INTEGRATED SEGMENTATION AND RECOGNITION OF CONNECTED OTTOMAN SCRIPT

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BİLKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
İsmet Zeki Yalnız
August, 2008

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Özgür Ulusoy (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Uğur Güdükbay (Co-supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. İbrahim Körpeoğlu

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Ahmet Coşar

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Dr. Cengiz Çelik

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

INTEGRATED SEGMENTATION AND RECOGNITION OF CONNECTED OTTOMAN SCRIPT

İsmet Zeki Yalnız
M.S. in Computer Engineering
Supervisors: Prof. Dr. Özgür Ulusoy and
Assoc. Prof. Dr. Uğur Güdükbay
August, 2008

In this thesis, a novel context-sensitive segmentation and recognition method for connected letters in Ottoman script is proposed. This method first extracts a set of possible 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 very high 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.

Keywords: Optical character recognition (OCR), segmentation and recognition of connected scripts, connected scripts, information retrieval (IR).

ÖZET

BİTİŞİK OSMANLICA YAZI İÇİN TÜMLEŞİK BÖLÜTLEME VE TANIMA

İsmet Zeki Yalnız
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticileri: Prof. Dr. Özgür Ulusoy ve
Doç. Dr. Uğur Güdükbay
Ağustos, 2008

Bu tez çalışmasında, Osmanlıca bitişik metinlerdeki bağlı karakterler için içeriğe dayalı tümleşik bir bölütleme ve tanıma yöntemi önerilmiştir. Bu yöntem, öncelikle dökümanlardaki bağlı bileşenlerden sistematik olarak bir takım olası parçalar çıkartır ve bu parçalara en çok benzeyen aday karakterleri belirler. Sonrasında ise belirlenen çok sayıdaki aday karakterlerin arasından sözdizim olarak en olası olanlarını seçmek için bir skor fonksiyonu tanımlanmıştır. Bu skor fonksiyonunu maksimize eden karakterler, döngüsüz bir çizge oluşturulup bu çizge üzerindeki en uzun yolu hesaplayarak bulunmaktadır. Deney sonuçları göstermektedir ki, önerilen yöntem yüksek duyarlık ve tanıma oranları sağlamaktadır. Yapılan bilgi erişim deneyleri ise, önerilen yöntemin Osmanlıca matbaa metinler için tasarlanmış bir bilgi erişim sisteminin bir parçası olarak kullanılabileceğini göstermektedir.

Anahtar sözcükler: Optik karakter tanıma (OCR), bağlı karakterin bölütlenmesi ve tanınması, bitişik yazı, bilgi erişim (IR).

Acknowledgement

First of all, I would like to express my sincere gratitude to my supervisors Prof. Dr. Özgür Ulusoy and Assoc. Prof. Dr. Uğur Güdükbay for their instructive comments, suggestions, support and encouragement during this thesis work.

I am grateful to Asst. Prof. Dr. İbrahim Körpeoğlu, Assoc. Prof. Dr. Ahmet Coşar and Dr. Cengiz Çelik for reading and reviewing this thesis.

I would also like to thank to my colleagues, especially to İsmail Sengör Altıngövde, for their support and collaboration during all stages of this study.

Special thanks to my family for their support during my education and their never lasting efforts on me.

Finally, I would like to thank to TÜBİTAK for their financial support during my graduate education.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ottoman archives include a wealth of historical documents that provide valuable information on many aspects of an empire, which shaped the history of the "old world" for several centuries. Not surprisingly, providing means of electronic access even for a subset of this huge collection of handwritten and printed documents would be an exciting and priceless contribution for researchers from several disciplines and countries. Along with the recent advances in the hardware and reduction in the costs, digitizing collections of historical documents or even entire books becomes a more attainable goal. Accordingly, a growing body of Ottoman archives is also being digitized by the State Archives Office of Turkey (Devlet Arşivleri Ofisi), where the majority of this cultural heritage is kept. That is, digital images of historical Ottoman documents are obtained for the purposes of persistent storage and electronic access.

A robust and effective character recognition approach is the first stage of providing automatic access and sophisticated search and retrieval functionalities for such textual image archives. In this thesis, an integrated segmentation and recognition method for connected letters in digital Ottoman documents is described. Although Ottoman has several common features with Arabic, it has also significant differences, which disallows using the off-the-shelf OCR (Optical Character Recognition) software. Furthermore, even for Arabic, OCR is still a problem that is not completely solved and the recognition rates reported in the

literature are less successful in comparison to those for other languages based on, say, Latin characters [3]. Due to the difficulty of the problem, we essentially 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 can be found in different fonts and styles due to available technology of the age they are created. Thus, the recognition of letters from these printed documents is not a trivial problem and can serve as a first step 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 single letter or a connected group of letters. Next, a set of (possibly overlapping) segments are obtained by applying sliding-windows of varying sizes over these connected components. It should be noted that the sliding-window does not attempt to figure out the real letter boundaries, what it does is simply extracting some possible segments within predefined width ranges. The determination of the actual letter boundaries is postponed to and interleaved with the recognition stage. Then, each such segment is compared to a predefined letter library and determined as a candidate letter, for which it produces the highest similarity score.

Given a set of candidate letters, we define a score function to rank the possible sequences of candidate letters that do not overlap and can precede each other in the connected component, and choose the one with the highest score as the recognized letter(s). It should be noted that the highest scoring sequence may include candidate letters of varying sizes as the letters in the actual documents would not be of the same size [9]. In this way, the score function takes into account the size 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. The final recognition of letters is obtained 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 is very successful and provides precision and recall figures that are greater than 90 percent. Further experiments indicate that resulting OCR errors have limited effect on information retrieval (IR) performance; therefore the proposed framework can be used as a building block for an information retrieval environment for printed digital Ottoman archives.

The organization of the rest of the thesis is as follows. Various properties of Ottoman script are described in Chapter 2. In Chapter 3, related studies in the literature are briefly reviewed. Chapter 4 describes the proposed segmentation and recognition framework. In Chapter 5, an IR framework that is built upon the output of the segmentation and recognition process is discussed. Chapter 6 presents the experimental evaluation of the both recognition and retrieval stages. In Chapter 7 gives conclusions and future research directions. Appendix A describes the Web user interface of Ottoman Archives Explorer. Appendix B briefly explains the annotation tool used for annotating Ottoman texts. Appendix C mentions about the codebook viewer tool.

Chapter 2

The Ottoman Script

Ottoman and Arabic scripts have common characteristics in many ways. Ottoman script is also read from right to left and most letters in its alphabet are the same as Arabic alphabet. There are only 5 additional letters compared to Arabic alphabet [4]. After the removal of diacritics and dots, the Ottoman alphabet is shown in Figure 2.1. It should be noted that, a letter may have one of the four different forms according to its position in a word, namely being the beginning, medial or end letter in a word, or being written isolated. From the figure, it can be observed that there are repetitions. This is due to the fact that a particular form of a letter at a particular position may be the same as the form of another letter at another position. Some of the letter repetitions also caused by the removal of the diacritics and dots, which is a simplifying assumption for the purposes of this thesis. Another slight modification to alphabet is removing these letters that can be constructed by using the other letters. For instance, in Figure 2.1, the first letter at line 2 can be constructed from the second and fourth letters at the same line. After eliminating such letters, 48 distinct letters remain and they are referred as the *letter library* in this thesis.

In Ottoman script, each connected component should either involve a single -isolated- letter, or a group of letters, which strictly starts with one of the letters of the beginning form, continues with zero or more letters of the medial form end finally finishes with a letter of the end form. One or more connected components

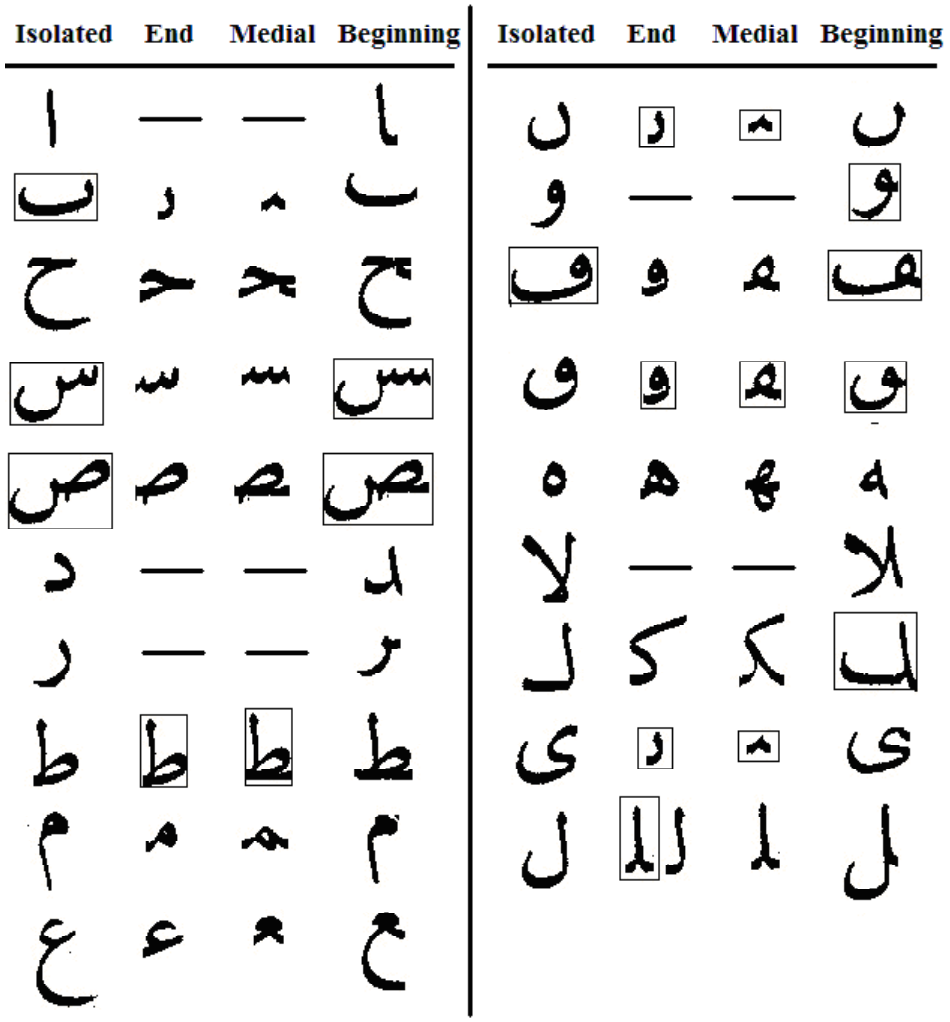


Figure 2.1: The Ottoman alphabet without diacritics and dots. Letters in the rectangles are either repeated letters or can be formulated by the other letters, and thus they are not included in the library.

Figure 2.2: An Ottoman word "Ensari" written using connected script.

form an Ottoman word. In Figure 2.2, there is a word which has 4 connected components. Three of them include single letters and one of them contains three letters.

