Dependence models for searching text in document images

Ismet Zeki Yalniz and R. Manmatha

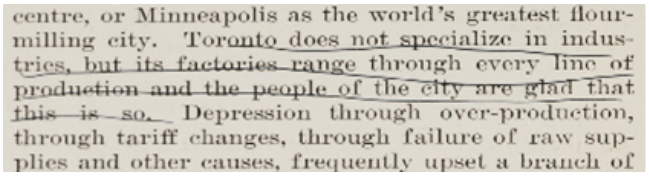
Abstract—The main goal of existing word spotting approaches for searching document images has been the identification of visually similar word images in the absence of high quality text recognition output. Searching for a piece of arbitrary text is not possible unless the user identifies a sample word image from the document collection or generates the query word image synthetically. To address this problem, a Markov Random Field (MRF) framework is proposed for searching document images and shown to be effective for searching arbitrary text in real time for books printed in English (Latin script), Telugu and Ottoman scripts. The English experiments demonstrate that the dependencies between the visual terms and letter bigrams can be automatically learned using noisy OCR output. It is also shown that OCR text search accuracy can be significantly improved if it is combined with the proposed approach. No commercial OCR engine is available for Telugu or Ottoman script. In these cases the dependencies are trained using manually annotated document images. It is demonstrated that the trained model can be directly used to resolve arbitrary text queries across books despite font type and size differences. The proposed approach outperforms a state-of-the-art BLSTM baseline in these contexts.

Index Terms—Markov Random Fields, document image search, image retrieval, word spotting

1 Introduction

One way to search printed document images is to recognize characters and perform text queries. This approach is feasible as long as the document is not noisy and recognition accuracy is high enough for searching text. For noisy documents, Optical Character Recognition (OCR) accuracy can be very low and recognized text becomes unusable. One remedy is to use error correction schemes [1], [2], [3]. Another option is to compensate for OCR errors in the query resolution stage [4]. For example searching for n-grams of letters is shown to improve retrieval accuracy [5], [6]. One can also use both of these approaches to improve text search. However, these methods have limited capability in the sense that they cannot retrieve information which is not captured by the OCR engine. In such cases, the information is permanently lost and there is no way to recover it. Figure 1 shows an example page image where recognition errors are severe and they cannot be fixed using OCR error correction schemes. The proposed image match framework addresses this problem by using image based features for searching text in document images. The proposed approach is able to efficiently resolve arbitrary text (i.e., unlimited vocabulary) in document images even if there is no OCR engine available for the script or OCR fails due to document noise. Combining n-gram based OCR text search with the proposed image search framework further improves text search due to their complementary strengths.

Image search based word spotting approaches make use of image features directly in the search process. Various approaches have been proposed for text search on noisy printed or handwritten documents [8], [9], [10], [11] but they have limitations. One important limitation is that it is not possible to perform arbitrary queries. Users have to find an example word image from the same document collection and use it for querying (query by example, QBE). Query words which are out of vocabulary are therefore problematic. Scalability is yet another issue since computationally intensive operations are performed over high dimensional feature vectors for each query. Efficient indexing and retrieval schemes for large document collections are needed.



(a)

```
centre, or Minneapolis as the world's greatest flourmi
lling city. Toron to does no _LÃ£Â£>Â£ci arize in industries,
but its factories ra^oro ti-ivmiprVi every~lme of
^ji'lm4m> n"A +lâ„¢ pÂ«^p1c of _thecity are glacTtlia t
llux â€" i*-â€" sll__ Depression through over-production,
through tariff changes, through failure of raw supplies
and other causes, frequently upset a branch of
```

(b)

Fig. 1. a) a page image from the book "Tremendous Toronto" from the Internet Archive [7], b) their provided ABBYY OCR output. Crossing out text leads to segmentation and therefore OCR errors.

Here, an efficient image search framework is proposed for searching arbitrary query words in the document images. The proposed approach has two main components: i) efficient processing and matching of visual features and ii) a discrete Markov Random Field (MRF) retrieval model to rank relevant word images given a text query. The first stage is to extract visual features from the word images in the document collection. This is achieved by placing a square shaped image patch on each corner point located in

• *The authors carried out this work while at the Department of Computer Science, University of Massachusetts, Amherst, MA, 01003. E-mails: {zeki, manmatha}@cs.umass.edu*

