Optical Engineering 48(11), 117205 (November 2009)

Integrated segmentation and recognition of connected Ottoman script

Ismet Zeki Yalniz
Ismail Sengor Altingovde
Uğur Güdükbay
Özgür Ulusoy
Bilkent University
Department of Computer Engineering
Bilkent, Ankara, 06800
Turkey
E-mail: gudukbay@cs.bilkent.edu.tr

Abstract. We propose a novel context-sensitive segmentation and recognition method for connected letters in Ottoman script. This method first extracts a set of segments from a connected script and determines the candidate letters to which extracted segments are most similar. Next, a function is defined for scoring each different syntactically correct sequence of these candidate letters. To find the candidate letter sequence that maximizes the score function, a directed acyclic graph is constructed. The letters are finally recognized by computing the longest path in this graph. Experiments using a collection of printed Ottoman documents reveal that the proposed method provides >90% precision and recall figures in terms of character recognition. In a further set of experiments, we also demonstrate that the framework can be used as a building block for an information retrieval system for digital Ottoman archives.
[DOI: 10.1117/1.3262346]

Subject terms: optical character recognition; historical document analysis; connected scripts; information retrieval.

Paper 090628R received Aug. 17, 2009; revised manuscript received Sep. 18, 2009; accepted for publication Sep. 19, 2009; published online Dec. 2, 2009.

1 Introduction

Ottoman archives include a wealth of historical documents that shed light on many aspects of an empire that shaped the direction of the "old world" for several centuries. Not surprisingly, providing a means of electronic access for even a subset of this huge collection of handwritten and printed documents would be an exciting and worthwhile development for researchers from several disciplines and countries. Because of recent advances in the hardware and a reduction in the associated costs, digitizing collections of historical documents, or even entire books, becomes a more attainable goal (see Ref. 1, for example). Accordingly, a growing body of Ottoman archives is being digitized by the State Archives Office of Turkey, where the majority of this cultural heritage resides.[2] Digital images of historical Ottoman documents are being created for the purposes of long-term storage and electronic access.

A robust and effective character-recognition approach is the first stage of providing automatic access and sophisticated search and retrieval functions for such textual image archives. In this paper, we describe an integrated segmentation and recognition method for connected letters in digital Ottoman documents. Ottoman has several common features with Arabic; however, it also has significant differences, which disallows using off-the-shelf Optical Character Recognition (OCR) software. Furthermore, even for Arabic, OCR is a problem that is not completely solved; the recognition rates reported in the literature are less successful in comparison to those of other languages based on, for example, Latin characters.[3] Because of the difficulty of the problem, we focus on the printed Ottoman script in this study. Note that the printed documents in the Ottoman archives mostly belong to the last eras of the empire and are found in different fonts and styles, depending on the technology of the age in which they were created. Thus, digital recognition of letters from these printed documents is not a trivial problem and can serve as a first cut to fuel further research in this area.

The proposed recognition method first extracts all connected components from a document. A connected component may be a dot, diacritic, single letter, or connected group of letters. Dots and diacritics are discarded for simplification purposes in this study. For each of these remaining connected components containing letter(s), a set of (possibly overlapping) segments are obtained by applying sliding windows of varying sizes. All the segments of a connected component are compared, one by one, to the letters in the letter library. The letter that produces the highest similarity score is selected as the candidate letter of its corresponding segment. Then, a subset of these segments is selected according to their similarity scores to their candidate letters and forwarded to the recognition stage. It should be noted that sliding windows on connected components has nothing to do with determining actual letter boundaries, but it simply extracts a number of segments within predefined width ranges, some of which can possibly convey letters. The determination of the actual letter boundaries is postponed to and interleaved with the recognition stage.

In the recognition stage, the aim is finding a sequence of candidate letters whose segments formulate the connected component the best. For this purpose, we define a score function to rank the possible sequences of candidate letters whose segments that do not overlap in the connected component and can precede each other syntactically, and

0091-3286/2009/$25.00 © 2009 SPIE

choose the sequence with the highest score as the final recognition. Ideally, segments of selected candidate letters in the sequence should fully span the connected component and the total width of these segments must be smaller than or equal to the width of the connected component. Thus, the score function takes into account the width of the segment, the similarity score to the candidate letter, and the unigram and bigram frequencies of the consecutive letters in a sequence.

The candidate letter sequence maximizing the score function is efficiently computed by constructing a directed acyclic graph. Each candidate letter corresponds to a node in the graph. An edge connects two nodes if one of the candidate letters precedes the other in the connected component from which they are extracted. Edge weights are based on the score function. Finally, the letters are recognized by computing the longest path in this graph.

Our experiments with a set of printed Ottoman documents reveal that the proposed method for segmenting and recognizing letters resulted in precision and recall figures of $>90\%$. Because further experiments indicated that resulting OCR errors have limited effect on information retrieval (IR) performance, the proposed framework can be used as a building block for an IR environment for printed digital Ottoman archives.

In what follows, we first describe various properties of Ottoman script. In Section 3, we review the related studies. Section 4 describes the proposed segmentation and recognition framework. In Section 5, we discuss an IR framework that is built on the output of the segmentation and recognition process. Section 6 presents the experimental evaluation of both the recognition and retrieval stages. Finally, in Section 7, we conclude and point to future research directions.

2 Ottoman Script

Ottoman and Arabic scripts have many common characteristics. Ottoman and Arabic are read from right to left. Most letters in the alphabets are the same, with Ottoman having five additional letters.[4] The Ottoman alphabet, without its diacritics and dots, is shown in Fig. 1. It should be noted that a letter may have one of four different forms according to its position in a word, namely, being the beginning, medial, or end letter in a word, or being isolated. Looking at Fig. 1, it can be observed that some letters repeat themselves. This is because the form of a letter at one position may be the same as its form at another position or a form of a different letter. Some of these letter repetitions are caused by the removal of the diacritics and dots. Another slight modification to the alphabet has been done by removing letters that can be constructed by using other letters. For instance, in Fig. 1, the first letter in line two (reading left to right) can be constructed from the second and fourth letters on the same line. After eliminating such letters, 48 distinct letters remain; these are referred to as the letter library in this paper. These letters cover 98% of shape occurrences for the test data set used in the experiments. The remaining 2% of shapes consist of characters that are written on top of each other according to the Ottoman syntactic rules.

