TÜBİTAK

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2013) 21: 734 – 746

doi:10.3906/elk-1107-26

A word spotting method for Farsi machine-printed document images

Yaghoub POURASAD[1,*], **Houshang HASSIBI**[1], **Azam GHORBANI**[2]

[1]Department of Electrical and Computer Engineering, K.N. Toosi University of Technology, Tehran, Iran

[2]Department of Engineering, Saveh Branch, Islamic Azad University, Saveh, Iran

Received: 15.07.2011 • **Accepted:** 21.12.2011 • **Published Online:** 03.05.2013 • **Printed:** 27.05.2013

Abstract: In this paper, a word spotting approach for Farsi printed document images has been presented. The main idea of the paper is the font recognition of Farsi document images and query word modification according to the document image's font before searching. This operation increases the similarity between the query word image and its instances in the document image; therefore, the performance of the word spotting system increases. In the proposed word spotting approach, after the query word modification, the query word image rectangle is searched in the text lines of the document image using XNOR similarity measurement. In order to increase the recall rate, we considered an almost low value as an acceptance/rejection threshold (δ) and in order to increase precision rate, we used some other features, e.g., number of holes, ascenders, descenders, and dots. With multilevel matching and considering the mentioned features, the problem of justifying the operation (aligning the text to both the left and right) that occurs during the writing of Farsi documents has been solved. This approach was applied on a computer-made dataset consisting of 440 Farsi printed document images, and a precision rate of 97.5% at a recall rate of 92.1% was obtained. Moreover, when applying this approach on a dataset consisting of 224 Farsi scanned document images, a precision rate of 87.6% at recall rate of 79.3% was obtained.

Key words: Farsi document image, font recognition, word spotting, retrieval

1. Introduction

When there are text documents that have been scanned and are available only in image format, text search techniques are not applicable for searching through these document images. If we want to search for a keyword in these document images, we can first use optical character recognition (OCR) systems, in which the document images are converted into American Standard Code for Information Interchange (ASCII) texts, and then text search methods can be applied. However, OCR software cannot always transcribe document images to ASCII texts precisely, because when document images are degraded in quality and some adjacent characters are touching each other, the performance of the OCR software falls. Moreover, for the huge amount of document images archived in digital libraries, the OCR technique requires a very large amount of time. Therefore, for retrieval purposes such as word spotting, which is based on detecting and locating a few prototype/query images in the document image databases, resorting to full-scale OCR is wasteful and expensive. To overcome these problems, researchers have proposed another way, which is called keyword spotting. In keyword spotting methods, searching is done in the image domain without converting to text. Although there have been great attempts in producing OCR systems for the Farsi/Arabic language, such as those in [1,2], the overall performances of such systems are far from perfect.

*Correspondence: y_pourasad@ee.kntu.ac.ir

Most of the papers about word spotting have been presented for the English (Latin alphabet) language and some of them are for other languages such as Chinese [3], Korean [4], or Arabic [5–7]. So far, we have not seen any papers about Farsi spotting. There are only a few papers about Arabic word spotting, and most of these papers are about handwritten Arabic documents. For example, in [7], an algorithm and a system for searching handwritten Arabic documents were presented. In [5], a system for spotting words in scanned document images in 3 scripts, Devanagari, Arabic, and Latin, was described. In this system, the input query to the system can be either a word image or text. The candidate words that are searched for in the documents are retrieved and ranked using a normalized correlation similarity measurement. The features used for each word image are the gradient, structural, and concavity (GSC) features. In [8], a system for searching for keywords in Arabic handwritten historical documents using 2 algorithms, dynamic time warping (DTW) and the hidden Markov model (HMM), has been presented.

There are many statistical and structural features that are used in document image retrieval and word spotting systems, such as projection profiles, upper/lower word profiles, background-to-ink transitions [9], height, width, area, aspect ratio, moments, mean, variance, Fourier and wavelet transforms, gradient-based binary features (GSC) [5,6], holes, concavity, ascenders, and descenders [10]. Matching methods that are used in word spotting and retrieval systems fall into 2 groups: training-based and training-free methods. The training-based methods are mainly the k-nearest neighbors, HMM, and B-classification tree techniques. Training-free methods are divided into 2 groups: the shape code mapping and image-to-image matching approaches. In shape code mapping, a word or character is encoded into a relatively smaller set of predefined symbols, which is easier to recognize in comparison to the original character set. For the retrieval of document images and word spotting, earlier works are often based on the character shape coding (CSC) that annotates the image of the characters by a set of predefined codes [9]. The main limitation of CSC techniques is their sensitivity to the character segmentation error.

To overcome the limitations of CSC, word shape coding (WSC) schemes have been presented. In WSC approaches, instead of segmenting each word image to its characters, the word image itself is considered as a single component and its features are extracted. Hence, WSC approaches [10] are tolerant of character segmentation errors. In image-to-image matching, which is also called prototype matching, a keyword/prototype is matched with the target document images within a dataset. The matching can be either pixel-by-pixel or feature-based. In the pixel-by-pixel matching methods, prototype and target images are raster scanned and the distance between the corresponding pixel values is calculated. Minkowski distance measurement has been widely used for this type of matching, i.e. city block distance [4] and Euclidean distance [3]. In addition to the Minkowski distance measurement techniques, some other pair-wise image matching techniques have been reported. For example, Manmatha et al. [11] used the XOR operator for matching images. Another method of pixel-by-pixel matching is the sum of squared differences [12]. In the feature-based image-to-image matching, a fixed number of features are extracted and represented as feature vectors. Similarity between the prototype and target images is measured by comparing their corresponding feature vectors. Algorithms such as shape context [13], Scott and Longuet-Higgins [14], DTW [12,15,16], and correlation similarity measure [17,18] are the most important instances of feature-based image-to-image matching algorithms.