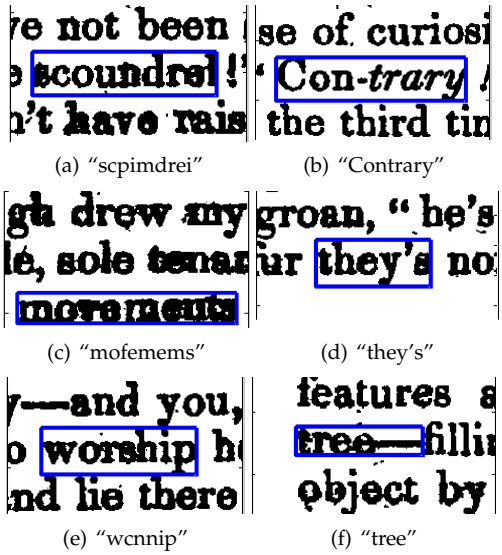
the word image. Visual features extracted from each local image patch are quantized into visterms or visual words for efficiency. The visterms along with their positions on the image plane are used by the retrieval model to resolve arbitrary text queries.

In the query resolution stage, the task is to rank the word images in the document collection according to their relevance to the query word. A discrete MRF model (also referred to as a "dependence" model) is proposed for determining the relevance of each test image to the query word by looking at the dependencies between the visual terms and the query letter bigrams. More specifically, the query word is decomposed into its letter bigrams and each letter bigram is regarded as a random variable in the dependence graph along with the visual terms in the word image. Following the Markov assumption, two types of dependencies are defined between the random variables in the dependence graph: (i) between all query letter bigrams and visual term pairs in the test image, (ii) between all distinct pairs of query letter bigrams. The first set of dependencies are learned offline from labeled data and used for determining the existence of the letter bigram in the test image. For this task, the "union" and the "intersection" probability estimation models are proposed. The second set of dependencies account for the order of the letter bigrams in the word image and require no training. Given a text query, all the word images in the collection are ranked according to the final MRF score which accounts for all types of dependencies defined in the graph.

The proposed dependence model is shown to be effective for searching text in books printed in English, Telugu and Ottoman scripts. For searching English books, the dependencies between the visual terms and letter bigrams are automatically learned using noisy OCR output. OCR text search accuracy is significantly improved by combining with the proposed approach. The proposed approach outperforms a state-of-the-art Bidirectional Long-Short Term Memory (BLSTM) baseline model when both of them are combined with OCR text search. Since there are no commercial OCR engine available for Telugu and Ottoman scripts, the dependencies are trained using manually annotated document images. For these scripts, the trained model is directly used to resolve arbitrary text queries in other scanned books despite font type and size differences. The BLSTM baseline provides poor results for Telugu and Ottoman scripts. The proposed framework has a real time performance for searching an entire book containing hundreds of pages with a speed of under 5 ms/query on a single core.

To our best knowledge, the proposed framework is also the first image search approach which resolves arbitrary text queries in document collections without any search ambiguity. This is discussed further in the related work section. Figure 2 shows example word images for a number of query words on a book "Wuthering Heights" by Emily Brontë. The proposed combined text search approach (OCR text baseline + image search) retrieves all six word images correctly along with other true positive instances at the top of the ranked list.

The rest of the paper is organized as follows: a literature overview is provided in Section 2. Section 3 describes the visual features used. Section 4 elaborates on the proposed dependence model for searching arbitrary text in document images. Experimental results and future research directions are discussed in Sections 5 and 6, respectively.



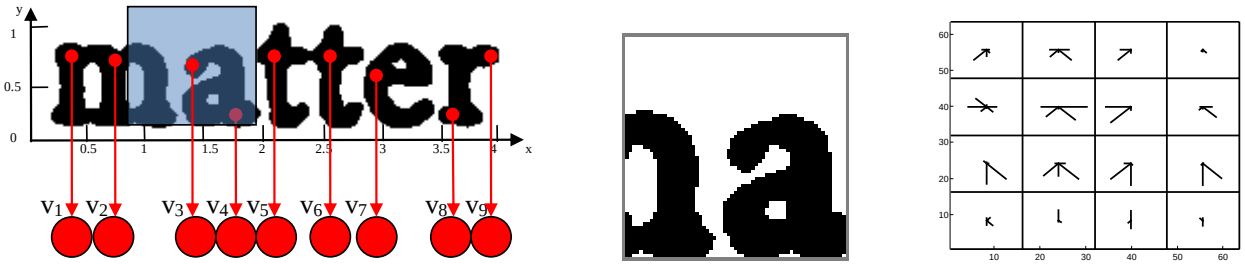
(a) "scpimdrei" (b) "Contrary" (c) "mofememes" (d) "they's" (e) "wcnnip" (f) "tree"

Fig. 2. Word image examples: a), c), e) correctly retrieved by the proposed framework but missed by the OCR text search baseline due to OCR errors as shown. b), d), f) correctly recognized by ABBYY OCR engine and retrieved by OCR text search baseline but missed by the proposed approach. The combined approach (image + OCR text search) correctly retrieved all these word images.