In Ottoman script, each connected component—except dots and diacritics—is considered to be a subword and must involve one isolated letter, or a group of letters that

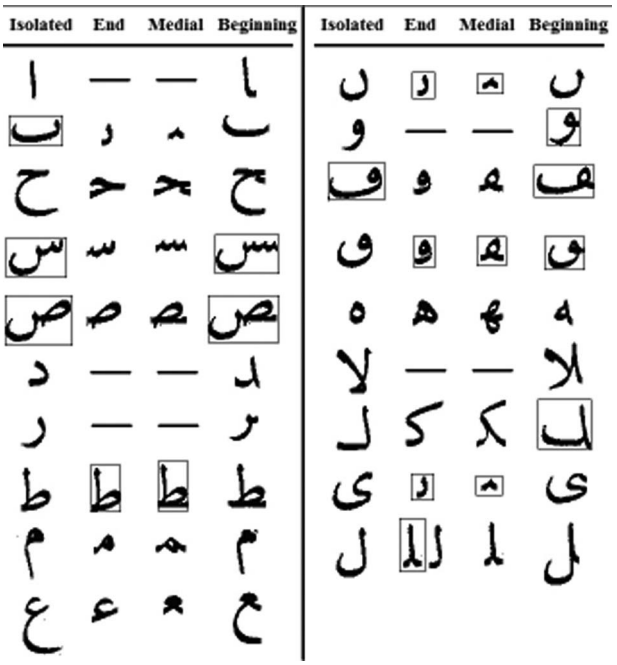

Fig. 1 The Ottoman alphabet without diacritics and dots. Letters in the rectangles are either repeated letters or can be formulated by the other letters, and thus, they are not included in the library.

starts with a letter of the beginning form, continues with zero or more letters of the medial form, and finishes with a letter of the end form. One or more subwords form an Ottoman word. Figure 2 shows a word that has five connected components. One of these connected components contains three letters. Three of them represent single letters. The remaining one is just a dot.

3 Related Work

OCR has been studied extensively. Promising solutions have been proposed for various scripts with different characteristics.[5] However, in the literature, there are relatively few works that attack the character recognition problem for Ottoman script. This may be mostly due to the fact that Ottoman is not a currently spoken language. Ottoman is only captured in the historical documents and attracts the interests of scholars. Furthermore, until recently, there were relatively few digital Ottoman documents available. On the other hand, Arabic and Farsi, which share some features with Ottoman, is a language still spoken by millions of people all over the world. We first briefly review the works for character recognition in Arabic documents. Next, we provide a more detailed discussion of the works focusing on character recognition and automatic retrieval for Ottoman documents.

Fig. 2 The Ottoman word "Ensari."

Comprehensive surveys[3,6–9] on off-line Arabic and Farsi character recognition discuss a variety of approaches adapted for segmentation and recognition, including methods based on local and global features, neural networks, graph-based algorithms, stochastic methods, such as hidden Markov Models (HMMs), and others. Arabic and Farsi OCR, especially for highly degraded and/or handwritten documents, is still an open research area and current results are inconclusive because they are provided on small data sets that are not available to others. Recently, promising results are also reported.[10–14] The method described in Ref. 14 has some similarities to our approach. In both works, the segmentation is achieved in an interleaved manner with the recognition. However, the actual recognition in their work employs the HMM, whereas we use a graph-based model.

Öztürk et al. apply a neural network recognition approach to Ottoman characters.[15] Although the recognition rate is said to be high, both the training and testing stages seem to be applied on manually segmented characters; thus, the segmentation problem of the connected letters is not attacked.

In Ref. 4, a retrieval system for Ottoman documents is proposed that involves first segmenting lines and words in a document and then comparing words, as a whole, for the querying purposes. In this approach, word comparisons are performed by using quantized vertical projection profiles. In a more recent work,[16] querying Ottoman documents is considered as an image retrieval problem. More precisely, each word image in a document is represented by a set of visual terms, which are obtained by the vector quantization of scale-invariant feature transform (SIFT) descriptors extracted from salient points. Words are matched by comparing the similarity of these visual terms. In both works, queries can only be constructed by finding examples of the words over sample documents. Because there may be rare words that a user may want to search for, this method can be a time-consuming task for the user.

Şaykol et al. propose an effective method for the compression and content-based retrieval (CBR) of Ottoman documents.[17] In their work, instead of a static character library as in the typical practice of OCR methods, a dynamic library of symbols is constructed. The construction process begins with an empty library, and each component extracted from the document is compared to the current members of the library to see if it is (or it contains) an already discovered symbol. If the symbol already exists in the library, then the location of the occurrence is recorded to the document's codebook. If it is a new symbol, then it is added to the library, as well. Of course, this comparison stage includes further complexities to ensure that the symbols occurring in the connected components can be correctly deduced. Once the codebooks are constructed, the user queries can be processed. The number of symbols in the dynamic symbol library is likely to increase as more documents are processed. Consequently, when thousands of symbols exist in the library, processing new documents becomes inefficient. Once the codebooks are constructed, user queries can be processed, but only through query by example (QBE). That is, the user must find an instance of the query word in the processed documents.

In another study, we adapted the symbol-segmentation approach proposed in Yalniz et al.[18] and employed a static character library for a more traditional OCR task (i.e., where only textual images are considered and documents involving figures, drawings, decorations, etc., are discarded).[19] It uses a greedy segmentation and recognition approach for connected letters. In a nutshell, this algorithm slides several windows of varying sizes over each connected component, and for each window, it computes the similarity between the segment extracted by the window and letters in the library. The highest-scoring segment is then decided to be the correct recognition, and it is removed from the connected component. The same approach is then recursively applied for the remaining parts of the connected component, until the component is totally consumed. Queries can be posed by either using a virtual keyboard designed for Ottoman script or by selecting a query region in processed documents (QBE).

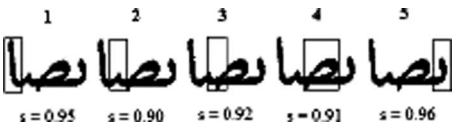

Fig. 3 Illustration of the Greedy approach.

The Greedy segmentation of letters may not always provide optimum results. Because of the greedy nature of the algorithm, an incorrect choice, especially at the earlier stages of the recognition, may propagate to the following stages and significantly reduce the overall recognition rate. For instance, in Fig. 3, we illustrate how the Greedy approach may fail while segmenting and recognizing the letters from a connected component that actually includes four letters. In Fig. 3, only five windows are plotted over the connected component for simplification purposes. Similarities of the segments in these windows to their matching letters are also given. The third window would be a false segmentation because it is over two medial letters and overlapping with windows 2 and 4, whereas windows 1, 2, 4, and 5 exactly fit on actual letters in our letter library. At each iteration, the Greedy approach finds the window with highest similarity and removes it from the connected component. The iterations continue until the component is consumed. For this example, extracted windows are 5, 1, and 3 in their extraction order. The Greedy approach ignores windows 2 and 4 because their overlapping regions with window 3 are already removed, and consequently, letters in windows 2 and 4 remain unrecognized.

Furthermore, the Greedy approach cannot use statistical information such as letter-occurrence frequencies and *n*-gram probabilities. To overcome these problems, we propose a new method that has almost the same complexity as the Greedy approach but postpones the recognition decision to the end. This allows finding optimum letter boundaries and exploit occurrence probabilities among letters.