Another property of Ottoman script is that, a letter may have different sounds in different words. For example, the last letter of the word in Figure 2.2 can give three different sounds in different contexts: 'y', 't' and 'i'. Sometimes there may be no way to understand the sound of a letter by just considering the word itself. There are words for which writing is exactly the same but the pronunciation and meaning are different. The reader should guess the intended word according to the context. In addition to this, Ottoman script lacks a set of important diacritics called 'harakat' unlike Arabic script. They are placed under and over some letters in order to identify the sound of a letter. Without these diacritics, Ottoman script becomes harder to read in comparison to Arabic script [17].

Chapter 3

Related Work

Text recognition is the automatic reading of the text written in digital images. The ultimate aim is to imitate human ability to read printed text with human accuracy with a higher speed [16]. Text recognition is widely studied in the literature and various methods are proposed for different scripts, fonts, noise level, etc. There are various problems involved in text recognition, such as segmentation of connected letters. Digital documents may contain connected letters due to noise or low resolution. There are also some scripts, such as Ottoman, Arabic and Hebrew, whose letters are meant to be connected in certain ways according to some syntactic rules. In such cases, recognition of individual letters in documents becomes quite hard. Advanced techniques should be developed in order to handle such problems.

In Section 3.1, character segmentation methods in the literature are classified and briefly explained. Our proposed method is placed into the context. Later, in Section 3.2, related work for recognizing text in Ottoman documents is elaborated.

3.1 Segmentation Methods

There are two major unknowns for the recognition of connected letters. These are the number of letters and their letter boundaries in a word or connected component. There may also be unexpected dissections or connections between letters due to noise in the document image. In such cases, recognition of individual letters becomes even harder. The aim of segmentation methods is to find better letter boundaries so that these letters can be classified in a straight-forward manner. Once the letters are segmented correctly, it is possible to recognize these isolated letters with high accuracy. Apparently, letter segmentation is a critical stage for the OCR process.

There are various segmentation methods proposed in the literature [16, 8]. Systems that employ any letter segmentation method are called "Segmentation-based systems". These systems can be classified as given below [16]:

1. *Isolated/pre-segmented characters:* The focus of such systems is not the segmentation of letters. It is assumed that letters are obtained from a reliable segmentation algorithm. The aim is to recognize isolated or segmented letters with high accuracy. These systems are not practical. They can not be deployed and used directly, especially for connected scripts. As an example of such systems, see [19].

2. *Segmenting a word into characters:* Letters in a connected component can be divided into letters by using an explicit segmentation algorithm prior to recognition. In this way, segmented letters can be classified directly by a classifier, just like in Latin OCR. However, overall success heavily depends on the accuracy of the segmentation algorithm used. It is not easy to build a robust segmentation algoritm, which tries to segment a connected component by using only features like vertical and horizontal projection vectors, connected component contours, etc. In Figure 3.1, a connected component with connected letters is segmented by tracing vertical projection vector. In this method, a threshold is provided for detecting connectivity points. It can be seen that connectivity points exhibit the least sum of the average

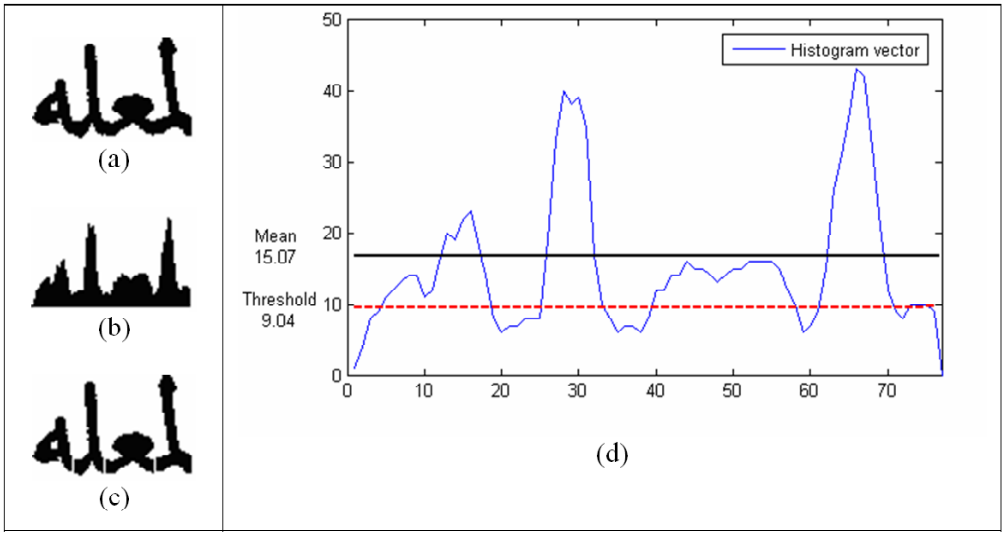


Figure 3.1: (a) The original isolated component, (b) its vertical histogram, (c) the segmented characters, and (d) a larger view of (b). The threshold is computed as the 0.6 of the histogram mean. Each segment is required to be larger than a predefined minimum width [25].

summation over all columns [2, 3, 25]. It should also be noted that, explicit segmentation methods do not use outputs of any letter classifier for establishment of letter boundaries.

3. *Segmenting a word into primitives:* There are methods that tries to segment a connected component into symbols, which may represent letters, fractions of letters or ligatures. One way to obtain these symbols can be oversegmentation of the connected component by tracing the contour of the connected components. It can be done by finding points on the contour where there is a transition from a column, which has all its black pixels within the baseline boundaries, to another column, which does not [1]. See Figure 3.2 as an example. Associated symbols that form particular letters can later be learned and used for recognition in further stages.

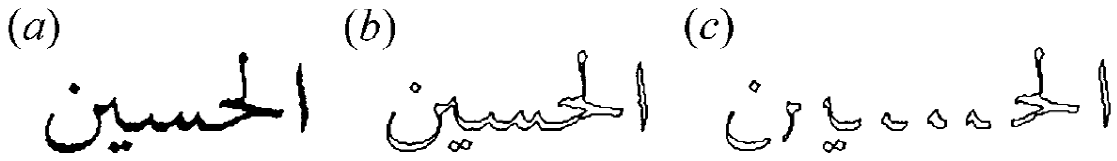


Figure 3.2: The segmentation of the contour of a word. (a) Original, (b) the contour, (c) segmented contour [16].

4. *Integration of segmentation and recognition:* The segmentation and recognition stages can be interleaved so that better letter boundaries and consequently recognition rates can be obtained. In such systems, the connected component is systemmatically divided into many overlapping segments without regarding their contents. Each such segment is later classified and used for determination of a coherent segmentation and recognition. The main principle of such methods is to bypass the segmentation problem. Such methods can also be called segmentation-free, since they do not employ any feature-based segmentation method. In such systems, recognition errors are mostly due to failures in classification [8].

   The segmentation and recognition framework proposed in this thesis is also in this catagory. The segmentation problem is bypassed by integrating segmentation and recognition stages. By sliding windows over each connected component, a set of segments are gathered and classified. Then, a set of non-overlapping segments are chosen according to a score function so that the recognition accuracy can be maximized. Details are explained in Chapter 4.

There are also text recognition systems which do not employ any segmentation method. These are called "segmentation-free systems". In principle, these systems tries to recognize words as a whole instead of individual characters or primitives. It is achieved by extracting features from each word and matching these features to the database of features extracted from the words in the vocabulary. The most likely word is selected as the match. One drawback for such

systems is the limited vocabulary. Rare words may not be included in the vocabulary. Moreover, broader vocabulary may degrade the system performance significanty. Examples for segmentation-free systems include [4, 5].

3.2 Recognition of Ottoman Script

In the literature, there are relatively few works that directly attack the character recognition problem for Ottoman documents. This may be mostly due to the fact that it is not a currently spoken language. Ottoman is essentially 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, which reveals some common features with Ottoman, is a language still spoken by millions of people all over the world. Therefore, 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.

Comprehensive surveys [3, 16] on off-line Arabic character recognition discuss a variety of approaches adapted for segmentation and recognition, involving methods based on local and global features, neural networks, graph-based algorithms, stochastic methods like Hidden Markov Models (HMMs), etc. Amin [3] concludes that Arabic OCR is still an open research area and most of the research results are inconclusive as they are provided on small datasets that are not available to others. More recently, encouraging results for automatic retrieval on textual images of Arabic documents are reported [9]. The method described there has some similarities to our approach. In both works, the segmentation is achieved in an integrated manner with the recognition. However, the actual recognition in that work employs the HMM, whereas in this thesis a graph-based model is constructed.

In one of the earliest works for Ottoman character recognition, first a chain code transformation is applied to the main strokes of the characters and then the

recognition of these transformed characters is achieved by using the HMM [6]. The success of this approach is said to be dependent on the performance of the thinning process. In [19], a neural network (NN) based recognition approach is applied for Ottoman characters. Although the recognition rate is said to be high, both of the training and testing stages seem to be applied on manually segmented characters. The segmentation problem of the connected letters is not attacked. In [4], a retrieval system for Ottoman documents is proposed, which 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 [5], 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 SIFT descriptors extracted from salient points. Words are matched by comparing the similarity of these visual terms. In [4] and [5], queries can only be constructed by finding examples of the words over sample documents. There can also be rare words that a user may want to search for. It may be a time consuming task for the user.

Saykol et al. propose an effective method for the compression of Ottoman documents and content-based retrieval (CBR) on them [21]. 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 extracted component is compared to the current elements of the library to check if it is (or, it contains) an already discovered symbol. If so, the location of the occurrence is recorded to the document's codebook and the symbol is removed from the document. If it is a new symbol, 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 tends to increase as more documents are processed. Consequently, processing new documents becomes inefficient, when there are thousands of symbols to be compared in the symbol library. Users can construct queries only by query by example

	GREEDY_APPROACH (A connected component CC, a letter library LL)
1.	Found segment FS = null
2.	**Repeat**
3.	Obtain a segment S of CC
4.	Compare S with each letter L in LL
5.	**if** for some L, Similarity (S, L) > threshold T
6.	T = Similarity (S, L)
7.	$FS = S$
8.	**end if**
9.	**Until** (All possible windows are slid horizontally on CC)
10.	**if** (FS is not null)
11.	Remove FS from CC
12.	Record FS and its matching letter
13.	Call GREEDY_APPROACH for remaining parts of CC
14.	**Endif**

Figure 3.3: The pseudocode for the greedy segmentation and recognition algorithm.

(QBE) as in [4] and [5]. 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 described in [21] 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 [2, 25]. In Figure 3.2, we provide the pseudo code for the greedy segmentation and recognition approach for connected letters [25]. In a nut shell, this algorithm slides several windows with varying sizes over each connected component, and for each such window it computes the similarity of the letter in the library and the segment extracted by the window. The highest scoring segment is then decided as a correct recognition and removed from the connected component. The same approach is then recursively applied for the remaining parts of the connected component, until it is totally consumed. Queries can be given by either using a virtual keyboard designed for Ottoman script or selecting a query region over processed documents (QBE).