2 Literature Overview

Searching text in document images without explicit character or word recognition is referred to as "word spotting" in the literature [12]. More specifically, a query word image is given and the task is to search for other instances of the same word in other documents using raw image features. Several word spotting approaches have been proposed for printed [13], [14] and handwritten documents [8], [9], [10], [11], [15], [16], [17]. These approaches have been shown to be effective especially for searching text in degraded historical documents. The most important drawback for word spotting systems is that a query image is required for each query word. Therefore a user cannot search for arbitrary text unless the word image is available.

Word spotting frameworks mainly differ from each other in three ways: the word image segmentation method, the image features used, and the word image matching/retrieval approach. Projection profiles [18], scale-space approaches [19], Hough-based methods [20] and gap metrics [21] have been applied for automatically segmenting text lines and word images. Several image features have been proposed for representing word images including variants of projection profiles, DFT features extracted from projection profiles, ink transitions, gray level variance and local gradient histograms [18], [22]. Rath and Manmatha [10] proposed Dynamic Time Warping for matching word images represented by projection profiles. Other methods include aligning the word images and computing a similarity based on pixel wise comparisons using XOR, Sum of Squared Distances and Euclidean Distance Mapping.



(a) Visual terms for a given word image (b) A local image patch (c) Corresponding SIFT features

Fig. 3. Visual features are shown for an example word image. In a) small dots correspond to the local interest points. Local patches extracted from interest points are quantized into visual terms $(v_1, v_2, ...v_9)$ which are represented with large big circles at the bottom. b) and c) shows an example image patch from the word image and the corresponding SIFT features respectively. There are typically around 100 visual terms per word image.

Graves et al [23] proposed the use of a Bidirectional Long Short Term Memory (BLSTM) model for handwritten recognition. Their approach has two stages. First the probability of each character is estimated for each location of an input sequence. Second, this sequence of probabilities for the letter is used along with a language model and a dictionary as input to a CTC Token Passing algorithm and the output is a sequence of words. A modified version of the CTC algorithm was used in Frinken et al [24] for the handwritten word spotting problem. The modification was necessary since the input sequence was an image of an entire text line. We refer the reader to [23], [24] for a detailed explanation of the BLSTM networks. In this work, a BLSTM baseline is also adopted and elaborated further in the experiments section.

The word spotting paradigm has also been extended to perform holistic word recognition. Given a query word image whose text content is known, one can propagate the text label to other visually similar word images in the document set. Marinai et al. [25] and Pramod et al. [14] use clustering techniques to group similar word images and the word images are labeled based on which cluster they belong to. However, manual labeling of the word image clusters is not practical for large datasets with diverse fonts and writing styles. In addition, a major limitation of these approaches is that they cannot label word images which are not in the vocabulary of recognizable words.

Lu et al. [26] adopt a word shape coding approach for searching text in document images given a text query. It is reported that these shape code collisions happen 28% of the time for a dictionary of size 50K words [26]. Shape collisions cause ambiguous search results and are known to be sensitive to subtle ink deformations.

3 Visual Features

OCR engines rely on features extracted from connected components and tend to make errors in recognizing underlined or crossed out word images which are very common noise types in real/noisy document images. Image search mechanisms are capable of accounting for partial matches between the word images for both search and retrieval tasks by using features which are robust to document noise and similarity functions which account for partial matches.

In this paper we adopt the local visual features proposed in [27]. "Keypoints" are identified in the document images using the Fast-Corner-Detector [28]. SIFT [29] features are extracted from each local patch placed over the corner points. The scanned page images do not have significant page skew so the patch orientation is set to zero degrees. In the original paper [27], the patch size is set to be equal to the height of the box for the purpose of holistic matching of noisy word images. In this work, two other patch scale estimation approaches are also investigated: a) fixing the patch size (if the font size is uniform across the documents) and b) determining the patch size relative to the line height.

Hierarchical K-Means (HIKMEANS) [30] is used for quantizing SIFT descriptors to visual terms as described in [27]. Large visual vocabularies provide better results for instance matching of natural images [30]. However, smaller vocabularies (4K terms) are observed to provide higher matching performance for other retrieval tasks and do best for searching text in document images [27].