4 Segmentation and Recognition Framework

In Fig. 4, we illustrate the stages of the proposed segmentation and recognition framework. We start by extracting connected components from the Ottoman documents and then obtain a set of possible segments from a component by applying sliding windows of varying sizes with small sliding intervals. The purpose of sliding windows is not actu-

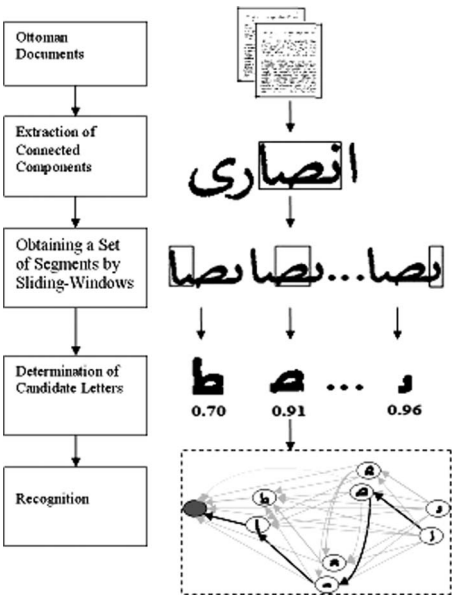

Fig. 4 Stages of the proposed segmentation and recognition approach.

ally to determine the real letter boundaries, which will be achieved along with recognition in the final stage. For each extracted segment, a number of features, such as the segment's aspect ratio and distance and angular span vectors, are computed. Each segment is compared to the letters in the library and matched to a candidate letter that yields the highest similarity score. Finally, by using a graph-based language model, we can efficiently compute the sequence of the candidate letters maximizing the score function, thus achieving recognition.

4.1 *Extracting and Segmenting Connected Components*

We define a connected component as a connected group of black pixels in the document image. Connected components are extracted using the approach outlined in Ref. 20; that is, the document is scanned from left to right and top to bottom. Whenever a black pixel is encountered a four-connected boundary-detection algorithm is employed to obtain the bitmap of the component and remove it from the original document. It should be noted that nested components, diacritics, and dots are detected separately. For the purposes of this study, we have restricted ourselves to recognizing letters without diacritics and dots (see Fig. 1). Diacritics and dots are relatively small connected components placed under or over the letters, and the ratios of their width and height to the line height do not vary significantly for printed documents with different fonts. Therefore, it is possible to discard them successfully by simply tuning the thresholds for the ratios mentioned above while ensuring that valid character components are not removed. This operation also removes some of the noise in the document images.

The line-height information is extracted from documents while connected components are being identified. This is achieved by using the horizontal projection profile of the document (see Fig. 5). Upper and lower boundaries for

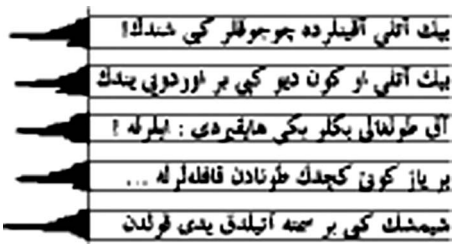

Fig. 5 Determining the line height information.

each line are found by tracing horizontal projection vectors and detecting peaks. In this way, all connected components learn the height of the line that they belong to. The ratio of the letter height to the line height is later used during the determination of candidate letters. It should be noted that page skew is avoided manually in this step. Advanced document analysis techniques for skew detection and text line segmentation are studied extensively in the literature.[21]

Once the connected components are extracted, the next step is to obtain an initial set of segments—some of which can correspond to actual letters—by sliding windows with varying sizes over the bitmap of the connected component. In order to avoid redundant computations during sliding windows, the minimum ($W_{\min}$) and maximum ($W_{\max}$) width of the sliding window and the sliding interval (ΔW) are defined. It is observed that, for a particular letter in the library, the ratio of letter widths to the corresponding line heights does not vary significantly in printed Ottoman documents. Therefore, it is possible to set these constants for sliding windows automatically for each line using the line-height feature. The height of a line actually designates the font size of its characters. In particular, $W_{\min}$, $W_{\max}$, and ΔW are respectively set to 7, 70, and 5% of the line height of the connected component. If any of these quantities are 1, then it is simply set to 1. Ratios 7 and 70% are set according to the width property of the thinnest and the largest letters in the letter library, respectively. Using these values, it is seen that OCR performance is almost unaffected as the processing time per document has fallen drastically.

The content of each window moving over the connected component is stored as a segment, along with a number of features. It should be noted that while some segments may correspond to a single letter (which is the preferred case), others may include parts from one or more letters. The latter type of erroneous segments will be eliminated in the candidate letter-determination stage.

4.2 *Extraction of Segment Features*

A number of features for each segment are computed, as described follows:

1. Segment aspect ratio is the ratio of the width to the height of the extracted segment.
2. Segment height ratio is the ratio of a segment's height to the height of the line to which it belongs.
3. Angular span vector is the number of black pixels in θ-deg slices centered at the center of mass with respect to the horizontal axis.

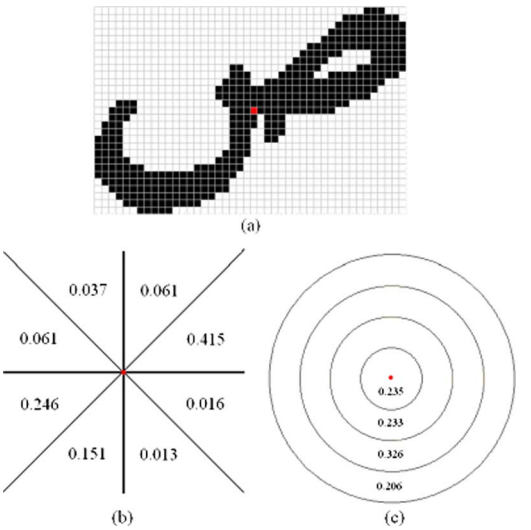

Fig. 6 (a) A letter, (b) angular span of the letter, and (c) distance span of the letter.

4. Distance span vector is the number of black pixels between concentric circles centered at the center of mass with radii of r, $2r$, $3r$, etc.

The first two features are useful for pruning the search space during the candidate letter-determination stage. These features are highly descriptive for Ottoman character recognition, especially for printed documents. The latter two vectors are the spatial domain features described in Ref. 17. The entries of these vectors are normalized by the area, which is the total number of black pixels of the segment. Angular and distance span features are employed in this work because they are not sensitive to font variation and letter thickness, and are scale and rotation invariant. Rotation invariance of the angular span vector can be achieved simply by shifting vector entries by one slice to the left and right. In this work, angular and distance span vectors of the sizes 12 and 8, respectively, are used. Figure 6 illustrates these vectors for an example letter from the library.