Because of the special characteristics of Farsi and Arabic scripts [1], the precision and recall rates of word spotting methods in these languages are low. Hence, in this paper, we propose a high-precision and high-recall spotting approach for Farsi scripts.

The rest of this paper is organized as follows: Section 2 describes our proposed method; Section 3 gives the results and discussion; and, finally, Section 4 gives the conclusion.

2. Proposed method

Machine-printed documents are different than handwritten documents. These documents certainly have been prepared with a special font face and font size. In our word spotting system, the used datasets consist of Farsi document images, where each of them is written in a special font face and font size and the font face and font size of each document does not vary throughout it. Understanding the font face and font size of a document can be very helpful in every document image analysis, especially keyword spotting. In keyword spotting, we want to find a keyword in a dataset of document images. Our aim is to find all of the various shapes of the same query keyword that exist in document images. This means that if a query image appears in different font faces, sizes, and styles in different documents, we must attempt to have a spotting system that can find all of them. This requires introducing some features that are invariant when the font face and font size of that word image varies. Finding such features for most of languages, especially for Farsi and Arabic scripts, is difficult and perhaps impossible, and for this reason, the precision and recall rates of these languages are low. Hence, we propose a new approach. When we want to search for a word through each document of the dataset, we first recognize the font face and font size of that document image, and then we modify the font face and font size of the query word according to the document's image of the keyword, which is constructed and searched for through the document image. Hence, if the keyword image exists in that document image, it will be completely

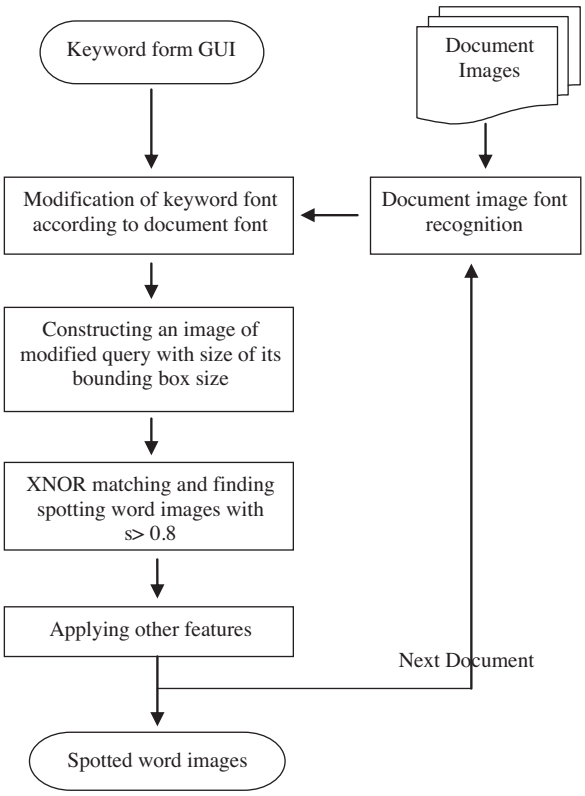

Figure 1. Flowchart of the proposed approach.

similar to the target word images in the document image. Therefore, all of the statistical and structural features of the keyword image and target word images in the document image will be completely the same. In this case, even the height and width of the query keyword image and target word images in the document image will be the same. There are many methods that can be used to match the 2 images, but we used the XNOR operation. The main reason for using XNOR matching is its precision. Indeed, the XNOR operation is a precise similarity detector and it can detect even a single pixel difference between 2 binary images. In Figure 1, a flowchart of the proposed approach is presented. Overall, word spotting systems consist of 3 main components: preprocessing, feature extraction, and similarity measurement. The proposed method consists of one component more than other methods, which is the document image's font face and font size recognition.

2.1. Preprocessing

Since the document images that exist in our databases are noiseless and without skew, they do not require operations for skew correction or noise removal. Moreover, there is no need for text line segmentation into words or word segmentation into the characters. Among preprocessing operations, we use only 2 of them: binarization and text line detection. Binarization uses Otsu's global thresholding method [19] and text line detection is done using a horizontal projection profile.

In our approach, there is no imperative need for text line detection, but we did it in order to reduce calculations and the number of matchings. There are many approaches to detect the location of text lines and one of them is the horizontal projection profile. In gray-scale document images, background pixels are usually white and their intensity values are near 255, while foreground pixels are dark and their intensity values are in the range (0,255) and near 0. We first changed the document image to its complement. If an m $\times$ n image is shown with f, its complement is:

$$\tilde{f} = \sum_{\substack{1 \le i \le m \\ 1 \le j \le n}} (255 - f(i,j)) \tag{1}$$

where m is the number of rows and n is the number of columns in the image. In $\tilde{f}$, background pixels are zeros. The horizontal projection profile of image f is:

$$HPP(i) = \sum_{j=1}^{n} \tilde{f}(i,j) \tag{2}$$

The indexes of the HPP, for which HPP > 0, indicate the location of the text lines.