After feature extraction, a word image I is represented with a set of visterms $\{v_1, v_2, \ldots, v_m\}$. Each visterm v_i is represented by a discrete number and its (x, y) coordinates in the word image frame I (normalized by box or line height). The visterms are stored in order according to their x coordinate (see Figure 3). As shown an image patch placed over any corner point covers ink from one or more characters. Partial matches are quite useful for searching text in noisy document images [27].

4 A Discrete MRF Model for Searching Arbitrary Text in Document Images

Our proposed approach adapts the general MRF framework proposed by Metzler and Croft for text retrieval [31]. This framework has been widely used and also shown to be effective for the photographic image retrieval problem [32].

Searching text in document images is different from text retrieval. An arbitrary text query Q (such as "Sherlock") and visual features for each word image I in the book are given. The task is to rank all the word images according to their relevance to the query word $P(I|Q)$. In our framework, the query word is decomposed into its letter bigrams q_i and the posterior probabilities $P(I|q_i)$ are estimated efficiently for each word image and are later combined to estimate $P(I|Q)$. After elaborating on the general MRF framework ([31], [32]), the details of the proposed framework are discussed in the subsections below.

Markov random fields (MRFs) are useful for modeling the joint distribution of a set of random variables. In a

nut-shell, a Markov random field is an undirected graph, where nodes represent random variables. Edges between nodes represents conditional dependencies between random variables. Based on the Markov property, it is assumed that certain random variables are independent of all others. Dependencies are therefore defined between certain groups of random variables. These groups are called "cliques". For each type of clique c in the graph, a non-negative potential function $\phi_{c;\Lambda}$ is defined. These functions are parameterized by Λ (capital lambda symbol) and they are used for estimating joint probabilities.

The ultimate aim is to calculate the posterior probability $P_\Lambda(I|Q)$ for all word images in the collection and then rank them based on their relevance to the query. We follow the derivation in [31] to estimate the joint probability $P(Q, I)$:

$$P(Q,I) = \frac{1}{Z_\Lambda} \prod_{c \in C(G)} \phi(c;\Lambda) \tag{1}$$

where Z_Λ is:

$$Z_\Lambda = \sum_{Q,I} \prod_{c \in C(G)} \phi(c;\Lambda) \tag{2}$$

It is computationally expensive to compute Z_Λ. In our case, the problem is to rank word images based on their posterior probabilities $P_\Lambda(I|Q)$, so we can ignore the constant Z_Λ.

Once the normalizing constant is ignored, estimating posterior probabilities becomes much easier. According to [31], [32], posterior probabilities may be estimated as follows:

$$\begin{aligned} P(I|Q) = \frac{P_\Lambda(Q,I)}{P_\Lambda(Q)} &\overset{rank}{=} \log P_\Lambda(Q,I) - \log P_\Lambda(Q) \\ &\overset{rank}{=} \sum_{c \in C(G)} \log \phi(c;\Lambda) \end{aligned} \tag{3}$$

where $\overset{rank}{=}$ indicates rank equivalence. The resulting formula is a sum of logarithm of potential functions over all cliques. The potential function is often assumed to have the following form:

$$\phi(c;\Lambda) = \exp[\lambda_c f(c)] \tag{4}$$

where $f(c)$ is some feature function over clique c, and λ_c is the weight of this particular feature function. Then the ranking function simplifies to:

$$P_\Lambda(I|Q) \overset{rank}{=} \sum_{c \in C(G)} \lambda_c f(c) \tag{5}$$

which is a linear function over feature functions and may be computed efficiently. λ_c is a weight factor and it is defined for each clique in the MRF model.

4.1 The proposed framework

Both letter bigrams and their order are important to qualify a word image for being a match. Here we devise an MRF model so that both conditions are satisfied. We assume that all visual terms v_i are independent of each other given the query word Q. We do not use higher order dependencies because training is impractical given the dimensionality of all bigram letter classes (4K for English) and the visterm vocabulary size (4K in our experiments). Two types of

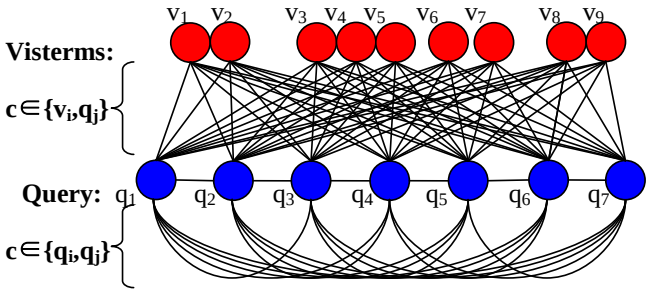


Fig. 4. The configuration of our MRF model for searching arbitrary text in document images. Red nodes (top) represents the visual terms $v_1, v_2, ..., v_9$ extracted from the test image. Blue nodes (bottom) denotes the letter bigrams $q_1, q_2, ..., q_{(n+1)=7}$ of the example query word "matter" with n characters. The dependencies between random variables are shown with straight lines.

cliques are defined in our model. The first type vq consists of all pairs between visterms v_i of the word image I and letter bigrams q_j of Q. Q includes $(n+1)$ character letter bigrams including the space characters at the beginning and the end of the word. The second set of cliques qq include all letter bigram pairs in Q.