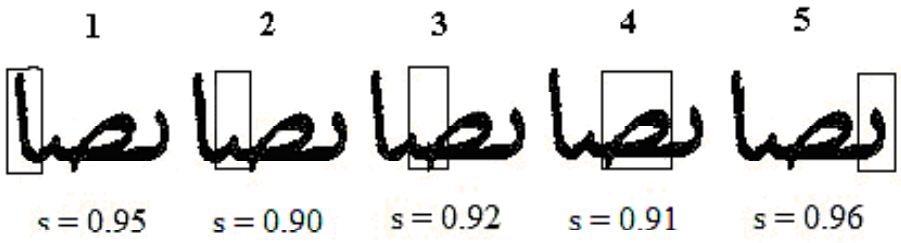


Figure 3.4: Illustration of the greedy approach: windows are segmented in the order of descending similarity score, i.e., 5, 1 and 3. Since the third window is an incorrect segment that overlaps with windows 2 and 4; once it is removed, it is not anymore possible to recognize the other two medial type letters in windows 2 and 4.

The greedy segmentation of letters may not always provide the optimum results. This is essentially caused by the greedy nature of the algorithm: an incorrect choice especially at the earlier stages of the recognition may mislead the following stages and significantly reduce the overall recognition rate. For instance, in Figure 3.4, we illustrate how the greedy algorithm may fail while segmenting and recognizing the letters from a connected component that actually includes 4 letters. Furthermore, the greedy method can not use the statistical information like letter occurrence frequencies and n-gram probabilities. To overcome these problems, we propose a new method that has almost the same complexity with the greedy approach but postpones the recognition decision at the end, in order to find optimum letter boundaries and exploit the occurrence probabilities among letters.

Chapter 4

The Segmentation and Recognition Framework

In Figure 4.1, we illustrate the stages of the proposed segmentation and recognition framework. In this study, we start with extracting connected components from the Ottoman documents and obtain a set of possible segments from a component by applying sliding-windows of varying sizes. As we mention before, the purpose of obtaining these segments is not actually determining 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 such segment is compared to the letters in the library and matched to a candidate letter that yields the highest similarity score. It should be noted that, there might be several different (overlapping) segmentations and candidate letter recognitions for a particular connected component due to the varying sizes of sliding-windows. Finally, by using a graph-based model, we efficiently compute a particular sequence of the candidate letters that maximizes a scoring function over all possible candidate letter sequences. Each stage is described in detail in the following sections.

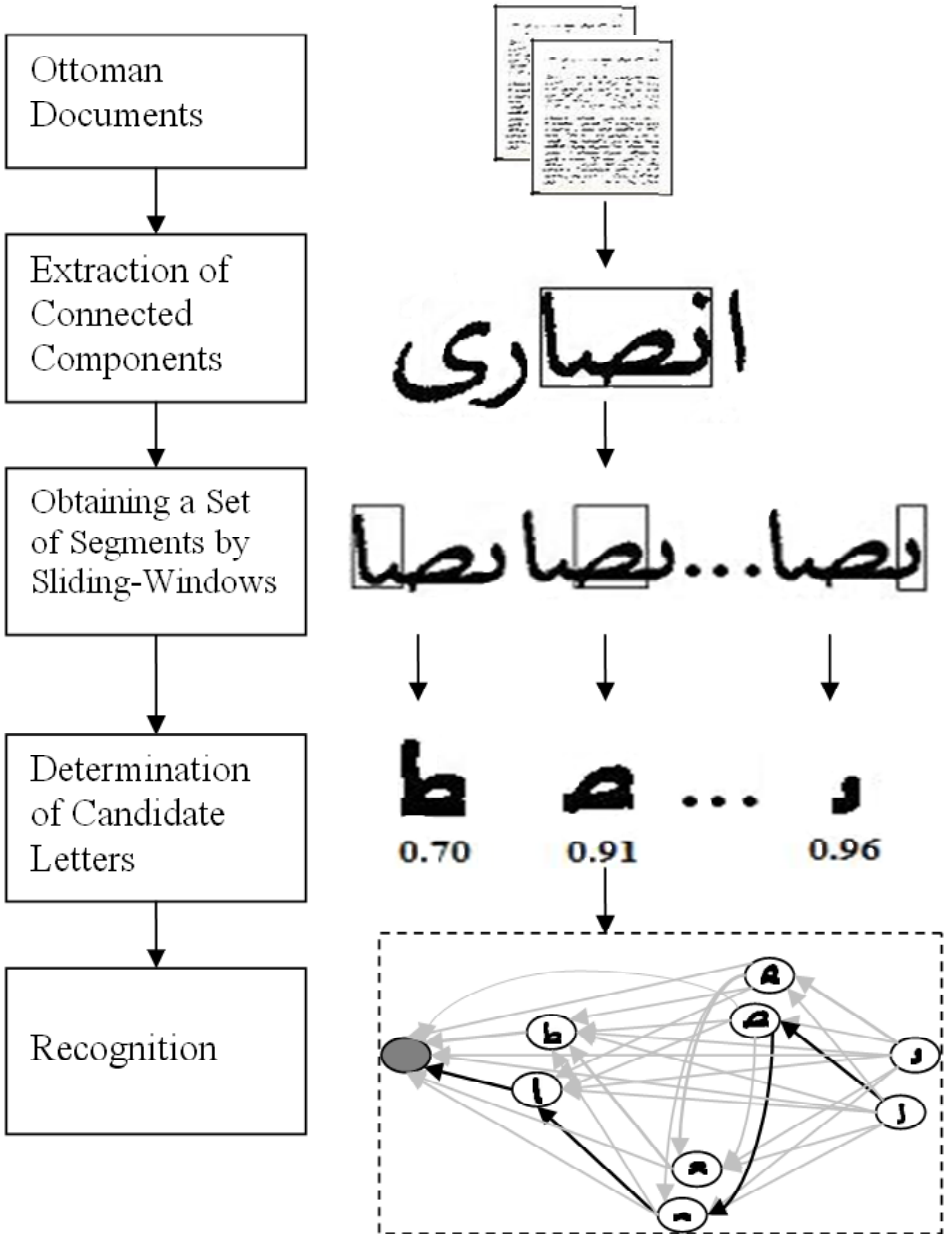


Figure 4.1: The stages of the proposed segmentation and recognition approach.

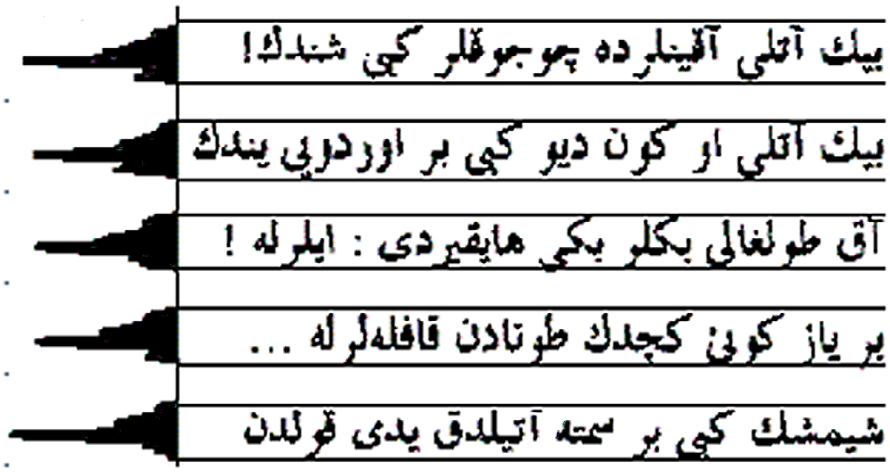


Figure 4.2: Determining the line height information.

4.1 Extracting and Segmenting Connected Components

A connected component is defined as a connected group of black pixels in the document image. Connected components are extracted using the approach outlined in [24]; that is, the document is scanned from left to right and top to bottom, and whenever a black pixel is encountered a 4-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 the nested components, diacritics and dots are detected separately. For the purposes of this study, we restrict ourselves to recognize letters without diacritics and dots (as shown in the alphabet in Figure 2.1), thus these components are discarded by using the predefined thresholds for component width, height and area. This operation also removes some of the noise in the document images.

The line height information is also extracted from documents while connected components are identified. This is achieved by using the horizontal projection vector of the document (see Figure 4.2). Upper and lower bounds for each line are found by tracing horizontal projection vector and detecting peaks. In this

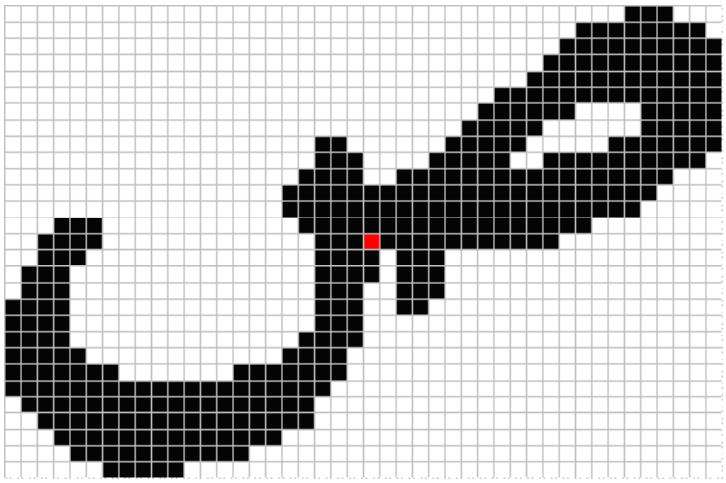


Figure 4.3: An Ottoman letter.

way, all connected components learn the height of the line that they belong to. The ratio of the letter height to the height of the line is later used during the determination of candidate letters.

Once the connected components are extracted, the next step is obtaining a set of segments. The sliding-window segmentation approach does not make an initial "guess" about the letter boundaries, but simply slides a window of varying sizes over the bitmap of the connected component. In particular, the window is applied for all sizes between the component width and a predefined minimum width, so that almost all possible sizes can be tried. This is necessary in order to find the letter boundaries as accurate as possible in the following stages. The content of each such window is stored as a segment, along with a number of features as discussed in the next subsection. It should be noted that, while some segments may correspond to a single letter, which is the preferred case, some others may include parts of one or more letters. The latter type of erroneous segments will be eliminated in the candidate letter determination stage, as discussed in Section 4.3.

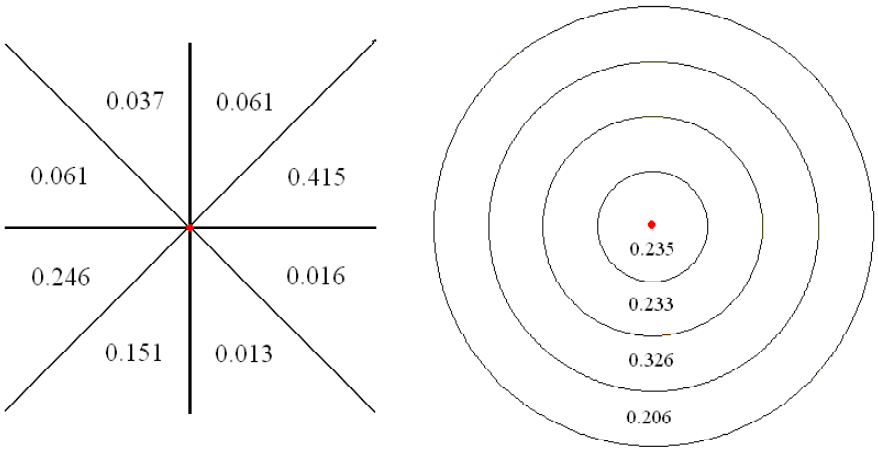


Figure 4.4: Angular and distance span of the letter in Figure 4.3.