In a binary text image f, often foreground pixels are 0 and background pixels are 1. In order to complement such an image, it is sufficient to change 0s to 1s and 1s to 0s. In the obtained image, $\tilde{f}$, the foreground pixels are 1. The horizontal projection profile of a binary image, f, is the number of foreground pixels in each row. For a binary image, the HPP vector is calculated using Eq. (2).

2.2. Font recognition and keyword modification

Almost all of the previously presented Farsi font recognition approaches [20,21] recognize only the font face of Farsi document images and do not recognize font size in them. In this paper, using the special features of Farsi documents, a new approach is presented that can recognize the font size of Farsi document images. Most of the Farsi characters (17 out of 32) have 1, 2, or 3 dots, which can be situated at the top, inside, or bottom of the characters. In fact, dot components are the most frequent components in each Farsi document image, and

rarely can one find a text line that does not have a dot component. Moreover, the extraction of dot components is very easy because their bounding box sizes are smaller and more frequent than the size of other connected components.

Most of the text documents that have been scanned and reserved in image form are gray-scale. In these documents, the background pixels are white and the foreground pixels that construct the letters are black.

Each character or dot that is written in a special font and font size has a special configuration of pixels with different intensity values. The configuration of the pixels of a character or dot is different from one font to another. For this reason, the behavior of a character or dot in a special font is different from the behavior of that character or dot written in different font during binarization. This point is very interesting, in that a dot or character written in a special font and font size represents predictable behaviors (shapes and sizes) during binarization with specified threshold values.

The proposed method recognizes the font face and font size of Farsi documents by analyzing the size of the bounding box of their single dot or double dot components after binarization.

In order to implement and evaluate our system, 2 datasets were constructed. The first dataset (dataset1) contained 560 Farsi printed document images (10 images for each state) and was used only for feature extraction in the font recognition system. The second dataset (dataset2) contained 440 Farsi printed document images and was used for font recognition and spotting systems evaluation.

'lotus', 'nazanin', 'mitra', 'yaghut', 'zar', 'homa', and 'koodak' are some of the most popular fonts in Farsi scripts, and so we focused on them. The font sizes that were considered in this paper were 8, 10, 12, 14, 16, 18, and 20. Therefore, we have 56 different states. It should be noted that 1 special font in 1 special font size is called 1 'state' in this paper. Each document image in both of the datasets was written in 1 font face out of 8 defined font faces and in 1 font size out of 7 defined font sizes.

To obtain the features of a document that was in 1 state out of 56 states, we first binarized it with a threshold value of T, where T was obtained from Otsu's global thresholding method. Next, the connected component labeling algorithm was applied to the binarized document. After that, a histogram of the components with sizes smaller than 7×6 was obtained. This is done easily using the "hist" command in MATLAB software. The experimental results show that in 56 described states, dots have bounding boxes with sizes smaller than 7×6. For this reason, we focused on the histogram of the components with sizes from 1×1 up to 7×6. In Figure 2, 4 histograms related to the size of the bounding box of the dot components for 4 states are illustrated. The horizontal coordinate is related to the size of the bounding box of the connected components. As is seen, the values of the horizontal coordinates are from 1 to 42. The numbers in the range of (1–6) in the horizontal coordinates are symbols for the connected components with sizes 1×1, 1×2, 1×3, 1×4, 1×5, and 1×6; and the numbers in the range of (7–12) are symbols for the connected components with sizes 2×1, 2×2, 2×3, 2×4, 2×5, and 2×6, respectively.

After analyzing all 10 histograms of each state, we described each state, S_i, with A_i and B_i vectors. For each state, the A_i vector elements are related to the indexes of the histograms where in all 10 histograms of that state, $H > 0$, where H is the histogram vector. In fact, the elements of the A_i vector are related to the size of the bounding box of the dots in the i*th* state. These dot components may be situated in the top, bottom, or inside the letters in a document. The B_i vector elements are related to the indexes of the histograms where $H = 0$ always. In fact, the elements of the B_i vectors are related to the sizes that dots or other symbols such as the comma, colon, or semicolon never have in that state. There are some indexes where sometimes $H = 0$ and sometimes $H > 0$. These indexes are related to some shapes similar to dots such as the comma or semicolon, which sometimes exist and sometimes do not exist in documents. These indexes are not reliable, and therefore