The final MRF score based on combining clique potentials for different types of cliques is:

$$P_\Lambda(I|Q) = \lambda_M NMRF_{vq} + (1-\lambda_M) MRF_{qq} \tag{6}$$

where λ_M is a parameter whose range of values is defined to be [0,1]. $NMRF_{vq}$ stands for normalized MRF score for the sum of clique potentials of type vq. Similarly MRF_{qq} stands for the sum of clique potentials of type qq. The estimation procedure for these scores is explained in the following subsections. The configuration of our discrete MRF model is depicted in Figure 4.

4.1.1 Modeling visterm-letter bigram dependencies

The posterior probability of a word image I given a query word is formulated as follows:

$$MRF_{vq} = P_\Lambda(I|Q) = P_\Lambda(v_1, v_2, \ldots, v_m | q_1, q_2, \ldots, q_n) \tag{7}$$

where v_i and q_j correspond to visterm i and letter bigram j respectively in the query word. Using Eq.5, Eq.7 becomes:

$$MRF_{vq} = \sum_{c \in \{v_i, q_j\}} \lambda_c f_{vq}(c) \tag{8}$$

where c stands for a clique formed by a visterm v_i in I and a letter bigram q_j in Q. The feature function $f(c)$ is defined to be the posterior probability of q_j given v_i and reformulated using Bayes' rule as:

$$f_{vq}(c) = \Pr(q_j|v_i) = \frac{\Pr(v_i|q_j)\Pr(q_j)}{\Pr(v_i)} \tag{9}$$

where Pr denotes probability distributions.

We slide a Gaussian window as shown in Figure 5 to determine the location of each letter bigram in the word image - this makes it easier to handle varying text font widths. It is possible to incorporate a Gaussian window into our MRF model by varying the values of λ_c as follows:

$$\lambda_c = G_{\mu,\sigma}(x_i) \tag{10}$$

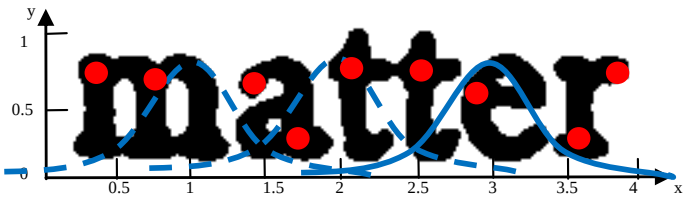


Fig. 5. Sliding a Gaussian window along the horizontal axis of the image plane is illustrated for an example word image.

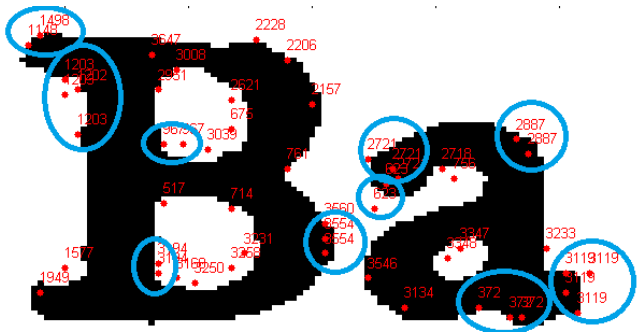


Fig. 6. Visual terms and their positions are shown for a sample image containing the letter bigram "Ba". Circles indicate the groups of visterms with the same visterm ID. The visual terms extracted from nearby locations have quite similar feature vectors, therefore they end up having the same visterm ID.

where x_i is the height normalized coordinate of visterm i on the X axis of the word image. The Gaussian window is parameterized by μ and σ. μ designates the location of the letter bigram q_j and σ determines the visual term v_i's weight given its position along the X-axis. The aim is to find a value μ for each q_j so that Eq.7 is maximized:

$$MRF_{vq} = \sum_{c \in \{v_i, q_j\}} G_{\mu_{q_j},\sigma}(x_i) f_{vq}(c) \tag{11}$$

and the estimated location of letter bigram q_j in the word image is given by:

$$\mu_{q_j} = arg\max_{\mu} \sum_{c \in \{v_i, q_j\}} G_{\mu,\sigma}(x_i) f_{vq}(c) \tag{12}$$

Often visterms with identical visterm ID's are right next to each other (see Figure 6). This is an artifact of the keypoint detectors and they are not desirable. A remedy for this problem is to account for the existence but not the frequency of visterms in word images using a Bernoulli Model [33]. We only account for the visterms whose λ_c weight is the highest for a given μ and a Bernoulli Model is adopted for estimating probabilities as described in Section 4.2.