4.3 *Determining Candidate Letters*

During candidate letter determination, the similarity of each segment to the letters in the library is computed one by one, and the top score for each segment (and the corresponding letter) is stored. If the highest similarity score for a segment is less than a threshold value, then this segment is discarded because it may include a part of a letter or more than one letter. Otherwise, the best-matching letter for that particular segment is called the "candidate letter," and it is stored along with the similarity score and segment width to be used in the final recognition stage.

Recall that, a letter can have four different positions, and for each position, it can have different shapes in Ottoman script. This fact can be exploited to reduce the number of similarity computations between a segment and the letters in the library. More specifically, for each segment, we keep track of its relative position type (i.e., beginning, medial, end, or isolated) with respect to the component from which it is extracted and compare the segment to only those letters that can appear at that particular position.

We adapted the histogram intersection technique[22] for measuring the similarity between two feature vectors $\mathbf{H}_1$ and $\mathbf{H}_2$, for an extracted segment and a letter in the library, respectively,

$$\text{Similarity}(\mathbf{H}_1,\mathbf{H}_2) = \frac{\sum_{i=1}^{n} \min(\mathbf{H}_1[i],\mathbf{H}_2[i])}{\min(|\mathbf{H}_1|,|\mathbf{H}_2|)}, \quad (1)$$

where $|\mathbf{H}|$ denotes the sum of all entries of vector $\mathbf{H}$. Here, the range for the similarity measure is [0, 1], where 1 means that the vectors are the same and 0 means that they are totally different.

The overall similarity of the segment S and the letter L is computed as an equally weighted (i.e., 0.5) linear sum of the histogram intersection scores for the distance and angular span vectors. There are two more features involved in the similarity calculation. These are the aspect ratio of the segment and the ratio of the segment's height to the height of the line to which it belongs. If at least one of these ratios for S is different from the corresponding ratio for L by a predefined threshold, the similarity of S and L is set to 0 without further computation.

4.4 *Recognition*

Given a set of segments and corresponding candidate letters, our goal in the recognition stage is to define a function for scoring each different syntactically correct sequence of these candidate letters, and to choose the sequence that maximizes this function. To this end, we first formally define the scoring function, which also exploits letter statistics, and then describe a graph-based model to find the sequence that maximizes the scoring function.

Recall that segments and corresponding candidate letters obtained from a particular connected component can have varying sizes, overlapping boundaries and various similarity scores. Some of these segments may also correspond to intermediate matches. Thus, while determining one sequence of these candidate letters as the final recognition, we want to satisfy the following goals to the greatest extent possible: (*i*) the selected set of candidate letters should cover all of the connected component, (*ii*) the boundaries of the candidate letters should not overlap, (*iii*) the candidate letters should have high similarity scores, and (*iv*) the probability of letters being consecutive in the sequence should be as high as possible. Remarkably, it may not be possible to maximize all these goals at the same time. For instance, the two candidate letters with the highest similarity scores may have overlapping boundaries. Or, according to the rules of Ottoman script, it may be impossible for these letters to appear consecutively (e.g., both letters may be in the form that can only appear at the beginning of a component). Thus, the overall problem can be seen as a maximization expressed in the form of the score function in

$$\text{Score} = \sum_{i=1}^{n} [w_i \times s_i + P(l_i|l_{i-1}, l_{i-2}, \ldots, l_0) \times c] + [w_0 \times s_0 + P(l_0) \times c]$$

such that End-X-coordinate$(i-1)$ < Start-X-coordinate(i). (2)

In Eq. (2), w_i denotes the ratio of the width of the candidate letter i to the width of the connected component from which it is extracted, and s_i denotes the similarity score of this candidate letter. $P(l_i|l_{i-1}, \ldots, l_0)$ denotes the probability of encountering the letter l_i after seeing a sequence of letters l_0 to l_{i-1}. Finally, c is a constant $(0<c<1)$, that assigns weight for the letter statistics. It should be noted that these three elements of the equation maps to the goals (i), (iii), and (iv) discussed above. The constraint stating that the end x coordinate of the $(i-1)$th letter should be less than the start x coordinate of the ith letter corresponds to the goal (ii), and, as long as this constraint is not violated, the new candidate letters can be added to the current sequence. Typically, the number of letters in a sequence (n) would be fewer than the total number of candidate letters extracted from the connected component. Because it is impractical to obtain the all probabilities $P(l_i|l_{i-1}, \ldots, l_0)$, we use the unigram and bigram frequencies of the letters, which can be learned even from a small dataset, and approximate Eq. (2) with Eq. (3). The sequence of candidate letters that maximizes (3) can be found in polynomial time by using a graph-based model

$$\text{Score} = \sum_{i=1}^{n} [w_i \times s_i + P(l_i|l_{i-1}) \times c] + [w_0 \times s_0 + P(l_0) \times c]. \quad (3)$$

The graph is constructed as follows: each candidate letter corresponds to a node in the directed acyclic graph. An edge from node i to node j is added if these candidate letters do not overlap and if the position of j precedes i (as Ottoman is written from right to left) in the component from which they are extracted. The syntactical constraints for letter forms in a connected component are also considered. For instance, no edges can exist between any two candidate letters that are in either the beginning or the end form. There is no incoming edge for a letter in the isolated form, and the only outgoing edge is to the final node. Finally, no edge can occur from a letter in the end or medial form to a letter in the beginning form.

The weight assigned for an outgoing edge is $w_i \times s_i + P(l_i|l_{i-1}) \times c$, where $P(l_i|l_{i-1})$ is the probability of the letter sequence implied by the transition. There is an additional node in the graph called the "final node," which is needed for transforming the problem into a simple search for the longest path. The final node has incoming edges from all nodes and no outgoing edges. The edges that arrive at the final node are assigned the weight $w_i \times s_i + P(l_0) \times c$, because there is no bigram frequency for the letter sequence arriving to this special node. In this way, it is still possible to recognize letters of subwords whose connected components are divided into separate pieces because of low document quality. Figure 7 gives the algorithm for graph construction. Remarkably, the construction procedure outlined here yields a topologically sorted graph. Once the graph is constructed, we find the longest path in this graph. The resulting path, which maximizes (3), includes the sequence of recognized letters for a particular connected component.

```
1.  Create a graph G with a single node FINAL
2.  Sort CLL in descending order according to the end-X
    coordinates of candidate letters.
3.  for each candidate letter i in CLL in descending order
4.     Add node i to G
5.     Add an edge weighted w_i × s_i + P(l_i) × c from i to FINAL
6.     for each node j in G except FINAL node
7.        if segment of j and segment of i does not overlap
            && candidate letter j can be followed by i
9.           Add edge from i to j
10.          Set edge weight to w_i × s_i + P(l_j|l_i) × c
11.       end if
12.    end for
13. end for
14. Return G
```

Fig. 7 Pseudo code for graph construction. Note that the resulting graph is topologically sorted.