4.2 Extraction of Segment Features

A number of features for each segment are computed, as described below.

- *Segment aspect ratio* is basically the ratio of width and height of the extracted segment.
- *Segment height ratio* is the ratio of a segment's height to the height of the line that it belongs to.
- *Angular span vector* is the number of black pixels in θ-degree slices centered at the center of mass with respect to the horizontal axis.
- *Distance span vector* is the number of black pixels in between the concentric circles centered at the center of mass with radius r, 2r, 3r, etc.

The angular and distance span vectors are the spatial domain features as described in [21] for the Ottoman character recognition task. The entries of these vectors are normalized by the area, which is the total number black pixels, of the segment. In this work, angular and distance span vectors of the sizes 12 and 8 are used, respectively. Figure 4.4 illustrates these vectors for an example letter from

the alphabet. It should be noted that, both of these features are scale invariant. Rotation invariance can be achieved by simply shifting entries of angular span vector by one slice to the left and right.

4.3 Determining Candidate Letters

At this stage, the segments are compared to a predefined library of Ottoman letters to obtain a similarity score. For each letter in the library, the features discussed in the previous section are also computed and stored. During candidate letter determination, the similarity of each segment to the letters in the library is computed and the top score for each segment (and the corresponding letter) is obtained. If the highest similarity score for a segment is less than a threshold value, this segment is discarded (e.g., it may include a part of a letter or more than one letter, etc.). Otherwise, the best-matching letter for that particular segment is called a "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 information 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 with only those letters that can appear at that particular position.

In this work, the histogram intersection technique [23] is adapted for measuring similarity between two feature vectors H_1 and H_2, that belongs to an extracted segment and a letter in the library, respectively. In Equation 4.1, $|H|$ denotes the sum of all entries of vector H. Here, the range for the similarity measure is [0, 1]. If two letters are similar, their feature vectors are also similar and their histogram intersection value is close to 1. Conversely, if they are dissimilar, similarity value is relatively lower.

$$Similarity(H_1, H_2) = \frac{\sum_{i=1}^{n} \min(H_1[i], H_2[i])}{\min(|H_1|, |H_2|)} \tag{4.1}$$

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 similarity calculation. These are the aspect ratio of the segment and the ratio of the segment's height to the height of the line that it belongs to. If at least one of these ratios for S is different than 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 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 efficiently find the sequence that maximizes the scoring function.

Recall that, the segments and corresponding candidate letters obtained from a particular connected component can have varying sizes, overlapping boundaries and various similarity scores. Thus, while determining a sequence of these candidate letters as the final recognition, we want to satisfy the following goals, to the greatest extent possible:

1. the candidate letters in the sequence should cover the connected component,
2. the boundaries of candidate letters should not overlap,
3. the candidate letters should have high similarity scores, and
4. the probability of letters for being consecutive should be as high possible.

Remarkably, it may not be possible to maximize all of these goals at the same time. For instance, two candidate letters with the highest similarity scores may have overlapping boundaries. Or, it may be impossible for these letters to appear consecutively in a connected component according to the Ottoman language rules (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 problem expressed in the form of the score function shown in Equation 4.2.

$$Score = \sum_{i=1}^{n} (w_i \times s_i + P(l_i|l_{i-1}, l_{i-2}, ..., l_o) \times c) + (w_0 \times s_0 + P(l_0) \times c) \quad (4.2)$$

$$\text{such that } \textit{End-X-coordinate}(i\text{-}1) < \textit{Start-X-coordinate}(i)$$

In this equation, w_i denotes the ratio of the candidate letter i's width to the width of the connected component from which this letter is extracted, and s_i denotes the similarity score of this candidate letter. $P(l_i|l_{i-1}, l_{i-2}, ..., l_o)$ denotes the probability of encountering the letter l_i after seeing a sequence of letters l_o to l_{i-1}. Finally, c is a constant ($0 < c < 1$), which assigns weight for the letter statistics. It should be noted that, these three elements of the equation maps to the goals 1, 3 and 4 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 2, and the new candidate letters can be added to the current sequence as long as this constraint is not violated. Typically, the number of letters in a sequence (n) would be less than the total number of candidate letters extracted from the connected component.

It should be noted that Equation 4.3 is an approximation of Equation 4.2 and it uses only unigram and bigram frequencies of the letters. In this work, we use Equation 4.3 since unigram and bigram frequencies can be easily obtained from the datasets and it gives way to an efficient solution. That is, the sequence of candidate letters that maximizes Equation 4.3 can be found in polynomial time by using a graph-based model.

$$Score = \sum_{i=1}^{n} (w_i \times s_i + P(l_i|l_{i-1}) \times c) + (w_0 \times s_0 + P(l_0) \times c) \quad (4.3)$$

	CONSTRUCT_GRAPH (A candidate letter list CLL)
1.	Create a graph G with a single node FINAL
2.	**Sort** CLL in descending order according to end-X coordinate 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 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
8.	Add edge from i to j
9.	Set edge weight
10.	**end if**
11.	**end for**
12.	**end for**
13.	**Return** G

Figure 4.5: The pseudo code for the graph construction. It should be noted that, the resulting graph is topologically sorted.

Graph construction is achieved as follows: Each candidate letter is added as a node to construct a directed acyclic graph. An edge from the node i to node j is created if these candidate letters do not overlap and the position of j precedes i (as Ottoman is written from right to left) in the component from which they are extracted. Ottoman syntax constraints for the letter forms appearing in a connected component are also considered. For instance, no edges can exist between any two candidate letters that are both in the beginning or 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 for an outgoing edge is given as $w_i \times s_i + P(l_i|l_{i-1}) \times c$, 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 "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. Edges that arrive to the final node is given weight $w_i \times s_i + P(l_i)$, since there is no bigram frequency for the letter sequence arriving to this special node. In this way, partially recognized connected components are not ignored. In

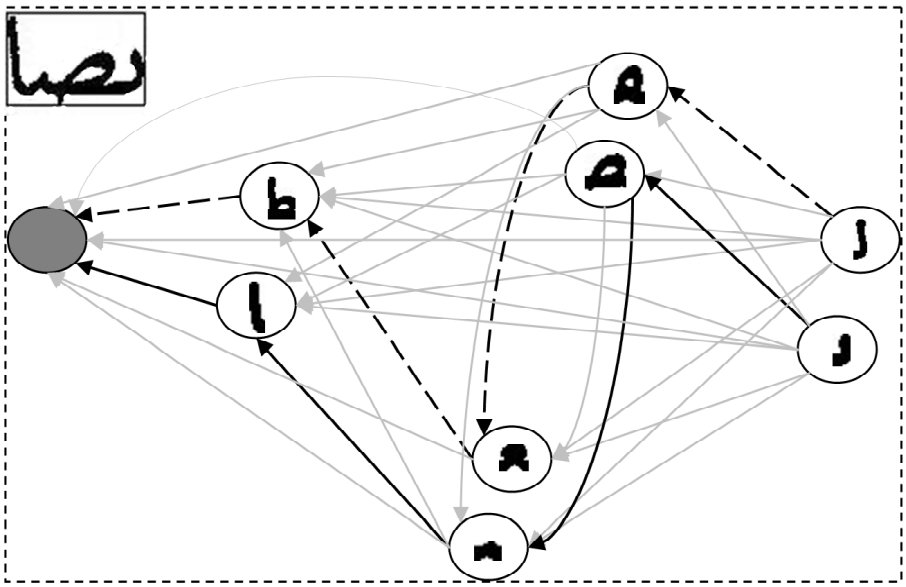


Figure 4.6: The graph constructed for a sample connected component shown at the upper left corner. The longest path corresponding to the recognition is indicated with straight lines. Final node is shown in gray.

Figure 4.5, we provide the pseudo code for the 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 Equation 4.3, includes the sequence of letters that are returned as the recognized letters for a particular component.

In Figure 4.6, an example graph is depicted for a connected component including 4 letters. Recall that Ottoman script is written from right to left. Subsequently, the nodes are topologically sorted from right to left according to their distance to 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. Indeed, the actual graph for this example involves approximately 65 nodes and more than 1000 edges, which are not shown for simplicity. That is, the segments with different sizes may lead several candidate letters for a particular connected component.

For instance, consider the candidate letters for the leftmost position of the component in Figure 4.6. A wider segment (incorrectly) led to the 'b' like candidate letter whereas a narrower one yields the (correct) 'l' like candidate letter. In the figure, we illustrate two paths on the graph: the longest path and another 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.

Chapter 5

An Information Retrieval Framework for Ottoman Archives

The ultimate aim of the character-based recognition of Ottoman documents is to build an information retrieval (IR) environment for digital Ottoman archives, which provides effective query formulation and resolution within reasonable time constraints. As it is mentioned in the related work chapter, earlier works in the literature essentially focus on content-based retrieval [5, 4, 21], which is flexible yet far from providing the efficiency advantages of a typical IR system, especially for large collections of documents. Furthermore, since CBR approaches essentially rely on image similarities, they tend to allow QBE based user interfaces, and users can not construct their queries as in a typical keyword-search scenario. Thus, a compromise may be constructing an IR system for the printed and/or well-written Ottoman documents, for which a recognition system as described above can reach high accuracy rates. This would allow both fast access and sophisticated querying features of an IR system for a large subset of the Ottoman archives. For harder documents (i.e., unreadable handwritten documents, etc.) that would probably yield lower recognition rates, content-based solutions can still be applied.

In what follows, essentials of an IR system and its performance on OCRed collections are briefly reviewed as published in the literature. Then, our IR framework is presented, which is based upon the document contents that are OCRed as described in the previous section. In the next Chapter, the effects of the proposed recognition framework on the information retrieval performance is evaluated.

5.1 Typical Components of an IR System

In an IR system, finding the set of relevant documents to a user query depends on both the representation of documents and the similarity metric defined accordingly between a document and a query. *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. For a document, frequencies of its index terms are used to assign a value to the respective 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 as document vectors [20].

In a full-text index, all of the terms encountered in the documents are used for indexing and constitute the vocabulary of the dataset. Stopping is a widely used technique for ignoring terms that are ineffective in discriminating documents for query resolution. A *stopword list* is a collection of such terms. Since the storage efficiency is not a concern for our experiments and there is no available stopword list constructed for Ottoman language, a stopword list is not used in our framework. Stemming is another method that not only shrinks the vocabulary of the dataset, but may also increase the effectiveness of an IR environment depending on design factors such as the stemming algorithm and the language used [15]. For highly inflected languages, such as Arabic and Ottoman, developing effective stemmers is a hard task and not within the scope of this thesis.

Table 5.1: Definitions of some statistical terms.

Term	**Meaning**
fd,t	The frequency of term t in document d
fq,t	The frequency of term t in the query
ft	The number of documents containing one or more occurrences of term t
Ft	The number of occurrences of term t in the dataset
N	The number of documents in the dataset
n	The number of indexed terms in the dataset

An inverted file is the state-of-the-art indexing preference for large-scale search engines and IR systems [29, 28]. Inverted file structure is composed of posting lists for each index term and an array of document weights. A posting list typically contains <*document ID, frequency*> pairs and may also include term positions in each document. A pair indicates existence of the term in a document by keeping the ID of the document along with the term frequency.