Since each visterm class can contribute to the sum at most once and $Pr(q_j|v_i)$ is a distribution over query bigrams, the upper bound for $\sum_{c \in \{v_i, q_j\}} \lambda_c f_{vq}(c)$ becomes $G_{\mu,\sigma}(\mu)$. As a result, the range of values for MRF_{vq} becomes $[0, |Q| G_{\mu,\sigma}(\mu)]$, where σ is a parameter. It is not desirable to have different ranges of values for queries varying in length. $NMRF_{vq}$ is the MRF score normalized by the query length $|Q|$ and is given by:

$$NMRF_{vq} = \frac{1}{|Q|} \sum_{c \in \{v_i, q_j\}} G_{\mu_{q_j},\sigma}(x_i) f_{vq}(c) \tag{13}$$

4.1.2 *Modeling the order of letter bigrams*

The order of letter bigrams in the word image is constrained using a clique potential for letter bigram pairs based on Eq.5 as:

$$MRF_{qq} = \sum_{c \in \{q_i, q_j\}} \lambda_c f_{qq}(c) \tag{14}$$

where

$$f_{qq}(c) = \begin{bmatrix} 1 \text{ if } q_i \text{ and } q_j \text{ are in correct order} \\ 0 \text{ otherwise} \end{bmatrix}$$

$$\lambda_c = \frac{1}{n(n-1)/2} \tag{15}$$

and n is the number of letter bigrams in the query word. Each clique is equally weighted in a way and MRF_{qq} score's range is set to $[0, 1]$. Note that the estimated location of letter bigrams μ_{q_j} is used to determine the correct order compared to the original query word.

4.2 Probability estimation

Each word image in the training set C is represented by a set of visterms $\{v_1, v_2, \ldots, v_m\}$ and a set of letter bigrams $\{q_1, q_2, \ldots, q_n\}$. It is assumed that there is at least one character and one visterm. For a word with $n-1$ characters, there are n letter bigrams. A training set may be synthetically rendered with a large set of word images or individual letter bigrams in various fonts and sizes. Alternatively, the recognition output of an OCR engine on sample document images may be used for automatic training.

Given a training set, the aim is to learn $\Pr(v_i|q_j)$, $\Pr(v_i)$ and $\Pr(q_j)$ in Eq.9 for all visterm and letter bigram classes. In this work, it is assumed that $\Pr(q_j)$ is uniform, meaning that each letter bigram class is equiprobable. Similarly, $\Pr(C_k)$ is also assumed to be uniform. In other words, each word image in the collection is equiprobable.

A multiple Bernoulli model is adopted for learning $\Pr(v_i|q_j)$ and $\Pr(v_i)$. According to the model, the existence of visterms in a word image is important, not their respective frequencies. In other words, the probability of $P(v_i|I)$ is estimated using a discrete Kronecker delta function:

$$P(v_i|C_k) = \delta_{v_i,C_k} \tag{16}$$

where $\delta_{v_i,C_k} = 1$ if a visterm v_i occurs in the representation of the word image C_k.

Given a training collection C, $P(v_i)$ is calculated by marginalizing v_i over the entire collection C:

$$P(v_i) = \sum_k P(v_i|C_k) P(C_k) \tag{17}$$

where C_k is a word image in C. Similarly $P(v_i|q_j)$ is calculated by marginalizing v_i over the set of all word images in C which contain letter bigram q_j:

$$P(v_i|q_j) = \sum_k P(v_i|C_k, q_j) P(C_k|q_j) \tag{18}$$

This method is referred to as the **Union Model**. One problem with the union model is that it simply blends all the visterms of training images which contain the letter bigram q_j for learning $P(v_i|q_j)$. However, some visterms in the training image C_k may not be associated with q_j in particular but other letter bigrams in the word image. It is