In Fig. 8, an example graph is depicted for a connected component that includes four letters. Because Ottoman script is written from right to left, the nodes are topologically sorted from right to left according to their distance from the rightmost pixel of the connected component. In this graph, there are two candidate letters for the beginning position, four candidate letters for the medial position, and two candidate letters for the end position. It should be noted that the graph is partial; the actual graph contains approximately 65 nodes and 1000 edges. Most of these candidate letters actually represent intermediate matches while sliding windows over an actual letter on the connected component.

For instance, consider the candidate letters for the leftmost position of the component in Fig. 8. A wider segment (incorrectly) led to the candidate letter

whereas a narrower one yields the (correct) candidate letter as

In Fig. 8, we illustrate two paths on the graph: the longest path (with the highest score) and another (shorter) path with a lower score, corresponding to the correct recognition and a faulty recognition, respectively. Clearly, the actual segmentation of the letters is also achieved along with the recognition.

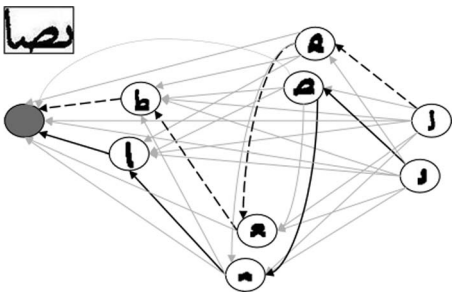

Fig. 8 Graph constructed for the sample connected component shown at the upper left corner. The longest path, corresponding to the correct recognition, is indicated with straight lines. The final node is shown in gray.

5 IR Framework for Ottoman Archives

The ultimate aim of CBR of Ottoman documents is to build an information retrieval (IR) environment for digital Ottoman archives that would provide effective query formulation and resolution within reasonable time constraints. As mentioned in Section 3, earlier works in the literature essentially focus on content-based retrieval,[4,16,17] which are flexible but far from providing the efficiency of a typical IR system, especially for large collections of documents. Furthermore, because CBR approaches essentially rely on image similarities, they tend to only allow QBE-based user interfaces and users cannot construct their queries through a typical keyword search. Thus, a compromise can be reached by constructing an IR system for the printed and/or clearly written Ottoman documents, for which a recognition system as described above can reach high accuracy rates. This would allow the fast access and sophisticated querying features of an IR system to be available for a large subset of the Ottoman archives. For harder documents (i.e., unreadable handwritten documents) that would probably yield lower recognition rates, content-based solutions can still be applied.

5.1 *Components of a Typical IR System*

In an IR system, finding the set of relevant documents for a user query depends on both the representation of documents and the similarity metric. The vector space model is one of the most widely used document representation methods in the literature. In this model, term frequencies are regarded as the content descriptors of documents. A document is represented as a vector of d dimensions, where d is the number of index terms. Each dimension corresponds to a separate index term. Frequencies of a document's index terms are used to assign a value to the corresponding dimension of the vector. This value may also be a Boolean value (0 or 1), solely indicating the existence of the index terms in documents. These vectors are called document vectors.[23]

In a full-text index, all the terms encountered in the documents are used for indexing and they constitute the vocabulary of the data set. "Stopping" is a widely used technique for ignoring terms that are ineffective at discriminating documents for query resolution, and thus, merely take up storage space. A "stopword list" is a collection of such terms. Because there has been no stop-word list constructed for Ottoman and because storage efficiency is not a concern in our experiments, we do not use a stop-word list in our framework. "Stemming" is another method that shrinks the vocabulary of the data set. Depending on design factors, such as the stemming algorithm and the language used,[24] it may also increase the effectiveness of the IR environment. For highly inflected languages, such as Arabic and Ottoman, developing effective stemmers is a hard task and not within the scope of this paper.

Table 1 Definitions of some statistical terms.

Term	Meaning
$f_{d,t}$	Frequency of term t in document d
$f_{q,t}$	Frequency of term t in the query q
f_t	Number of documents containing one or more occurrences of term t
N	Number of documents in the data set

An "inverted file" is the state-of-the-art indexing preference for large-scale search engines and IR systems.[25,26] An inverted file structure is composed of "posting lists" for each index term. A posting list typically contains ⟨document ID, frequency⟩ pairs and may also include the positions of terms in documents.

During the evaluation of a keyword-based query, only the posting lists of terms that appear in the query are retrieved. Partial similarity scores for each document are calculated by iterating through these posting lists, and they are summed up by a set of "accumulator" variables. When the evaluation is complete, the highest-scoring documents are retrieved using the accumulator set. Several optimizations of this basic processing scheme are proposed in the literature and effectively employed in the real-life systems.[25]

In the above process, the similarity between a query and documents can be calculated by one of several measures available in the literature. These measures basically assign weights for the terms in the query and in the documents to make use of some statistics derived from the collection (see Table 1).

Each query term's weight may be proportional to the inverse document frequency (IDF) of the term. The weights of terms in the documents may be proportional to the term frequency (TF). A general name for such weighting schemes is "TF×IDF" formulations. See Eq. (4) for a simple TF×IDF similarity formulation between a query q and a document d. Most similarity measures defined in the literature are variations of the TF×IDF formulation,

$$\text{Similarity}\ (q,d) = \frac{\sum_{t \in q} w_{q,t} \times w_{d,t}}{\sum_{t \in d} f_{d,t}},$$

$$\text{where}\ w_{d,t} = f_{d,t} \quad \text{and} \quad w_{q,t} = \frac{N}{f_t}. \tag{4}$$

The earlier works also identify a high need to evaluate

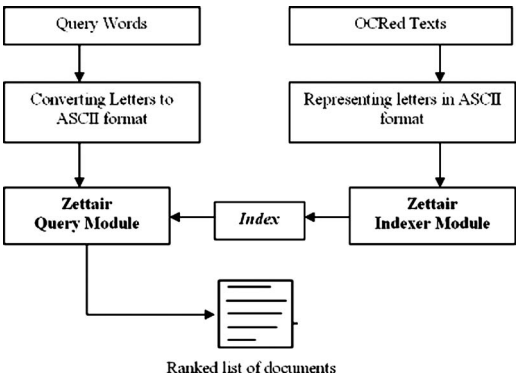

Fig. 9 Architecture of our IR framework for Ottoman archives.

Fig. 10 Grouping recognized letters into words and mapping these words into ASCII characters.

the IR performance of collections that have gone through OCR because of the fact that the recognition process may mislead IR systems. To remedy this problem, various methods have been proposed in the literature.[27–30] Most of these works try to correct OCR errors or expand the query terms. There may be no need for any further modification to the IR system if OCR is highly successful, because it is reported that IR performance is insensitive to minor (say 5%) recognition errors.[28]

5.2 *IR Framework for Ottoman Documents*

In this section, we describe our IR framework (Fig. 9) for Ottoman documents that have passed through the recognition stage. Because there are already several high-quality prototype systems for IR with sophisticated features, we have chosen to use one of these systems instead of constructing a new one from scratch. We chose the Zettair search engine, which is provided by The Royal Melbourne Institute of Technology (RMIT) and widely used in the literature.[31] Zettair creates an inverted index file for a given collection and then executes the queries on top of this index by employing one of the several available similarity metrics.