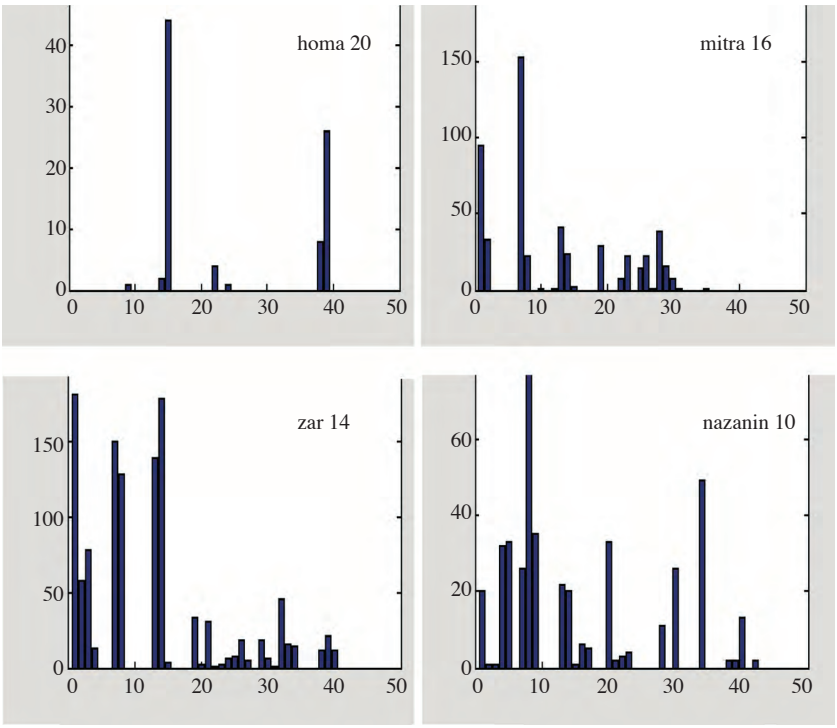

Figure 2. Histogram of the size of the bounding box of some fonts and font sizes.

we do not use them to describe states. In total, there are 42 indexes, but we use only the top 20 reliable indexes for each state. This means that if the length of A_i is m and the length of B_i is n, then m + n = 20. In fact, 20 is an experiential number that we use. For example, the A_i and B_i vectors of the states that are shown in Figure 2 are described below:

$$A_{\text{mitra 16}} = [1, 2, 7, 8, 13, 14, 19, 23, 25, 26, 28], B_{\text{mitra 16}} = [3, 4, 5, 6, 17, 18, 40, 41, 42],$$

where m = 11, n = 9.

$$A_{\text{nazanin 10}} = [1, 4, 5, 7, 8, 9, 13, 14, 20, 28, 30, 34, 40], B_{\text{nazanin 10}} = [10, 11, 24, 25, 26, 32, 33],$$

where m = 13, n = 7.

To have robust features, we used dataset1 and analyzed the histogram of every 10 images for each state. Often, the next 9 histograms in each state show the same behaviors as the first histogram of that state.

When a document image is entered into a system for font recognition, its histogram vector is constructed with the threshold value of T. We show it with QH. Next, the A_i and B_i vectors of all 56 states are compared with the QH, and for each state a distance D_i is calculated as:

$$d_i = \sum_{j=1}^{m} K_j, \tag{3}$$

$$K_j = \left\{ \begin{array}{lll} 1 & \text{if} & QH\left(A_i\left(j\right)\right) = 0 \\ 0 & \text{if} & QH\left(A_i\left(j\right)\right) > 0 \end{array} \right\}, \tag{4}$$

where m is the length of the A_i vector:

$$dz_i = \sum_{j=1}^{n} P_j, \tag{5}$$

$$P_j = \left\{ \begin{array}{lll} 1 & \text{if} & QH\left(B_i\left(j\right)\right) > 0 \\ 0 & \text{if} & QH\left(B_i\left(j\right)\right) = 0 \end{array} \right\}, \tag{6}$$

where n is the length of the B_i vector; and

$$m + n = 20. \tag{7}$$

The total distance between the i*th* state and QH is:

$$D_i = d_i + dz_i. \tag{8}$$

If a document image histogram (QH) is completely similar to the i*th* state, D_i will be 0, and if they are completely dissimilar, $D_i = 20$. After the calculation of D for all of the states, the state where D is the smallest among all of the Ds is recognized as the document image state.

When we want to search for a word in a document image, after recognizing the font face and font size of that document, first, the keyword is rewritten in the document's font face and font size by our system. This work is easily done using the "text" command in MATLAB software. Next, using the "getframe" command, an image is constructed from it. With the exception of the query word and its bounding rectangle, other parts of the image are deleted. The final query image prototype is only the query word and its bounding rectangle, which is named QI. All of these operations are done using MATLAB software.

2.3. Feature extraction and similarity measurement

In addition to the query image rectangle, which is constructed and used as a prototype, some other features are used as well. These features are the number of holes in that keyword, the number of dots and their position rather than the base line, and the number of ascenders and descenders. We applied a multilevel comparison to have precise spotting. The first-level XNOR function was applied between QI and the document image. In fact, QI acts as a window and is compared with the document image, pixel by pixel. In order to reduce comparisons, we first found the text line location and moved the query image window through all of these text lines, pixel by pixel. In each coordinate (x,y), a similarity value is obtained by applying the XNOR function between the query image and the document image in that coordinate. Both the XOR and XNOR functions can be used for matching purposes; XOR is a difference detector, while XNOR is an equality or similarity detector. If we have 2 images, a query image (QI) of size of i $\times$ j, and a part of the document image with the same size (DI), in a pixel with coordinate (x,y):

$$S(x,y) = \sum XNOR(QI, DI), \tag{9}$$

$$D(x,y) = \sum XOR(QI, DI). \tag{10}$$

S is the similarity value and D is the distance value between the 2 images. When we apply the XNOR function on 2 binary images and add all of the 1s of the resultant matrix, the result will be the number of corresponding pixels, which are equal in the 2 compared images. In this case, we can normalize the XNOR by dividing S by t, where t is the number of pixels of QI:

$$NS(x,y) = S(x,y)/t. \tag{11}$$

After applying normalized XNOR between the query prototype and all of the text lines of the document, pixel by pixel, for each pixel in the text line, a NS(x,y) is obtained. The value of NS in each coordinate, (x,y), indicates the value of the similarity between the query image and the phrase that is situated in that coordinate. Whenever the value of NS is higher, the similarity is greater. Certainly, the maximum value of NS is related to instances of the query in that document, if it exists. If we accept only the maximum value of NS as an acceptance/rejection threshold, some instances of the query, which has small variations, will be rejected and will not be spotted. Therefore, the recall rate will decrease. For this reason we considered a threshold value, δ, instead of the maximum value of NS. In this case, each candidate phrase in the document image with a corresponding NS value greater than δ is selected and goes to the next level of matching. In the next level, some other features are applied so that only all of the correct instances of the query images are spotted, and therefore both the precision and recall rates are increased. In Figure 3 this process is illustrated.

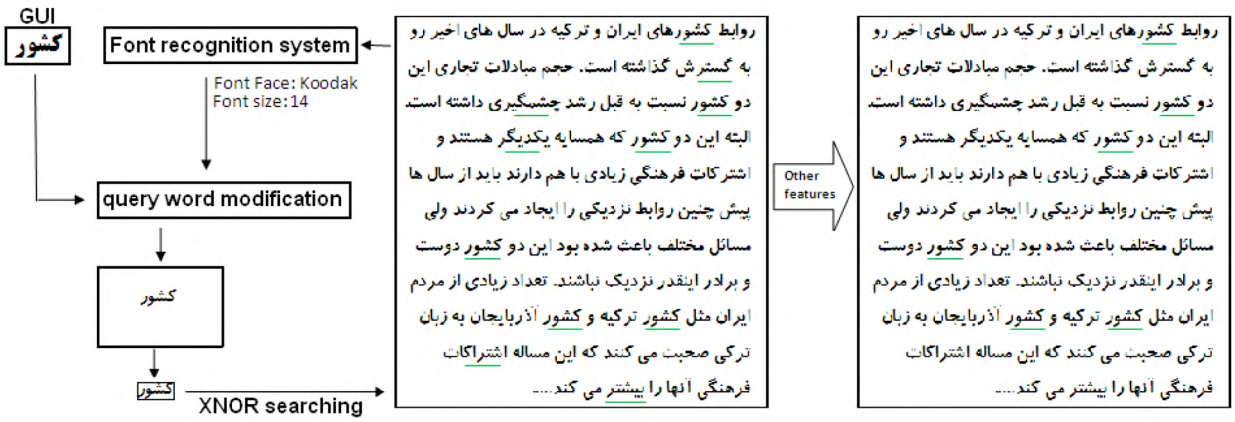

Figure 3. Spotting process levels for a Farsi query word.

The first feature that was considered is the number of holes that exist in the query word prototype. In many other previous works [10], holes have been used as features, but we think that if someone does not modify the font face and font size of the word before searching, this type of feature can be mistakable. This is because in different fonts and font sizes, the holes in the words present different behaviors. For example, in small sizes such as 8 and 10, some holes in the word images are filled and destroyed, and therefore are not extractable. Moreover, the font face affects the holes' behavior. However, when we modify the font face and font size of the query word according to the document's, the query word image will be completely similar to its own instance in the document image; therefore, the holes in both will be in displayed in the same conditions. This feature is very helpful and distinguishable because a large number of Farsi letters (10 out of 32) have at least 1 hole. The other used features for increasing the precision rate are the number of ascenders and descenders. In the Farsi alphabet, 6 letters out of 32 have ascenders and 18 letters out of 32 have 1 or 2 descenders. Therefore, these features play an important role in distinguishing the correct instances among the selected phrases from the document images. It should be noted that it is better to delete the dots of letters when we want to extract ascender or descender features. The last feature that is used is the number of dots above or under the base line of the words. In this feature, such as in the hole feature, modifying the font face and font size is extremely helpful, because the behavior of the dots in different fonts and font sizes is different. In Figure 4, Farsi letters that have one of the mentioned features are shown.

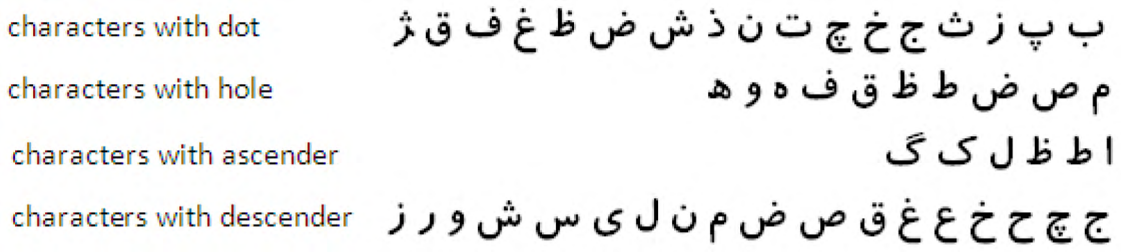

Figure 4. Farsi letters that have holes, dots, ascenders, and descenders.