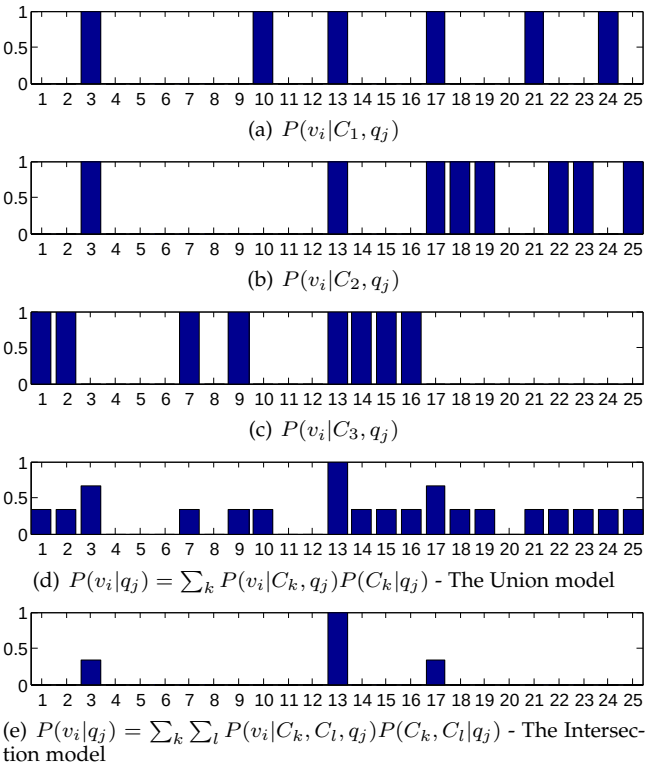


Fig. 7. The learning models illustrated for learning the probability distributions of visterms for the letter bigram class q_j from three training word images C_1, C_2 and C_3. The visterm distribution of visterms for each training sample are shown in a), b) and c). The horizontal and vertical axes represent the visterm IDs v_i and the corresponding probability respectively. Estimated visterm distributions for the letter bigram class q_j are shown using d) the Union and e) the Intersection learning models.

desirable to differentiate the visterms which are particular to letter bigram class q_j and use only them for estimating posterior probabilities.

Another method referred to as the **Intersection Model** is introduced for estimating $P(v_i|q_j)$. It gives less importance (weights) to visterms which belong to letter bigram classes other than q_j. The idea is to intersect visterms of word image pairs which are known to contain letter bigram class q_j. Visterms in the intersection are meant to be specific to q_j and therefore it is safe to use them for probability estimations. Formally, $P(v_i|q_j)$ is estimated by marginalizing v_i over all pairs of word images containing q_j:

$$P(v_i|q_j) = \sum_k \sum_{l \neq k} P(v_i|C_k, C_l, q_j)P(C_k, C_l|q_j) \qquad (19)$$

Assuming that training images C_k and C_l in C are independent of each other, Eq.19 becomes:

$$P(v_i|q_j) = \sum_k \sum_{l \neq k} P(v_i|C_k, q_j)P(v_i|C_l, q_j)P(C_k|q_j)P(C_l|q_j) \qquad (20)$$

$P(C_k|q_j)P(C_l|q_j)$ is equal to one if q_j occurs in both images C_k and C_l, zero otherwise. The Intersection method discards visterms which occur only once among all instances of word images containing q_j. This is desirable since such visterms are very likely to be products of document noise and/or discretization errors.

Figure 7 is an illustration of the idea of the proposed learning models for training the visterm distribution of the query letter bigram q_j ="th". Assume that there are only three training instances of word images C_1, C_2 and C_3 containing the letter bigram "th" corresponding to the words "their", "another" and "without" respectively. Each training image is also associated with a number of visual terms denoted by v_i. The training images contain visual features for not only the letter bigram class "th" but also others such as "he" and "an". In this example, each letter bigram is assumed to be directly related to only one visual term and the size of the visual vocabulary is set to 25 for illustration purposes. If a visual term appears in the training image, then the corresponding value $P(v_i|C_k, q_j)$ in the visterm distribution is set to one in the bar graphs shown in Figure 7 a), b) and c). Otherwise the value is set to zero. The first training image C_1 contains six visterms whereas the other two images contain eight visterms each.

Figure 7d) shows the distribution of visterms estimated by the Union model. Assuming that each training instance is equally likely, the Union model simply averages the corresponding probabilities for each visterm to learn the visterm distribution for the letter bigram class "th". Visual term v_{13} appears in all training images and therefore the estimated value for $P(v_{13}|q_j)$ is equal to one. Some visual terms appear only in a subset of the training samples and their probability values are directly proportional to the number of training instances that contain the corresponding visterms.