During the evaluation of a keyword-based query, only the posting lists of the terms that appear in the query are retrieved. Partial similarity scores for each document are calculated by iterating over these posting lists and they are summed up over an array of "accumulator" variables. At the end, highest scoring documents are retrieved by using the accumulator array. Several optimizations over this basic processing scheme are proposed in the literature and effectively employed in the real-life systems [28].

In the above process, the similarity between a query and documents can be calculated by one of the several measures in the literature. These measures basically assign weights for the terms in the query and documents, making use of some statistics derived from the collection (see Table 5.1).

$$Similarity\,(q,d) = \frac{\sum\limits_{t \in q} w_{q,t} \times w_{d,t}}{\sum\limits_{t \in d} f_{d,t}},\ \ where\ \ w_{d,t} = f_{d,t}\ \ and\ \ w_{q,t} = \frac{N}{f_t} \qquad (5.1)$$

Each query term's weight may be proportional to the inverse document frequency (IDF) of the term. The weights of terms in documents may be proportional to the term frequency (TF). A general name for such weighting schemes is TF x IDF formulations. See Equation 5.1 for a simple TF x IDF similarity formulation

between a query q and a document d. Most similarity measures defined in the literature are the variations of the TF x IDF formulation.

The earlier works also identify that there is an important need to evaluate the IR performance on OCRed collections as the recognition process may mislead IR systems, which essentially use the information that whether a term appears or not in a document, in one way or the other. To remedy this problem, several methods are proposed. These methods generally try to correct OCR errors or expand the query [11, 14, 18]. In these works, it is shown that negative effects of OCR errors can be reduced by such advanced methods. On the other hand, for the cases where OCR is highly successful, the IR performance is observed to be insensitive to the recognition errors and there may be no need for any further modification in the IR system [11].

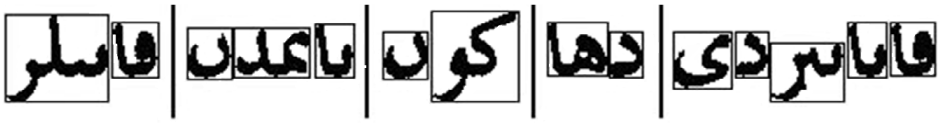
5.2 Information Retrieval Framework for Ottoman Documents

In this section, an IR framework is described for the Ottoman documents that have passed through the recognition stage as described before. Note that, since there are several high-quality prototype systems for IR with sophisticated features, we prefer to use one of these systems instead of constructing a new one from the scratch. In particular, we use the Zettair search engine, which is provided by RMIT and widely used in the literature [26]. 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 OCRed Ottoman documents, we have to solve two problems. First of all, IR systems typically index and search words as a whole, whereas our recognition results are per connected component. Recall that, an Ottoman word may involve several connected components separated by white space. Thus, we need a method for re-constructing terms to be indexed, at the

a)

b)



c)

d)

47-27-26-20-37-18	3-38-18-37-20	3-2-19	37-25-5	4-5-47-26-20-37-20-37-18

e)

V2A2Z1U1K2S1	C1L1S1K2U1	C1B1T1	K2Y1E1	D1E1V2Z1U1K2U1K2S1

Figure 5.1: Grouping recognized letters into words and mapping these words into ASCII characters.

first place. In this thesis, word boundaries are decided by using the original document collection, i.e., before the recognition process. In particular, it is achieved by applying a threshold for the distance between two connected components. It is obvious that there can not be a global threshold for word segmentation through the entire collection. Therefore, for each line, a threshold is calculated by considering the distances between its connected components. In Figure 5.1 (a) we illustrate a line from an original Ottoman document and in Figure 5.1 (b) we show how the word boundaries are detected. Next, the document is passed through our recognition process, after which the boundaries are still kept as obtained before (See Figure 5.1 (c)).

Clearly, 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 convert an OCRed Ottoman text (as in Figure 5.1 (c)) to a version including ASCII characters. More specifically, each

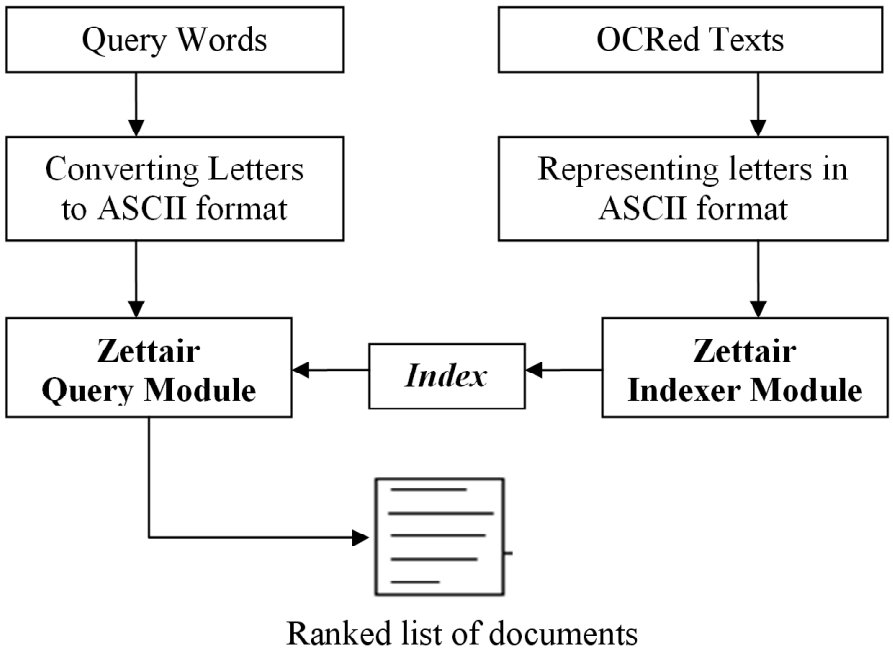


Figure 5.2: The architecture of IR framework for Ottoman Archives.

Ottoman letter is converted to a set of ASCII characters; for instance the leftmost Ottoman letter in Figure 5.1 is mapped to *V2* and the second one is mapped to *A2*. In Figure 5.1 (d), the letter-id of the Ottoman characters are illustrated as recognized by our system, and in Figure 5.1e their corresponding ASCII representation is shown. Note that, this basic approach is for experimental purposes only, i.e., to evaluate the IR performance on top of the OCRed documents. A real life IR system should better be extended to handle Ottoman characters, which could be expressed in some standard way, e.g., by UTF encoding, in both recognition results and in the queries. For the simplistic framework defined here, the queries are also processed in Ottoman in the same manner (i.e., by converting to ASCII characters before evaluation).

In Figure 5.2, the architecture of our IR framework for OCRed Ottoman documents are illustrated. Note that, in this framework, the query can be specified either by selecting a region from an example document (QBE) or by typing the actual word(s) using the letters in our library. For the former case, we decide

on the word boundaries in the query, recognize the letters and then map them to ASCII exactly in the same manner as for the documents. For the latter case, it is presumed that a virtual keyboard including the Ottoman letters in the library shown in Figure 2.1 are presented to the user, so that the word boundary detection and character recognition stages are redundant and the query word(s) are immediately sent to the query processing module once they are mapped to ASCII characters.

Chapter 6

Experimental Results

Experiments are divided into two sections. In the first section, the proposed segmentation and recognition method is evaluated. Next, the impact of OCR errors on the retrieval performance is analyzed.

Experiments are performed on 100 pages of printed text, scanned from two different books teaching Ottoman script [10, 17]. The documents are scanned at 300 dpi and saved as gray scale images. Page skew is avoided manually in this step. Nearest color reduction method is used to binarize images. Processing time is approximately 4 seconds per document on a personal computer with 2.0 GHz Intel processor.

6.1 Recognition Evaluation

The library used in this study includes the 48 letters shown in Figure 2.1. All model images in the library are manually extracted from a single Ottoman document. During candidate letter determination, we discard the segments that are not similar to any letter in the library more than a predefined threshold, i.e., 0.85.

The dataset includes documents with different font sizes and thickness (see Figure 6.1). To obtain the ground truth data, the letters in these documents

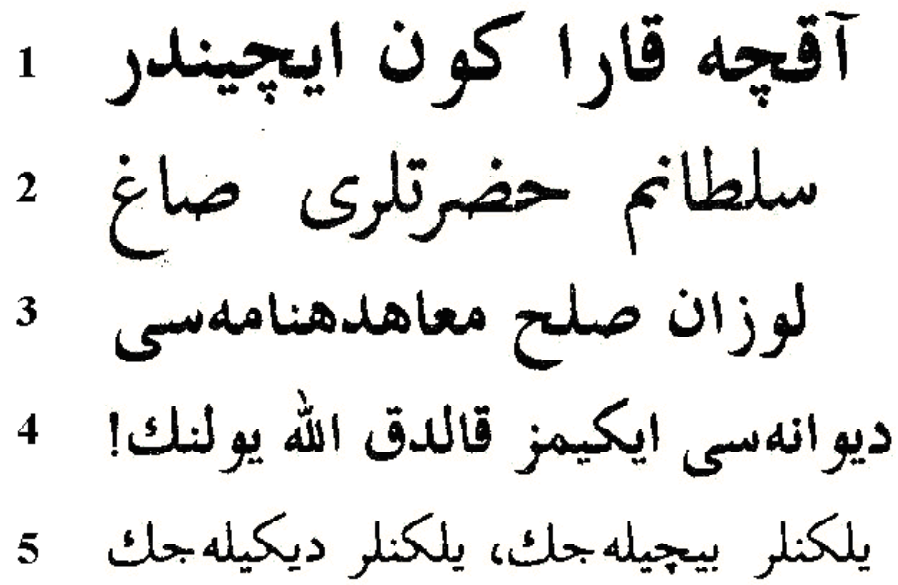


Figure 6.1: Sample lines from our dataset with different font size and letter thickness.

are manually annotated (i.e., matched against the letters in the library) by using a tool developed for this purpose in our research group. 50 randomly selected documents are used for learning unigram and bigram frequencies and the remaining documents in the dataset are used for testing. On the whole dataset, there were 34,298 annotated connected components, each of which contains 2.04 letters on the average. The experiments yield the highest recognition rates when the constant c in Equation 4.3 is set to 0.025. In Figure 6.2, the optimum value of c for our test case can be observed. As the bias introduced to the system is further increased, system performance measures (i.e., precision and recall rates) fall drastically. It is basically due to excessive promotion of links between nodes in the graph, whose occurance probabilities are higher.