To be able to use Zettair with the Ottoman documents that have gone through OCR, we have to solve two problems. The first problem is that, IR systems (including Zettair) typically index and search words as a whole, whereas our recognition results are per connected component. Recall that, because an Ottoman word may involve several connected components called a subword, we thus first need a method for reconstructing the words to be indexed. We decide on word boundaries before the recognition process. The distance between subwords of a word is relatively shorter than the distance between individual words, just as in Latin script. Using this observation, a histogram of distances between consecutive connected components is calculated for each line. In this histogram, these distances concentrate on two different values and the average of these values can be used to determine the word boundaries. If these distances concentrate around a single value, then the line is composed of a single word. More specifically, in a typical scenario, most of these distances get values closer to zero and the rest of them get significantly greater values. This is because of the fact that a word in Ottoman script contains more than two subwords on average. By using this fact, we have seen that word-boundary detection process can be significantly accelerated for printed documents with an approximation. It is achieved by calculating the mean of all distances between connected components in a particular line and then multiplying it with a constant.

In Fig. 10(a), we show a line from an original Ottoman document and, in Fig. 10(b), we illustrate the word boundaries that the system has detected. Next, the document is passed through our recognition process, after which the boundaries are kept as before [see Fig. 10(c)].

The second problem is that the documents to be indexed contain the recognition results involving Ottoman letters, which are not supported by the Zettair system. To overcome this difficulty, we simply decided to map an Ottoman text that has gone through OCR [as in Fig. 10(c)] to ASCII characters. In Fig. 10(d), we illustrate the letter identification of the Ottoman characters as recognized by our system and, in Fig. 10(e), their corresponding ASCII representation is shown. For example, the leftmost Ottoman letter in Fig. 10(c) is mapped to *V*2 in Fig. 10(e) and the next one is mapped to *A*2. It should be noted that this basic approach is for experimental purposes only (i.e., to evaluate the IR performance on top of the documents that had gone through OCR. A real-life IR system could be extended to handle Ottoman characters in a standard way (e.g., by UTF encoding, for both recognition results and the queries).

6 Experimental Results

Experiments are divided into two sections. In the first section, the proposed segmentation and recognition method is evaluated. In the next section, the impact of OCR errors on the retrieval performance is analyzed. Experiments were performed on 100 pages of printed text, scanned from two different books teaching Ottoman script.[32,33] The documents were scanned at 300 dpi and saved as gray-scale images. Page skew is avoided manually in this step. The nearest color reduction method is used to binarize images. Processing time is ~4 s per document on a personal computer with a 2.0-GHz Intel processor.

6.1 *Recognition Evaluation*

The library used in this study includes the 48 letters shown in Fig. 1. For each letter in the library, ten training samples are taken manually from documents including five different fonts. Then, feature vectors of these samples are averaged

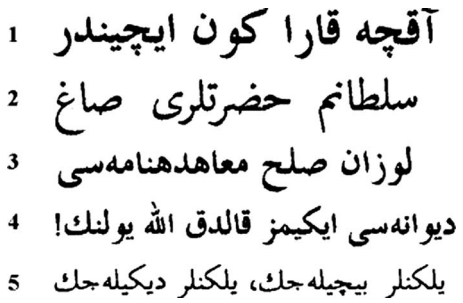

Fig. 11 Samples lines from our data set with different font sizes and letter thickness.

for each letter in the library and used for similarity calculations. During candidate letter determination, we discard the segments that are not similar to any letter in the library more than the predefined threshold of 0.85.

The data set includes documents with different font sizes and thicknesses (see Fig. 11). To obtain the ground truth data, letters in these documents were manually annotated (i.e., matched against the letters in the library) by using a tool developed in our research group for this purpose. Fifty randomly selected documents were used for learning unigram and bigram frequencies, and the remaining documents in the data set were used for testing. The entire data set contained 34,298 annotated connected components, each of which contained, on average, 2.04 letters. The experiments yielded the highest recognition rates when the constant c in Eq. (3) was set to 0.025.

In Table 2, we provide the recognition success of training and test sets in terms of precision and recall. Precision is the number of correct letter recognitions divided by the number of all recognitions as returned by our system. Recall is the number of correct letter recognitions divided by the total number of annotated letters in the data set. The results shown in the first two rows of Table 2 reveal that the proposed approach achieves very high recognition rates in both the training and test sets.

We also evaluated the effect of using the letter frequencies in Eq. (3) by setting the c constant to 0. As both precision and recall then dropped for the test set, this shows that using letter frequencies is an important factor for improving overall performance. Figure 12 shows the recognition results for a sample document.

Table 3 shows more detailed recognition statistics. Each connected component may include a number of letters.

Table 2 Recognition performance of the proposed method.

Collection	Precision	Recall	# annotated letters
Train set (50 pages) c=0.025	0.936	0.942	33,133
Test set (50 pages) c=0.025	0.931	0.939	36,908
Test set (50 pages) c=0	0.907	0.904	36,908

Fig. 12 Recognition results for a sample document. In each line, the lower part includes the original script and the upper part includes the recognized letters, as recorded in the library (i.e., without dots and diacritics).

More than half the components in our data set are composed of two or more letters. If all letters are correctly recognized in a component, it is called an exact match. If some additional letters are found for a component, it is called a superset match. If a subset of the letters in a component is recognized correctly, it is called a subset match. In the ideal case, the ratio of exact matches to the total number of components should be high. In our experiments, we observed that this ratio is 90%. This result is another indication that our method can be used as a building block for a retrieval system for Ottoman archives.

6.2 *IR Experiments*

We analyzed the possible effect of OCR errors on IR performance. Note that our goal was not to evaluate the IR effectiveness on Ottoman documents; this would require a larger collection, TREC-like query topics and their relevance judgments. Instead, for a given set of queries, we solely compared whether query results differ significantly when the query is executed for the original (ground-truth) document collection and its version that had gone through OCR.

Experiments were performed on both ground-truth documents and those that had gone through OCR, as follows. As is mentioned before, the ground-truth texts were obtained by manually annotating the original documents. These texts were then indexed by using the open-source Zettair search engine. The most frequent 50 words were identified from 6870 index terms and used as the query set. These queries were evaluated, and the top K (where K is typically $\leqslant 10$) most similar documents were obtained by using several different similarity measures implemented in Zettair. That is, during query evaluation, we experimented with four different similarity measures, namely Cosine,[34] Okapi BM25,[35] Hawking's Okapi variant,[36] and a language-modeling-based approach with Dirichlet smoothing.[37] For all these measures (and all other parameters), default settings of Zettair were used. Next, versions of the same documents that had gone through OCR were obtained by using our segmentation and recognition framework. These texts were indexed

Table 3 Recognition rates per connected component with varying number of letters.

