can see the results in Figure 4 below.

	ة	ه	ر	ك	م
ة	0	1	2	3	4
ه	1	0	1	2	3
ر	2	1	0	1	2
ك	3	2	1	0	1
مـ	4	3	2	1	1

Figure 4. An Example of Levenshtein Distance in Arabic Word

We calculated the distance by comparing the string between miss recognized word and lexicon in dictionary. There are two causes of misspelled word, the first is because of the recognition another is because of segmentation. In this paper, we made an assumption one character is either miss recognized or miss segmented. Therefore we calculate the distance with the lexicon in dictionary which length are the same as misspelled word, misspelled word +1, and misspelled word -1; For example the misspelled word has 4 characters, therefore we calculate Lavenshtein distance with 4 characters lexicon, 3 characters lexicon and 5 characters lexicon.

3. Experimental Results

To evaluate the performance of our algorithm, experiments have been conducted on some parts of the large APTI (Arabic Printed Text Images) database [11]. In all test, recognition rates have been evaluated at word and character level.

The APTI database was developed with 113,284 different Arabic words of decomposable and non-decomposable words and 10 fonts. These fonts have been selected to represent complexity of Arabic characters. Different font sizes are also available in APTI: 6, 7, 8, 9, 10, 12, 14, 16, 18 and 24 points. Each word image in the APTI database is fully described using an XML file containing ground truth.

To evaluate our proposed system, we used six different fonts from APTI database. The fonts are: Tahoma, Simlified Arabic, Traditional Arabic, Andalus, Naskh and Thuluth. These fonts cover different complexity scales ranging from Tahoma which is a simple font with no overlap or ligature to Thuluth which is more complex. In this paper, we selected 2,500 word images. From those images, we split the images into two purpose for training and testing. 1,500 word images is used to train HMM to create HMM model while 1,000 images is used as testing to evaluate our algorithm.

To show that our system is better than existing algorithm, we compare our system with other system in Tables 3 and 4. We also show the improvement of Arabic word recognition by using Levenshtein distance as post processing. Results are comparable to the state of the art in printed text recognition.

Table 3. Recognition Rates (%) in each Font

Font	Proposed method		IPSAR System [4]
	Without post processing	With post processing	Word Recognition
Thuluth	75.4	90.8	85.9
Naskh	79.4	92.6	85.9
Simplified Arabic	82.6	95.7	83.0
Traditional Arabic	83.4	98.2	88.7
Tahoma	82.8	96.5	92.4
Andalus	80.5	97.2	90.3

Table 4. Average Recognition Rates

System	Recognition rates
IPSAR System [4]	89.70%
Method [12]	81.50%
This paper	95.16%

From the results above, we can see the improvement of post processing using Lavenshtein distance. The post processing method effectively raise up the word recognition. The improvement could be achieved up to 15% compared with normal method. Among six fonts, Thuluth and Naskh font scored lowest performance. This might be because of complex structure of Arabic font such as overlaps, ligatures and flourishes.

Since our system based on segmentation where each characters is recognized individually there are some miss classification. For example ـفـ (faa' middle) and ـغـ (Ghayn) are miss recognized as ﻓ (faa' begin) and ـفـ (faa' middle) respectively. This individual recognition in each character which might be the reason why our algorithm is lower than the others. Another reason is because of miss segmentation. As we have explained, word recognition is calculated with lexicon which length is only one difference with misspelled word. As a result a word which has 2 characters difference length cannot be corrected with word correction.

4. Conclusion

A new system of printed Arabic text recognition has been presented. The proposed system is based on HMM and post processing with Levenshtein distance. The system is segmentation based which requires segmentation the Arabic words. Some miss recognized characters are because of similar shape with the difference only in the number of dots. The system performance has been improved when implementing Levenshtein distance algorithm. By adding error correction to our system, the performance increase up to 95.16%. Future work to improve the system performance may be introduced with different feature extraction method and expanding the classification model with various font style.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST)(2013-056480).

References

[1] A. H. Hassin and X.-L. Tang, "Printed Arabic Character Recognition Using HMM", Journal of Computer Science and Technology, vol. 19, no. 4, **(2004)** July, pp. 538-543.
[2] L. Rabiner, "A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", Proceedings of the IEEE, vol. 77, **(1989)** February, pp. 256-286.
[3] A. Mustafa Ali, M. Zeki Akram and M. Zeki Ahmed, "Recognition Techniques for Online Arabic Handwriting Recognition Systems", International Conference on Advanced Computer Science Applications and Technologies, **(2012)**, pp. 518-523.
[4] M. S. khorsheed, "Offline Recognition of Omnifont Arabic Text Using the HMM Toolkit (HTK)", Pattern Recognition Letters, vol. 28, **(2007)**, pp. 1565-1571.
[5] H. A. Al-Muhtaseb, S. A. Mahmoud and R. S. Qahwaji, "Recognition of Off-line Printed Arabic Text Using Hidden Markov Models", Signal Processing, vol. 88, **(2008)**, pp. 290-2912.
[6] M. A. Abdullah, L. M. Al-Harigy and H. H. Al-Fraidi, "Off-line Arabic Handwriting Character Recognition Using Word Segmentation", Journal of Computing, vol. l4, **(2012)**, pp. 40-44.
[7] P. Adhitama, S. Hyung Kim and I. Seop Na, "Arabic Character Segmentation Using Horizontal Projection and Sliding Window", KISM Spring Conference, vol. 2, **(2013)**, pp. 308-311.

[8] O. D. Trier, A. K. Jain and T. Taxt, "Feature Extraction Methods for Character Recognition a Survey", Patten Recognition, **(1996)**, pp. 641-662.
[9] M.-K. Hu, "Visual Pattern Recognition by Moment Invariants", Information Theory, vol. 8, **(1962)**, pp. 179-187.
[10] V. I. Lavenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals, Cybernetics and Control Theory", vol. 10, **(1966)**, pp. 707-710.
[11] F. Slimane, R. Ingold, S. Kanoun, A. M Alimi and J. Hennebert, "A New Arabic Printed Text Image Database and Evaluation Protocols", 10th International Conference on Document Analysis and Recognition, **(2009)**, pp. 946-950.
[12] M. S. Khorsheed and H. Al-Omari, "Recognizing Cursive Arabic Text: Using Statistical Features and Interconnected mono-HMMs", 4th International Congress on Image and Signal Processing, **(2011)**, pp. 1540-1543.

Authors

Perdana Adhitama, he received his B.S. degree in Informatics Engineering from Gunadarma University in 2008. From 2008 to 2011 he was working as a web developer in a software house company. In 2012, he continued his Master Degree at Chonnam National University in South Korea, majoring in Electronics and Computer Engineering. His main interest are in pattern recognition, information retrieval and web mining.

Soo-Hyung Kim, he received his B.S. degree in Computer Engineering from Seoul National University in 1986, and his M.S. and Ph.D degrees in Computer Science from Korea Advanced Institute of Science and Technology in 1988 and 1993, respectively. From 1990 to 1996, he was a senior member of research staff in Multimedia Research Center of Samsung Electronics Co., Korea. Since 1997, he has been a professor in the Department of Computer Science, Chonnam National University, Korea. His research interests are pattern recognition, document image processing, medical image processing, and ubiquitous computing.

In Seop Na, he received his B.S., M.S. and Ph.D. degree in Computer Science from Chonnam National University, Korea in 1997, 1999 and 2008, respectively. Since 2012, he has been a contract professor in Department of Computer Science, Chonnam National University, Korea. His research interests are image processing, pattern recognition, character recognition and digital library.