In Table 6.1, we provide recognition success over train 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 dataset. The results shown in the first two rows of Table 6.1 reveal that

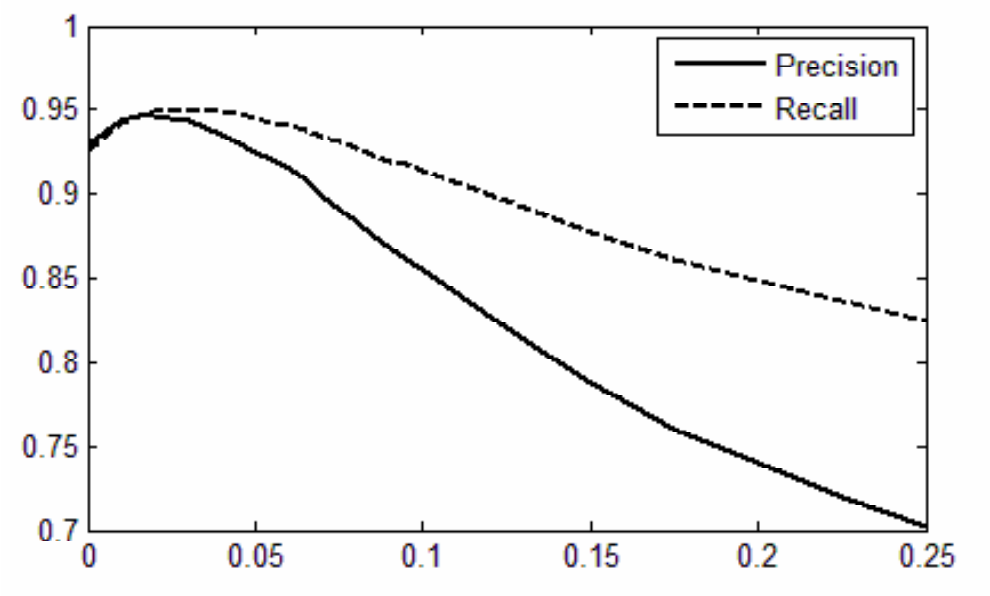


Figure 6.2: The change in recognition performance as the constant c is varied.

Collection	**Precision**	**Recall**	**Number of 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

Table 6.1: Recognition performance of the proposed method.

the proposed approach achieves very high recognition rates for both train and test sets. We also evaluate the effect of using letter frequencies in Equation4.3 by setting the c constant to 0. In this case, both precision and recall drop for the test set. This shows that, using letter frequencies is an important factor for improving overall performance. In Figure 6.3, the recognition results are shown for a sample document.

In Table 6.2, more detailed recognition statistics are given according to the number of letters in a connected component. Each connected component may include a number of letters. More than half of the components in our dataset are composed of two or more letters. In Table 6.1, if all letters are correctly recognized for a component, then it is called an exact match. If some additional

Number of letters per component	**Subset Match**	**Superset Match**	**Exact Match**	**Total number of 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

Table 6.2: Recognition rates per connected component including varying number of letters.

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 observe that this ratio is 90 percent. This result is another indication that system can be used as a building block for a content-based retrieval system for Ottoman archives.

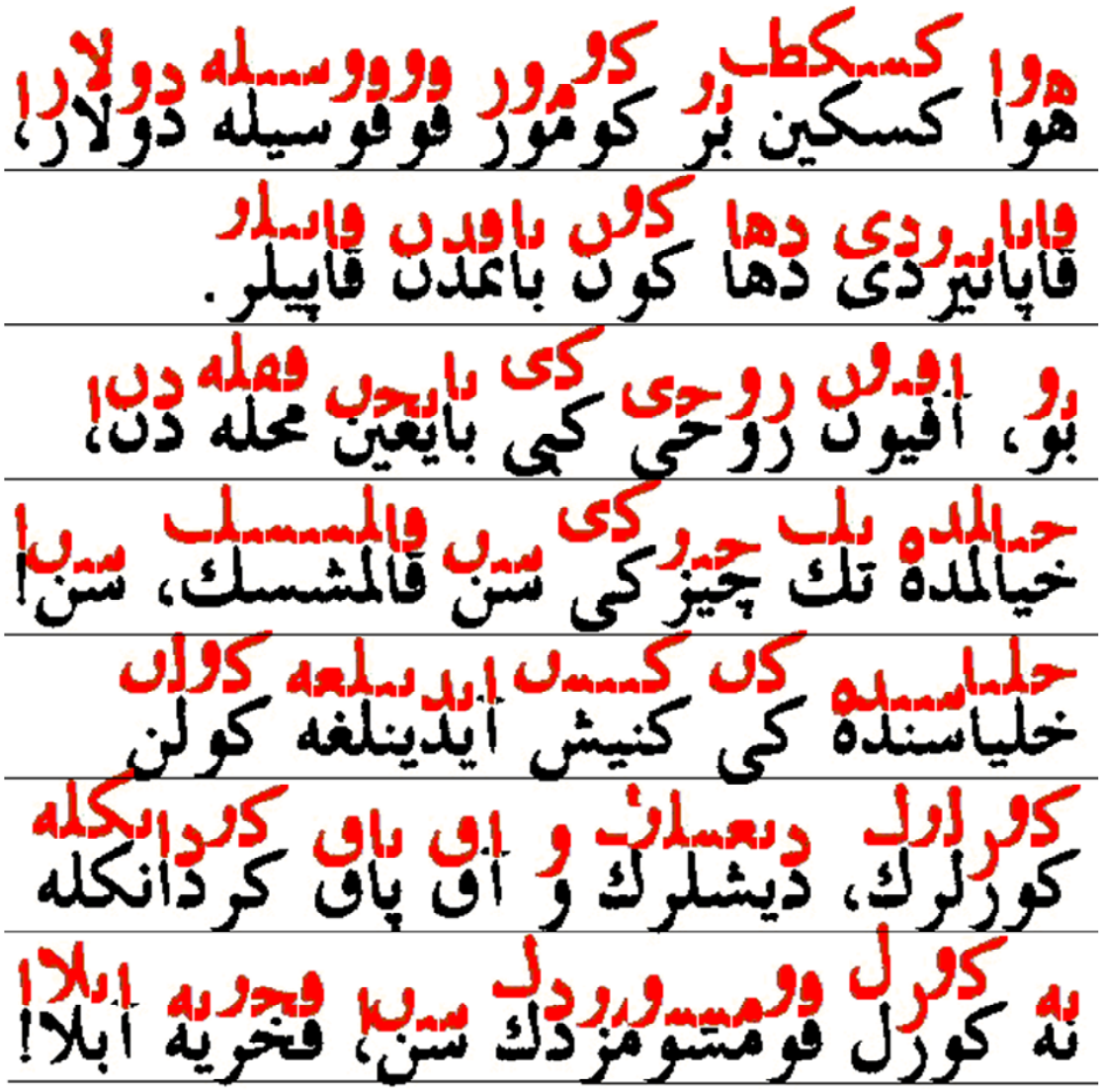


Figure 6.3: The recognition results for a sample document. For each line, the lower part includes the original script and the upper part includes the recognized letters, as recorded in the library (i.e., without dots and diacritics).

6.2 Information Retrieval Experiments

In the following experiments, the possible effect of OCR errors on IR performance is analyzed. Note that, the goal is not to evaluate the IR effectiveness on Ottoman documents, which would require a larger collection, TREC-like query topics and their relevance judgments. Instead, we solely compare for a given set of queries whether the query results significantly differ when the query is executed over the original (ground truth) document collection and its OCRed version.

Experiments are performed on both ground truth and OCRed documents as follows. As it is mentioned before, the ground truth texts are obtained by manually annotating the original documents (100 pages). These texts are then indexed by using the open source Zettair search engine. Most frequent 50 words are identified out of the 6870 index terms and used as the query set. These queries are evaluated and the top K (where K is typically 10) most-similar documents are obtained by using several different similarity measures implemented in Zettair. That is, during query evaluation, we experimented with four different similarity measures, namely Cosine [22], Okapi BM25 [13], Hawking's Okapi variant [12] and a language modeling based approach with Dirichlet smoothing [27]. For all of these measures (and all other parameters) default settings of Zettair are used. Next, OCRed versions of the same documents are obtained by using our segmentation and recognition framework. OCRed texts are also indexed in the same way and 7542 index terms are found. The same set of queries is also evaluated on the OCRed texts by using the same document similarity metrics. Ranked list of documents for each similarity metric are later compared by calculating a symmetric difference score (see Figure 6.4).

The symmetric difference score computes the similarity of the two results lists for a query, generated from the ground truth and OCRed documents, 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 only in one of the two lists, then the symmetric difference score is assigned to be 1-x/y. If the two lists are similar, the symmetric difference score is close to 1. If these two lists are totally disjoint, then the score becomes 0 [7].

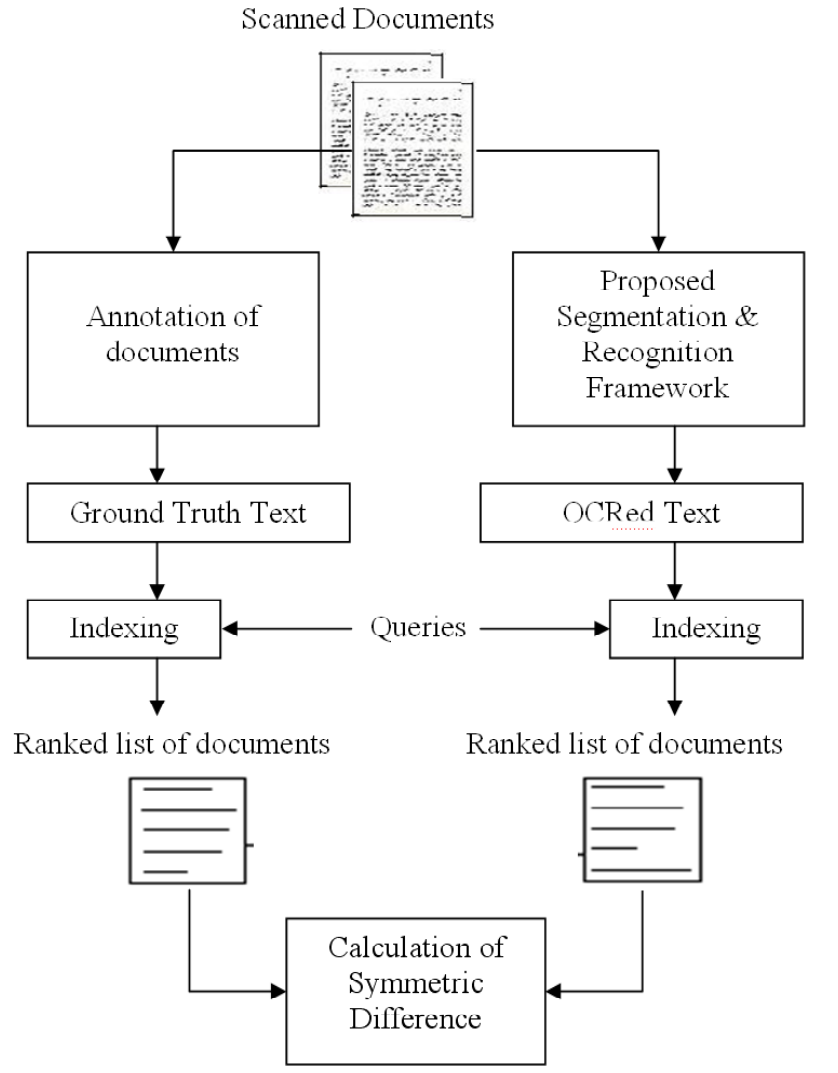


Figure 6.4: The method to test the effect of OCR errors on IR performance.

Table 6.3: Symmetric difference for the top K retrieved documents with different document 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

In Table 6.2, it is seen that top K documents that are retrieved by using the OCRed and ground truth texts 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 affect the retrieval results significantly in this case.