In this approach, we could use more features, but because of the higher computational cost and time consumption, we avoided doing so. In Figure 5, some Farsi words and their features are shown.

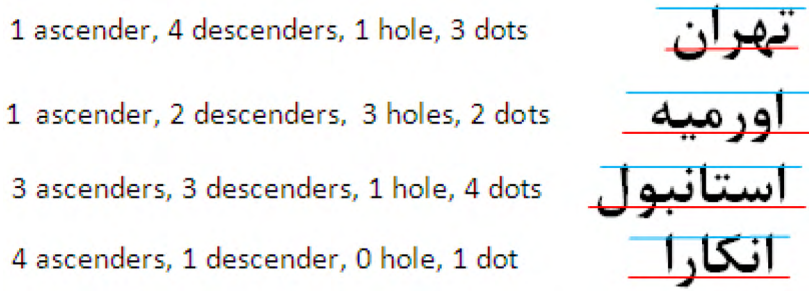

Figure 5. Some Farsi words and their features.

3. Results and discussion

This word spotting system has been implemented on a 2.4 GHz dual core Pentium PC. In this spotting system, a graphical user interface has been provided, where the user can enter a word for spotting. Moreover, 2 datasets (dataset1 and dataset2) were constructed using the computer (using Microsoft Word and Paint software). Dataset1 consisted of 560 document images and was used for feature extraction in the font recognition system. Dataset2 consisted of 440 documents and was used for evaluating both the font recognition and spotting systems. For dataset2, because it was used for the spotting system evaluation, its text equivalent file was constructed, as well. While evaluating the spotting system, first, we spotted the query word in dataset2 in the image domain, using our proposed spotting system; then we searched for it in the text equivalent of dataset2 in the text domain, using Microsoft Word software; and, finally, we compared the results of our system with the results of the Microsoft Word search.

After using dataset2 for the font recognition system, we observed that it recognized the font face and font size of the document images at a recognition rate of 94.6%. We investigated the effect of the errors of the font recognition system on the word spotting system. We observed that the main error in the font recognition system was related to cases where the system recognized an incorrect state in addition to the correct state. This means that the font recognition system recognized 2 states instead of only 1 correct state; in those cases, often, 2 font faces with the same font size were recognized. On the other hand, errors often occur in font face recognition and rarely in font size recognition. This problem does not have much of an effect on the spotting performance, because any doubts are between very similar font faces, and thus query prototypes constructed for a query word in both font faces are very similar to each other. Therefore, such doubt only has a small effect in the XNORing section and causes only a small decrease in the NS. To compensate for the effect of this decrease, we have considered a good tolerance for the NS; thus, the small reduction in the NS, influenced by the font

face error, does not disturb the spotting performance. When errors occur in font size recognition, this type of error can impair the spotting performance. One reason for our spotting system error is related to this issue. Another reason for errors in our spotting system is related to the query word rewriting and modification phase. This work is done using MATLAB software, in which the modified query word image has a small difference from its ideal and original case. We can solve this problem using Microsoft Word instead of MATLAB for the modification of the query word. However, this takes a long time, so we avoided doing that.

There are 2 criteria for evaluating spotting methods. One of them is precision (P) and the other is recall (R). They are defined as:

$$P = \frac{\text{No. of correctly retrieved words}}{\text{No. of all retrieved words}} \tag{12}$$

$$R = \frac{\text{No. of correctly retrieved words}}{\text{No. of correct words}} \tag{13}$$

Precision is a criterion of correctness and recall is a criterion of the completeness of the searching. In some papers [5], another evaluation criterion was defined as follows:

$$F1 = (2 \times P \times R)/(P + R). \tag{14}$$

Whenever P, R, and F1 are greater, our system works better. However, P and R are related inversely to each other. Increasing the value of P leads to a decrease in the value of R, and vice versa.

In order to evaluate our spotting system, we first applied only the XNOR operation without other features. The obtained results are shown in Table 1, where δ is the threshold value for rejecting or accepting a candidate word as a final answer. The resulting P, R, and F1 have been calculated by averaging over 150 selected keywords. These 150 keywords were from the most frequent words in our dataset and each of them was repeated at least 30 times throughout dataset2.

Table 1. Average precision, recall, and F1 for different threshold (δ) values, only with XNOR matching.

δ	0.7	0.75	0.8	0.85	0.9	0.95	1
P	0.593	0.693	0.752	0.810	0.894	0.918	0.932
R	1	0.985	0.95	0.921	0.849	0.760	0.650
F1	0.744	0.813	0.839	0.862	0.871	0.831	0.766

In Table 1, it is clear a lower threshold (δ) brings high recall but low precision. On the contrary, a higher threshold brings high precision and low recall. When δ is 0.9, the average F1 achieves the highest value.

In order to investigate the effect of other features (holes, dots, ascenders, and descenders), we applied these features after the XNOR operation and registered the results, which are shown in Table 2.

Table 2. Average precision, recall, and F1 for different threshold(δ) values, with XNOR and other features.