No. letters per comp.	Subset match	Superset match	Exact match	Total No. connected components	Exact/ total
1	0	4	7,619	7,746	0.98
2	11	47	4,608	5,138	0.90
3	14	53	2,392	2,919	0.82
4	5	35	1,014	1,263	0.80
5+	4	79	573	921	0.62
Total	34	218	16,206	17,987	0.90

in the same way as described above, and 7542 index terms were found. The same set of queries was evaluated on these texts by using the same document similarity metrics. Ranked lists of documents for each similarity metric were later compared by calculating a symmetric difference score (see Fig. 13).

The symmetric difference score computes the similarity of the two results lists (generated from the ground truth and OCR documents) of a query by ignoring the order of the documents retrieved. If y is the size of the union of the top K retrieved documents in the lists and x is the number of the documents that appear in only one of the two lists, then the symmetric difference score is assigned to be $1-x/y$. If the two lists are similar, then the symmetric difference score is close to 1. If these two lists are totally disjointed, then the score becomes 0.[38]

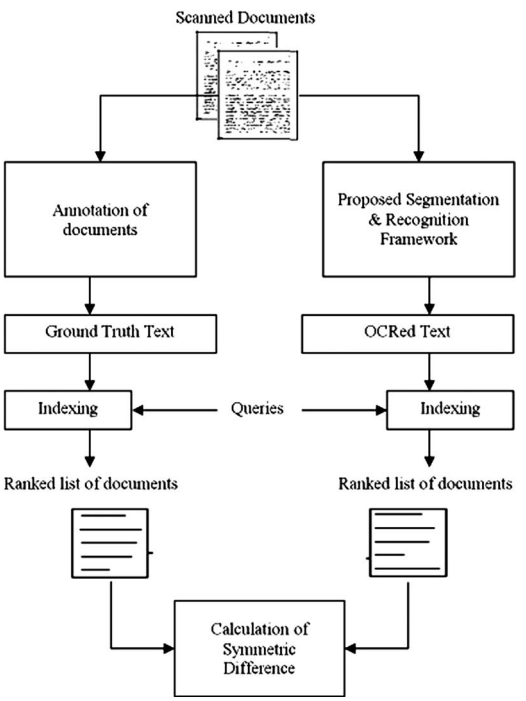

Fig. 13 Method of testing the effect of OCR errors on IR performance.

In Table 4, it is seen that top K documents retrieved using the ground-truth texts and those that had gone through OCR mostly overlap for varying values of K, as the symmetric difference score is over 0.8 in most of the cases. It can also be observed that the symmetric difference scores do not vary significantly for different similarity metrics and a particular K. This implies that the choice of the similarity measure does not significantly affect the retrieval results in this case. From these observations, we conclude that the 5% accuracy loss in OCR as reported in Section 6.1 has a limited impact on the retrieval performance. This also conforms to earlier findings in the literature (i.e., in Ref. 28, it is claimed most IR systems are almost unaffected by errors when OCR accuracy is $>95\%$.

7 Conclusion

The focus of this paper is to provide an efficient method for digitally recognizing connected letters in printed Ottoman script. In other words, segmentation and recognition of connected letters in connected scripts is studied. Advanced line detection, noise reduction, word boundary detection, etc., methods are not entailed in our discussions. Such schemes proposed in the literature can later be integrated with the current framework in order to build a complete system for printed documents insensitive to noise and document type.

In a nutshell, our framework integrates segmentation and recognition stages so that weaknesses of the classifier can be compensated by taking into account possible letter

Table 4 Symmetric difference for the top K retrieved documents with different similarity metrics.

K	Cosine measure	Dirichlet (μ=1500)	Okapi BM25	Hawkapi (α=0.5)
1	0.920	0.880	0.900	0.880
2	0.833	0.813	0.813	0.860
3	0.786	0.780	0.806	0.802
5	0.781	0.776	0.801	0.782
10	0.773	0.784	0.805	0.782

sequences for a particular connected component. Experiments show that high recognition rates are achievable by using the proposed approach. We also experimented on and verified that the negative effects of OCR errors on IR performance are tolerable when high recognition rates are obtained. Thus, it is possible to construct an IR framework for printed digital Ottoman archives on top of the proposed segmentation and recognition method.

For future work, it would be possible to improve the proposed method by integrating it with some of the OCR-correcting schemes in the literature in order to reduce misrecognition rates. Because alternate sets of candidate letters lie over different paths in the graph, it is also possible that the correct sequence of letters may be on the second- or even third-longest path. Using a dictionary and collecting frequencies of terms in the training set may further improve recognition rates. Another future research direction would be to examine the performance of the proposed approach on other printed or handwritten data sets with varying characteristics. Another area of interest would be to adapt our framework to other connected scripts, such as Arabic.

Acknowledgment

This work is supported by European Union 6th Framework Program under Grant No. FP6-507752 (MUSCLE Network of Excellence Project). We are grateful to Rana Nelson for proofreading and suggestions.