From the above observations we conclude that 5 percent accuracy loss in OCR as reported in Section 6.1 has a limited impact on the retrieval performance. This also conforms to the earlier findings in the literature, i.e., in [11] it is claimed most IR systems are almost unaffected by errors when OCR accuracy is higher than 95 percent.

Chapter 7

Conclusion

In this thesis, a novel method for recognizing connected letters for printed Ottoman script is proposed. Segmentation and recognition stages are integrated so that weaknesses of the classifier can be compensated by taking into account possible letter sequences for a particular connected component. The sequence of candidate letters that formulates a connected component is found by tracing the longest path over the acyclic graph constructed for the bigram approximation of our score function. Experiments show that high recognition rates are achievable by using the proposed approach. It is also experimented 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.

The framework proposed for recognition of Ottoman characters in this thesis can be generalized for other fonts and scripts as well. It is possible to learn a letter library from sample documents automatically instead of manual creation of the letter library. The letter library should involve a set of segments extracted from training documents so that all the connected components of training documents can be formulated with learned segments. These segments can later be used for recognizing the text in similar documents. It should be noted that letters may repeat in various connected components in different positions in training

documents. Therefore learned segments is also expected to match actual letters of the alphabet of the training documents.

As a future work, it is possible to improve the proposed method by integrating it with some OCR correcting schemes in the literature in order to reduce misrecognition rates. Since alternative set of candidate letters lays 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. The use of a dictionary and collection frequencies of terms in the train set may further improve the recognition rates. Another future research direction would be to see the performance of the proposed approach for other printed or handwritten datasets with varying characteristics.

Bibliography

[1] M. Allam. Segmentation vs. segmentation-free for recognizing arabic text. *Proceedings of the International Society for Optical Engineering*, 2422:228–235, 1995.

[2] I. S. Altingovde, E. Saykol, O. Ulusoy, U. Güdükbay, E. Çetin, and M. Göçmen. Osmanlı arşivleri içerik-bazlı sorgulama (ibs) sistemi. In *SUI '06: IEEE Sinyal İşleme ve Uygulamaları Kurultayı*. IEEE, 2006.

[3] A. Amin. Off-line arabic character recognition: the state of the art. *Pattern Recognition*, 31(5):517–530, 1998.

[4] E. Ataer and P. Duygulu. Retrieval of ottoman documents. In *Multimedia Information Retrieval*, pages 155–162, 2006.

[5] E. Ataer and P. Duygulu. Matching ottoman words: an image retrieval approach to historical document indexing. In *CIVR '07: Proceedings of the 6th ACM International Conference on Image and Video Retrieval*, pages 341–347, New York, NY, USA, 2007. ACM.

[6] A. A. Atici and F. T. Yarman-Vural. A heuristic algorithm for optical character recognition of arabic script. *Signal Process.*, 62(1):87–99, 1997.

[7] D. Carmel, D. Cohen, R. Fagin, E. Farchi, M. Herscovici, Y. S. Maarek, and A. Soffer. Static index pruning for information retrieval systems. In *SIGIR '01: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–50, New York, NY, USA, 2001. ACM.

[8] R. G. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(7):690–706, 1996.

[9] J. Chan, C. Ziftci, and D. Forsyth. Searching off-line arabic documents. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1455–1462, Washington, DC, USA, 2006. IEEE Computer Society.

[10] H. Develi. *Osmanlı Türkçesi Kılavuzu 1.* 3F Yayınevi, İstanbul, TURKEY, 2006.

[11] D. Doermann. The indexing and retrieval of document images: a survey. *Comput. Vis. Image Underst.*, 70(3):287–298, 1998.

[12] D. Hawking, T. Upstill, and N. Craswell. Toward better weighting of anchors. In *SIGIR '04: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 512–513, New York, NY, USA, 2004. ACM.

[13] K. S. Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.*, 36(6):779–808, 2000.

[14] P. B. Kantor and E. M. Voorhees. The trec-5 confusion track: Comparing retrieval methods for scanned text. *Inf. Retr.*, 2(2-3):165–176, 2000.

[15] M. Kantrowitz, B. Mohit, and V. Mittal. Stemming and its effects on tfidf ranking (poster session). In *SIGIR '00: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 357–359, New York, NY, USA, 2000. ACM.

[16] M. S. Khorsheed. Off-line arabic character recognition - a review. *Pattern Analysis & Applications*, 5:31–45, 2002.

[17] Y. Kurt. *Osmanlıca Dersleri 1.* Akçağ Yayınları, Ankara, TURKEY, 2006.

[18] W. Magdy and K. Darwish. Effect of ocr error correction on arabic retrieval. *Information Retrieval*, 11(5):405–425, 2008.

[19] A. Öztürk, S. Güneş, and Y. Özbay. Multifont ottoman character recognition. In *ICECS '07: Proceedings of the 7th IEEE Int. Conf. on Electronics Circuits and Systems*, pages 945–949. IEEE, 2007.

[20] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[21] E. Saykol, A. K. Sinop, U. Güdükbay, Ö. Ulusoy, and A. E. Çetin. Content-based retrieval of historical ottoman documents stored as textual images. *IEEE Transactions on Image Processing*, 13(3):314–325, 2004.

[22] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, New York, NY, USA, 1996. ACM.

[23] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[24] I. H. Witten, T. C. Bell, and A. Moffat. *Managing Gigabytes: Compressing and Indexing Documents and Images*. John Wiley & Sons, Inc., New York, NY, USA, 1994.

[25] I. Z. Yalniz, I. S. Altingovde, U. Güdükbay, and O. Ulusoy. Ottoman archives explorer: A retrieval system for digital ottoman archives. *under review*, 2006.

[26] Zettair. Zettair search engine, version 0.9.3. *Available: http://www.seg.rmit.edu.au/zettair/*, 2008.

[27] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.

[28] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2):6, 2006.

[29] J. Zobel, A. Moffat, and K. Ramamohanarao. Inverted files versus signature files for text indexing. *ACM Trans. Database Syst.*, 23(4):453–490, 1998.

Appendix A

Ottoman Archives Explorer

Figure A.1 shows the Web interface of Ottoman Archives Explorer. It is a prototype content-based retrieval system for digital Ottoman archives. The user can specify a query in two different ways, either by browsing through the available documents and selecting an example region to be searched for, or directly entering the query word in the virtual keyboard, which is at the right-hand side frame of the user interface. Figures A.2 and A.3, illustrate the former method, example-based querying (QBE). The user selects a rectangular area including one or more components, as shown in Figure A.2. The system consults to the corresponding document's codebook to obtain the recognized characters ids' within the coordinates of the given region and than looks for the same or similar character sequences in all other codebooks. Finally, matches are returned to user in the order of similarity. Figure A.3 shows the query result for the example. While computing the similarity score, each exact matching component is weighted by 1; each component that is a superset or subset of the query component is weighted by 0.5; and each component that has overlapping characters with the query components are weighted by 0.25.

In Figure A.1, the use of virtual keyboard can also be seen. In this case, the user enters the query using the alphabet including all forms of characters (e.g., isolated, beginning, middle and end) so that the word may be constructed with correct character sequence (but of course the connecting lines or curves between

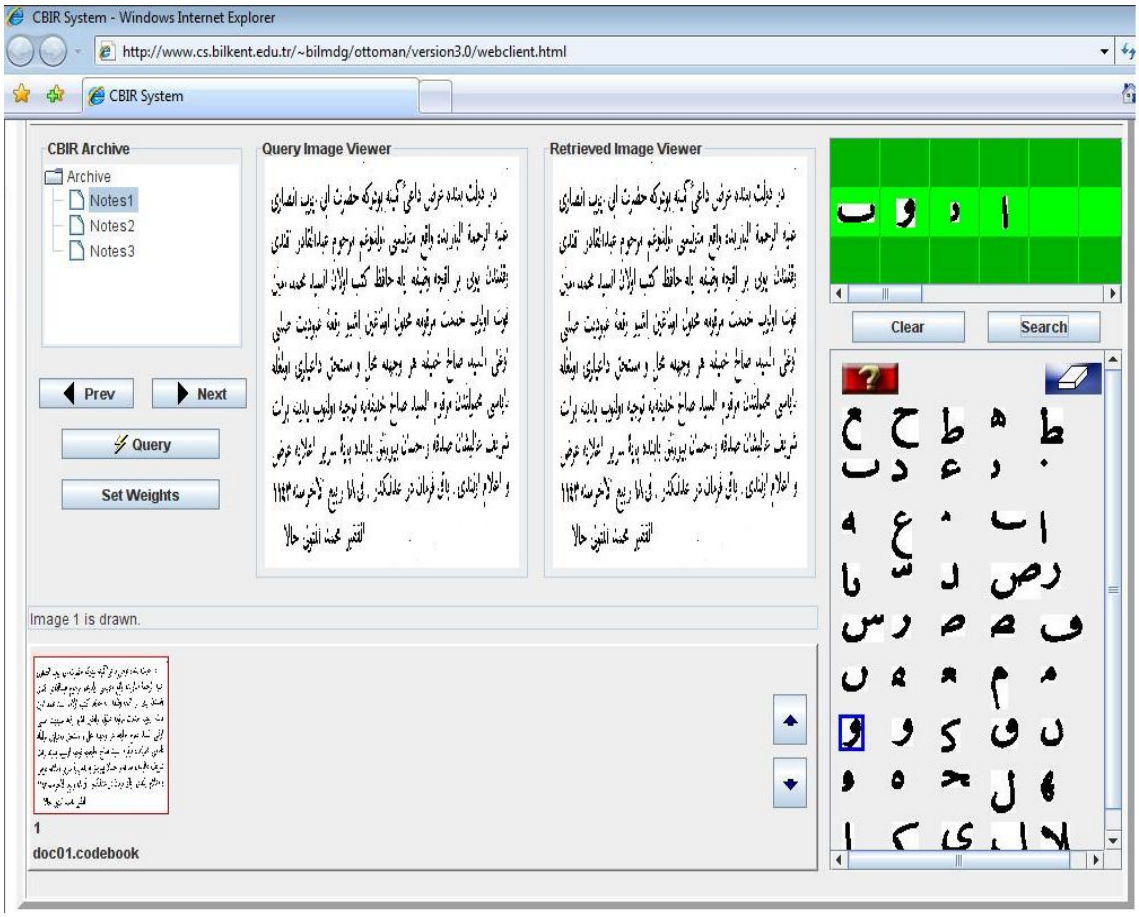


Figure A.1: Ottoman Archives Explorer Web user interface.

the characters are not shown). The query grid includes three lines, the middle one is used for entering the actual query word and the upper and lower lines are for the dots or other special characters that are above or below the characters in the query word [25].

The Ottoman Archives Explorer prototype system is accessible online: `http://www.cs.bilkent.edu.tr/~bilmdg/ottoman/demo.htm`.