δ	0.7	0.75	0.8	0.85	0.9	0.95	1
P	0.681	0.792	0.881	0.975	0.984	0.990	0.996
R	0.97	0.961	0.935	0.921	0.841	0.760	0.650
F1	0.800	0.868	0.907	0.947	0.906	0.855	0.786

In Tables 1 and 2, the comparison results show that applying additional features (holes, dots, ascenders, and descenders) increases the P and F1 values and therefore the performance of the spotting system. Another point that is seen in Table 2 is that in order to have the highest F1, and therefore the highest performance, we should set $\delta = 0.85$.

One thing that causes the decrease in the recall rate, and the rejection of some correct instances of the query in the document images after the XNOR matching, is the justifying operation (aligning text to both the left and right). This operation is done by Microsoft Word or other typing software while typing the documents. This problem has not been considered in other retrieval or word spotting papers because justifying in Latin documents only leads to adding some spaces between characters or words and probably has no bad effect on the word spotting performance. However, in Farsi or Arabic, justifying leads to the extension of some words along their base line and the increasing in width of such words; therefore, the justified word shapes are different than the original shapes. This leads to the rejection of the justified words when they are searched for and the reduction of the recall rate. In our method, when we put $\delta = 1$, only the instances of the query in the document image that have not been justified are accepted as answers, whereas the justified words are rejected. This leads to an increase in the precision but a decrease in the recall and F1 values. However, if we decrease δ to 0.85, all of the justified and unjustified instances of the query word are selected as candidates. In the next level of matching, because the other features are applied, the incorrect words are rejected and only the justified and unjustified correct instances of the query word remain as answers. The first feature that we applied after the XNORing phase to reject the incorrect candidate words was the hole feature. Next, the number of ascenders, the number of descenders, and, finally, the number of dots were applied. These features caused only the correct instances of the query word, both justified and unjustified, to remain, and both the precision and recall rates to increase.

Unfortunately, for the word spotting of Farsi scripts, we could not find any papers in the literature with which to compare our results; therefore, we decided to compare our results with the language most similar to Farsi, which is Arabic. In [5], for English handwritten text, a precision rate of 60% was obtained at a recall rate of 50%. For Arabic handwritten text at the same recall, the precision rate was 68%. For Sanskrit printed documents, a precision rate as high as 90% was obtained at a recall rate of 50%. In [18], for a dataset containing 20,000 word images from handwritten documents with good quality, a 70% precision rate was obtained at a 50% recall rate. In [6], a database of 20,000 word images contained in 100 scanned Arabic handwritten documents was used and a 55% precision rate was obtained at a 50% recall rate. In [8], a dataset of 40 pages written in Arabic that included more than 8000 words was used; 10 of these 40 documents were Arabic printed documents with good quality. The authors of [8] did not report any value for the recall rate, but they reported a value of 85% as the average precision rate.

These papers are some of the best works that have been done so far in the field of Arabic word spotting, but as can be seen, although most of these works were applied on manually segmented documents, the rates of precision and recall in all of these papers were low. This drawback is because of the special characteristics of Arabic and Farsi scripts. These scripts are cursive both in handwritten and printed cases; therefore, segmentation of the lines to words and the words to characters is a difficult problem. Moreover, finding features that are constant and invariant in different fonts and font sizes for these languages is nearly impossible.

Our proposed method at a 50% recall certainly represents 100% precision. As is seen in Table 2, our system presents its best performance at $\delta = 0.85$. In this case, we have a high precision rate of 97.5% at a high recall rate of 92.1%. This result is the best result among all of the Arabic spotting systems.

There are some reasons for our system's high precision/recall. The most important of these is that our method modifies the query word according to the document's font before searching. In other word spotting approaches, this work has not been done. Moreover, our approach does not require a segmentation process, and therefore has no segmentation error.

In addition to dataset1 and dataset2, which were constructed using a computer, another dataset (dataset3) was constructed, consisting of 224 Farsi document images. In this dataset, the Farsi document images were printed and scanned at a resolution of 150 dpi carefully, without noise and skew. The reason for such carefulness while printing and scanning is that we guessed that the skew and noise, especially the salt and pepper noise, may have a bad effect on our approach. We constructed dataset3 in order to investigate the performance of our proposed approach while applying it to real scanned documents. We observed that our approach is applicable on real scanned Farsi documents. In this case, the recognition rate of our font recognition system was 91.51%. This value is smaller than the value that was obtained for the computer-made dataset (dataset2). This is because we only used 112 document images for the feature extraction, while for the computer-made dataset, we used 560 document images for the feature extraction. If we increase the number of scanned documents, the recognition rate will certainly become greater. While evaluating the spotting system, we observed that the best result was obtained for $\delta = 0.78$, where the precision rate was 87.6% at a recall rate of 79.3%.

In addition to the high precision/recall rates, our method has another advantage, which is that it does not require common high-resolution document images such as 300 dpi, and it can operate correctly on low-resolution document images such as 150 dpi.