References

1. Google Inc., "Google Print Project," ⟨http://books.google.com/googlebooks/library.html⟩ (5 July 2009).
2. State Archives Office of Turkey, ⟨www.devletarsivleri.gov.tr⟩ (5 June 2008).
3. A. Amin, "Off-line Arabic character recognition: the state of the art," *Pattern Recogn.* **31**(5), 517–530 (1998).
4. E. Ataer and P. Duygulu, "Retrieval of Ottoman documents," in *Proc. of 8th ACM SIGMM Int. Workshop on Multimedia Information Retrieval*, ACM, New York, pp. 155–162 (2006).
5. K. D. Dhingra, S. Sanyal, and P. K. Sharma, "A robust OCR for degraded documents," in *Advances in Communication Systems and Electrical Engineering*, Lecture Notes in Electrical Engineering Vol. **4**, pp. 497–509, Springer, New York (2008).
6. M. S. Khorsheed, "Off-line Arabic character recognition—a review," *Pattern Anal. Appl.* **5**, 31–45 (2002).
7. R. Haraty and C. Ghaddar, "Arabic text recognition," *Int. Arab J. Info. Technol.* **1**, 156–163 (2004).
8. L. M. Lorigo and V. Govindaraju, "Offline Arabic Handwriting Recognition: a survey," *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(5), 712–724 (2006).
9. J. Sadri, S. Izadi, F. Solimanpour, C. Y. Suen, and T. D. Bui, "State-of-the-art in Farsi script recognition," in *Proc. 9th Int. Symp. on Signal Processing and its Applications*, IEEE, Piscataway, NJ (2007).
10. M. Soleymani and F. Razzazi, "An efficient front-end system for isolated Persian/Arabic character recognition of handwritten data-entry forms," *Int. J. Comput. Intell. Appl.* **1**, pp. 193–196 (2003).
11. L. Hamami and D. Berkani, "Recognition system for printed multi-font and multisize Arabic characters," *Arabian J. Sci. Eng.* **27**, 57–72 (2002).
12. S. A. Al-Qahtani and M. S. Khorsheed, "An Omni-font HTK-based Arabic recognition system," in *Proc. of 8th IASTED Int. Conf. Artif. Intell. Soft Comput.* (2004).
13. A. H. Hassin, X.-L. Tang, J.-F. Liu, and W. Zhao, "Printed Arabic character recognition using HMM," *J. Comput. Sci. Technol.* **19**(4), 538–543 (2004).
14. J. Chan, C. Ziftci, and D. Forsyth, "Searching off-line Arabic documents," in *Proc. of IEEE Conf. on Comput. Vision and Pattern Recognition*, pp. 1455–1462 (2006).
15. A. Öztürk, S. Güneş, and Y. Özbay, "Multifont Ottoman character recognition," in *Proc. of 7th IEEE Int. Conf. on Electronics Circuits and Syst.*, pp. 945–949 (2007).
16. E. Ataer and P. Duygulu, "Matching Ottoman words: an image retrieval approach to historical document indexing," in *Proc. of ACM Int. Conf. on Image and Video Retrieval*, ACM, New York, pp. 341–347 (2007).
17. E. Şaykol, A. K. Sinop, U. Güdükbay, Ö. Ulusoy, and A. E. Çetin, "Content-based retrieval of historical Ottoman documents stored as textual images," *IEEE Trans. Image Process.* **13**, 314–325 (2004).
18. I. Z. Yalniz, I. S. Altingovde, U. Güdükbay, and Ö. Ulusoy, "Ottoman archives explorer: a retrieval system for digital Ottoman archives," *ACM J. Comput. Cultural Heritage* (in press).
19. I. S. Altingovde, E. Şaykol, Ö. Ulusoy, U. Güdükbay, E. Çetin, and M. Göçmen, "Content-based retrieval (CBR) system for Ottoman archives," in *Proc. 14th Conf. on Signal Processing and Communications Application* (in Turkish), IEEE, Piscataway, NJ (2006).
20. I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes Compressing and Indexing Documents and Images*, 1st ed., Van Nostrand Reinhold, New York (1994).
21. L. Likforman-Sulem, A. Zahour, and B. Taconet, "Text line segmentation of historical documents: a survey," *Int. J. Document Anal. Recogn.* **9**(2–4), 123–138 (2007).
22. M. J. Swain and D. H. Ballard, "Color indexing," *Int. J. Comput. Vis.* **7**(1), 11–32 (1991).
23. G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM* **18**(11), 613–620 (1975).
24. M. Kantrowitz, B. Mohit, and V. Mittal, "Stemming and its effects on TFIDF ranking," in *Proc. of ACM SIGIR*, ACM, New York, pp. 357–359 (2000).
25. J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM Comput. Surv.* **38**(2), 1–56 (2006).
26. J. Zobel and A. Moffat, "Inverted files versus signature files for text indexing," *ACM Trans. Database. Sys.* **23**, 453–490 (1998).
27. P. B. Kantor and E. M. Voorhees, "The TREC-5 confusion track: comparing retrieval methods for scanned text," *Info. Retrieval.* **2**, 165–176 (2000).
28. D. Doermann, "The indexing and retrieval of document images: a survey," *Comput. Vis. Image Underst.* **70**(3), 287–298 (1998).
29. W. Magdy and K. Darwish, "Effect of OCR error correction on Arabic retrieval," *Info. Retrieval*, **11**(5), 405–425 (2008).
30. Y. Fataicha, M. Cheriet, J. Y. Nie, and C. Y. Suen, "Retrieving poorly degraded OCR documents," *Int. J. Document Anal. Recogn.* **8**(1), 15–26 (2006).
31. Zettair, Zettair Search Engine, version 0.9.3 (2006), ⟨http://www.seg.rmit.edu.au/zettair/⟩.
32. H. Develi, *Osmanlı Türkçesi Kılavuzu*, Vol. **1**, 7th ed., 3F Yayınevi, Istanbul (2006).
33. Y. Kurt, *Osmanlıca Dersleri*, Vol. **1**, 10th ed., Akçağ Yayınları, Ankara (2006).
34. A. Singhal, C. Buckley, and M. Mitra, "Pivoted document length normalization," in *Proc. of Int. ACM SIGIR Conf. on Res. and Dev. in Info. Retrieval*, ACM, New York, pp. 21–29 (1996).
35. K. Sparck Jones, S. Walker, and S. Robertson, "A probabilistic model of information retrieval: development and comparative experiments," *Inf. Process. Manage.* **36**, 809–840 (2000).
36. D. Hawking, T. Upstill, and N. Craswell, "Toward better weighting of anchors," in *Proc. of Int. ACM SIGIR Conf. on Res. and Dev. in Info. Retrieval*, ACM, New York, pp. 512–513 (2004).
37. C. Zhai and J. Lafferty, "A study of smoothing methods for language models applied to information retrieval," *ACM Trans. Inf. Syst. secur.* **22**(2), 179–214 (2004).
38. D. Carmel, D. Cohen, R. Fagin, E. Farchi, M. Herscovici, Y. Maarek, and A. Soffer, "Static index pruning for information retrieval systems," in *Proc. of 24th Annual Int. ACM SIGIR Conf. on Res. and Dev. in Information Retrieval*, ACM, pp. 43–50 (2001).

Ismet Zeki Yalniz is a PhD candidate at the Department of Computer Science, University of Massachusetts at Amherst, MA. He earned his MS in computer engineering from Bilkent University Ankara, Turkey, in 2008. His research interests include computer vision and content-based retrieval in historical document archives.

Ismail Sengor Altingovde is pursuing a postdoctoral study at the Department of Computer Engineering, Bilkent University, Ankara, Turkey. He received his PhD in computer engineering from Bilkent University in 2009. His research interests include information retrieval, XML, Web databases and query processing, multimedia databases, and content-based retrieval in historical document archives.

Uğur Güdükbay is an associate professor at the Department of Computer Engineering, Bilkent University, Ankara, Turkey. His research interests include multimedia databases, content-based retrieval in historical document archives, computer vision, and computer graphics. He received his PhD in computer engineering and information science from Bilkent University in 1994. He is a senior member of IEEE and ACM.

Özgür Ulusoy is a full professor at the Department of Computer Engineering, Bilkent University, Ankara, Turkey. His research interests include multimedia databases, content-based retrieval in historical document archives, Web databases, peer-to-peer systems, data management for mobile systems, and real-time and active database systems. He received his PhD in computer science from University of Illinois at Urbana-Champaign in 1992. He is a member of IEEE and ACM.