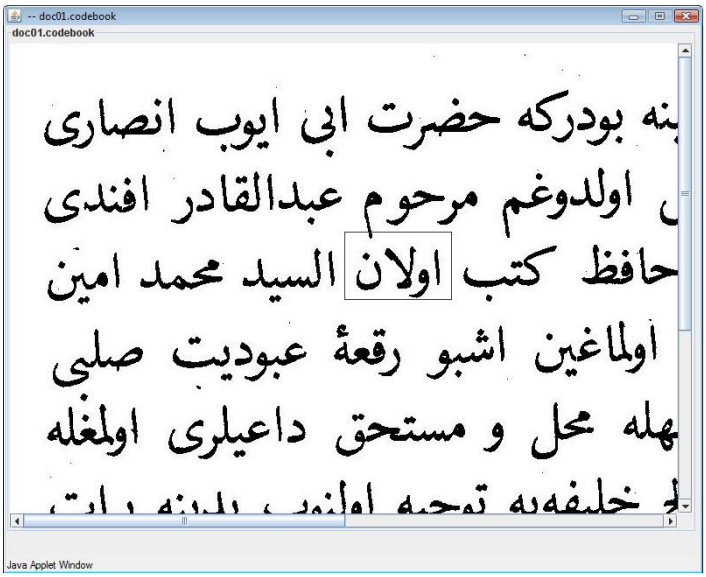


Figure A.2: A rectangular region over the document selected by the user for querying. Letters recognized inside the rectangle are used for query resolution.

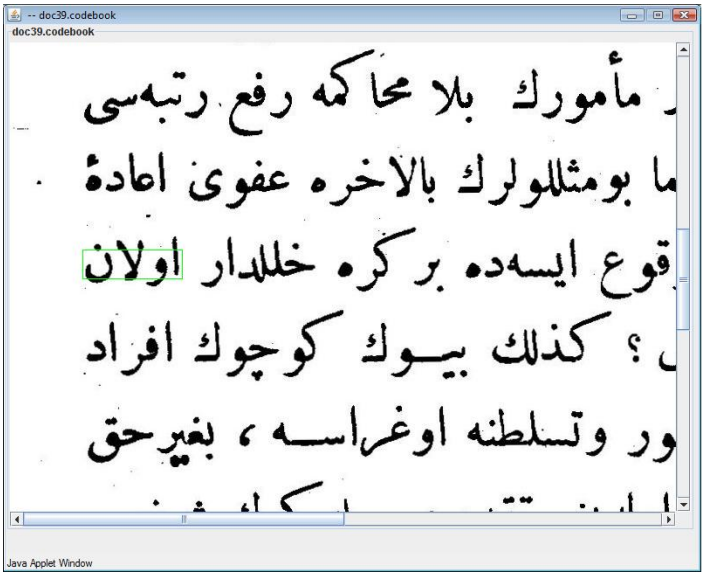


Figure A.3: A matching word is shown in the rectangle in a matching document for the query formulated in Figure A.2.

Appendix B

Annotation Tool

Figure B.1 shows the annotation tool with a sample Ottoman document. At the first step of the annotation, the document image is processed by the proposed OCR framework and an annotation file is automatically generated. Next, the annotator fixes the mistakes in recognition in order to create the ground truth data. In this way, it becomes easier and consequently faster to annotate large number of pages, especially when recognition rates are high. Annotation is performed by simply clicking on misrecognized connected components and selecting the correct letter(s) by using the letter library on the right of the screen. The annotation process for a document can be saved and resumed later.

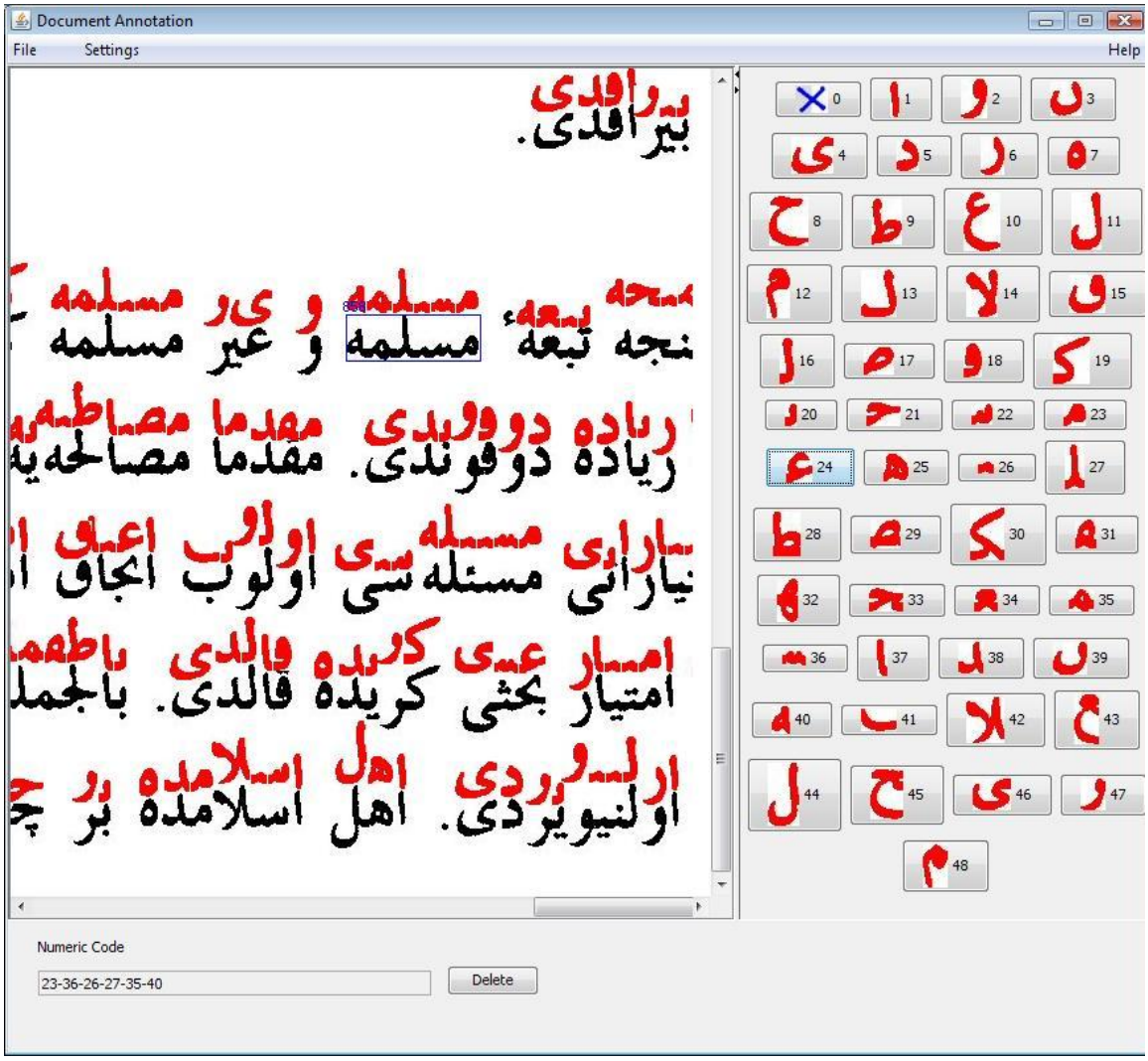


Figure B.1: The user interface of the annotation tool. The selected component (shown in the rectangle) is annotated with the letter ids 23, 36, 26, 27, 35 and 40 as entered to the textbox at the bottom. Annotated letters are redrawn over their connected components in order to make the annotater to see annotated letters better.

Appendix C

Codebook Viewer Tool

Figure C.1 depicts the codebook viewer. It shows the characters in the library at the right hand-side panel and the character id on top of each recognized segment in the main panel. This tool is useful for having a visual understanding of the performance of the character recognition task. Furthermore, this tool may be used to gather relevance feedback from the users, to improve the success of recognition [25].

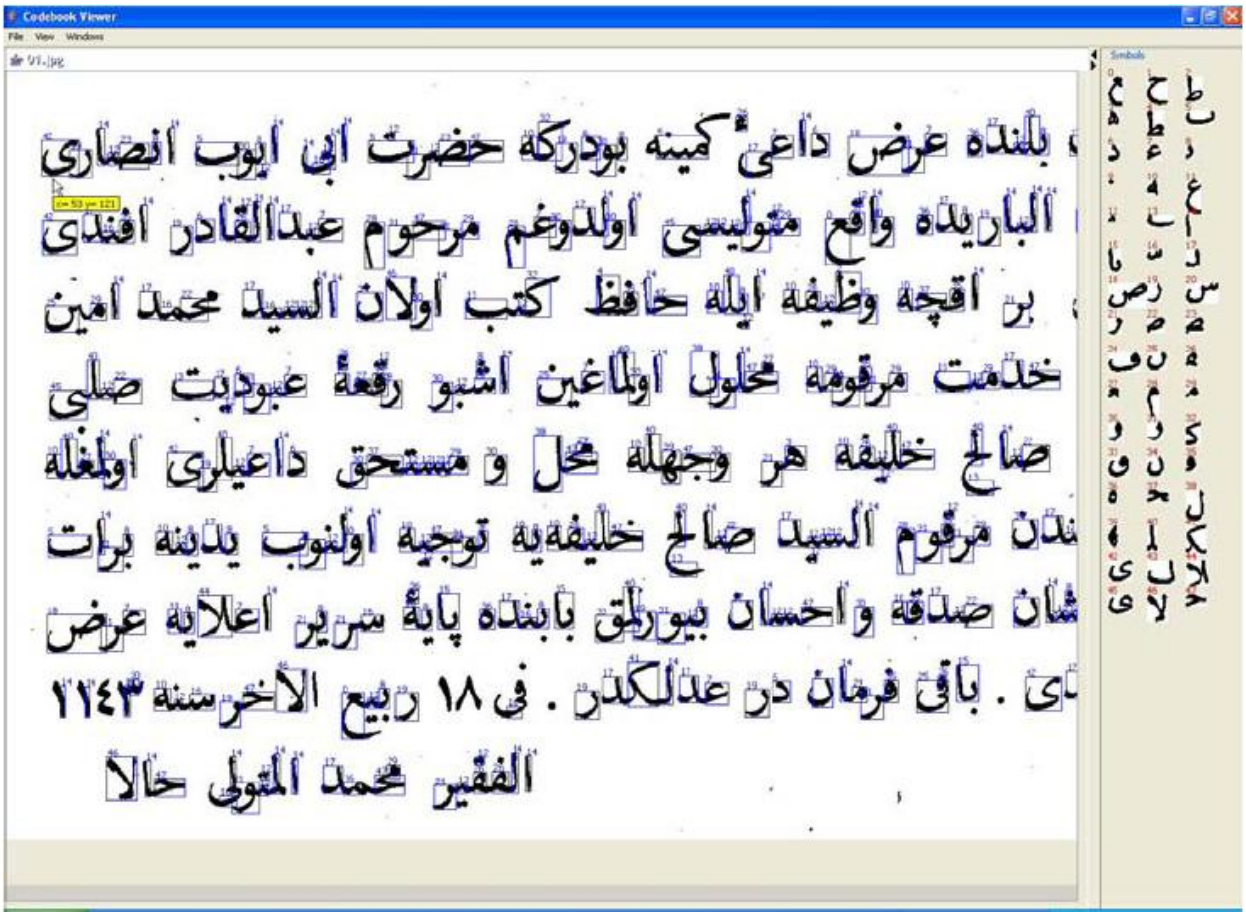


Figure C.1: Codebook viewer user interface.