4. Conclusion

Printed documents are different than handwritten documents. These documents are written in one standard font face and font size. Understanding the font face and font size of a document image can help us in keyword spotting, but this issue has not been considered in any of the keyword spotting approaches. In all of the keyword spotting approaches, the aim is to describe a keyword image with a set of comprehensive and font-invariant features, and then search within the document images to find target words with the same features. However, finding such features for some languages such as Farsi and Arabic is difficult and perhaps impossible. For this reason, the precision and recall rates of the word spotting methods in these languages are low. Hence, we present a different approach. In this method, when a user enters a keyword to be searched for in a document image, the font and font size of the document are recognized, the keyword is rewritten in the document's font and font size, an image is constructed from the modified keyword, and the resulting image is searched for through the text lines of the document image using the XNOR similarity measurement. Finally, by applying some other features, e.g., the number of holes, dots, ascenders, and descenders, only the correct instances of the query word are found. The proposed approach has 2 components more than other spotting approaches: document font face and font size recognition, and keyword modification. This approach has been applied on a computer-made dataset that included 440 Farsi printed document images in 8 fonts and 7 font sizes, and it has presented a high precision rate of 97.5% at a high recall rate of 91.2%. Moreover, this approach has been tested on a dataset consisting of 224 scanned Farsi documents and has presented a precision rate of 87.6% at a recall rate of 79.3%. Our system has been applied on only 8 font faces and 7 font sizes, but it is easily expandable for more fonts and font sizes.

References

[1] Y. Pourasad, H. Hassibi, M, Banaeyan, "Persian characters recognition based on spatial matching", International Review on Computers and Software, Vol. 6, pp. 55–59, 2011.

[2] R. Mehran, H. Pirsiavash, F. Razzazi, "A front-end OCR for omni-font Persian/Arabic cursive printed documents", Proceedings of Digital Image Computing, Techniques and Applications, pp. 385–392, 2005.

[3] L. Yue, C.L. Tan, "Chinese word searching in imaged documents", International Journal of Pattern Recognition and Artificial Intelligence, Vol. 18, pp. 229–246, 2004.

[4] H.K. Soo, C.P. Sang, B.J. Chang, S.K. Ji, H.R. Park, S.L. Guee, "Keyword spotting on Korean document images by matching the keyword image", Proceedings of the 8th International Conference on Asian Digital Libraries: Implementing Strategies and Sharing Experiences, Vol. 3815, pp. 158–166, 2005.

[5] S.N. Srihari, H. Srinivasan, C. Huang, S. Shetty, "Spotting words in Latin, Devanagari and Arabic scripts", Indian Journal of Artificial Intelligence, Vol. 16, pp. 2–9, 2006.

[6] S.N. Srihari, H. Srinivasan, P. Babu, C. Bhole, "Spotting words in handwritten Arabic documents", Proceedings of the SPIE, pp. 606702-1–606702-12, 2006.

[7] S.N. Srihari, H. Srinivasan, P. Babu, C. Bhole, "Handwritten Arabic word spotting using the CEDARABIC document analysis system", Proceedings of Symposium on Document Image Understanding Technology, pp. 123–132, 2005.

[8] R. Saabni, J. El-Sana, "Keyword searching for Arabic handwritten documents", Proceedings of the 11th International Conference on Frontiers on Handwritten Recognition, pp. 271–277, 2008.

[9] A.L. Spitz, "Using character shape codes for word spotting in document images", Shape, Structure and Pattern Recognition, pp. 22–27, 1994.

[10] S. Lu, C.L. Tan, "Retrieval of machine printed Latin documents through word shape coding", Pattern Recognition, Vol. 41, pp. 1816–1826, 2008.

[11] R. Manmatha, C. Han, E.M. Riseman, "Word spotting: a new approach to indexing handwriting", Proceedings of Computer Vision and Pattern Recognition Conference, pp. 631–637, 1996.

[12] R. Manmatha, T.M. Rath, "Indexing of handwritten historical documents - recent progress", Proceedings of the Symposium on Document Image Understanding Technology, pp. 77–85, 2003.

[13] S. Belongie, J. Malik, "Matching with shape contexts", IEEE Workshop on Content-Based Access of Image and Video Libraries, pp. 20–26, 2000.

[14] G.L. Scott, H.C. Languet-Higgins, "An algorithm for associating the features of two patterns", Proceedings of the Royal Society of London, Vol. B224, pp. 21–26, 1991.

[15] T.M. Rath, R. Manmatha, "Word spotting for historical documents", International Journal of Document Analysis and Recognition, Vol. 9, pp. 139–152, 2007.

[16] T. Konidaris, B. Gatos, K. Ntzios, I. Pratikakis, H. Theoridis, S.J. Perantonis, "Keyword-guided word spotting in historical printed documents using synthetic data and user feedback", International Journal of Document Analysis and Recognition, Vol. 9, pp. 166–177, 2007.

[17] R. Manmatha, J. Rothfeder, "A scale space approach for automatically segmenting words from historical handwritten documents", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 27, pp. 1212–1225, 2005.

[18] B. Zhan, S.N. Srihari, C. Huang, "Word image retrieval using binary features", Document Recognition and Retrieval XI, Proceedings of the SPIE, Vol. 5296, pp. 45–53, 2004.

[19] N. Otsu, "A threshold selection method from gray-level histograms", IEEE Transactions on Systems, Man and Cybernetics, Vol. 9, pp. 62–66, 1979.

[20] H. Khosravi, E. Kabir, "Farsi font recognition based on Sobel-Roberts features", Pattern Recognition Letters, Vol. 31, pp. 75–82, 2010.

[21] Y. Pourasad, H. Hassibi, M. Banaeyan, "Farsi font recognition based on spatial matching", 18th International Conference on Systems, Signals, and Image Processing, pp. 1–4, 2011.