Figure 7e) shows the visterm distribution estimated by the Intersection model. The Intersection model simply iterates over all distinct pairs of training examples and intersects the visterm distributions to eliminate visual features which are not particular to the letter bigram class q_j. Once the training instances are assumed to be equally likely, the resulting distribution is simply the average of the visterm distributions of each intersection. As seen in Figure 7 e), the Intersection model successfully eliminates the visual terms which are not related to the letter bigram class "th" in most cases. In this particular example, the only visual term which is related to the letter bigram class "th" is the visterm v_{13}. Visual terms v_3 and v_{17} obtained non-zero values since those visual terms appear in more than one example in the training images. More precisely, visual terms v_3 and v_{17} correspond to visual features which represent letter bigrams "he" and "r-" which are both common in the training samples "their" and "another".

It is clear that the selection of training instances plays an important role for estimating the probabilities in the proposed dependence model. Ideally, the training instances contain word images which are distinct from each other. In the best scenario, they should not have any common letter bigram other than the letter bigram being trained. For example, one should not include the training examples "there" and "the" in the same training set since they have four letter bigrams in common including the space character. This is not desirable since these training instances alone do not help distinguish the visual terms specific to the letter bigram class "th". If there is not a sufficient number of training word images for a given letter bigram class, these pre-conditions might be relaxed in the implementation.

Assuming that the training instances are equally likely and independent from each other, the probability estimate

for the Intersection model can be simplified as:

$$P(v_i|q_j) = \binom{f_i}{2} \Big/ \binom{n_j}{2} \quad (21)$$

where f_i is the total number of images containing the visterm v_i and letter bigram q_j, and, n_j corresponds to the total number of training images containing letter bigram q_j. This implies that there is no need to intersect the visterm distributions for all pairs of training images explicitly if the training images are assumed to be independent and identically distributed. The simplified Intersection model has $O(n)$ time complexity, the same as the Union model, since the frequency values f_i can be computed by iterating over the set of training images at once.

One last problem is that there may not be enough training instances to train visterm distributions for some letter bigram classes. It is desirable to estimate those probabilities using a smoothing technique. More specifically, the estimated probabilities are first normalized,

$$\Pr(v_i|q_j) = \frac{P(v_i|q_j)}{\sum_i P(v_i|q_j)} \quad (22)$$

$$\Pr(v_i) = \sum_j P(v_i|q_j)P(q_j) \quad (23)$$

and smoothed,

$$\tilde{\Pr}(v_i|q_j) = \lambda_S \Pr(v_i|q_j) + (1-\lambda_S)\Pr(v_i) \quad (24)$$

where λ_S is a parameter whose range is [0,1] and $\tilde{\Pr}(v_i|q_j)$ denotes smoothed probabilities. Query terms often correspond to words which appear rarely in the context such as names and places. It is quite likely that the query terms includes letter bigrams which are also rare in the text. It is not desirable to give a lower weight to the features belonging to rare letter bigrams in probability estimations because of their lower prior probabilities. To avoid this, $P(q_j)$ is assumed to be uniformly distributed in Equation 23.

4.3 Indexing letter bigrams

The final MRF score $P_\Lambda(I|Q)$ uses the likelihood of the query letter bigrams and their respective positions in the corresponding word image. For computational efficiency, our approach is to calculate those values only once for all letter bigrams along with their positions in all test images and use these values for resolving the queries instantly. Given a text query, the letter bigram likelihood values are simply looked up for calculating the final MRF score for each test image. Query resolution takes 5 ms for searching an entire book.

4.4 Fusing word image rankings

In this section we propose a general framework which combines the image based match scores with the noisy OCR text search scores to improve retrieval results. Ukkonen's well-known q-gram distance measure [34] is adopted for this purpose. It has been previously shown to be effective for searching OCR degraded texts [35]. In a nut-shell, the q-gram distance approach uses the letter bigrams to represent the input strings in the vector space. The distance between the two words are defined by the Manhattan distance between the q-gram vectors. The resulting score is a discrete number with a range $[0, n+m]$, where n and m are the number of letter bigrams in the input and test words, respectively. In our case, each word bounding box is associated with its OCR text output. The OCR output is used to rank all the word images according to the q-gram distance score to the query word. In this work, the "normalized q-gram similarity score" is introduced:

$$S_q(x,y) = 1 - \frac{D_q(x,y)}{|x|+|y|} \quad (25)$$

where $|x|$ and $|y|$ corresponds to the total number of letter bigrams in the two input words respectively. The normalized q-gram similarity score has a range $[0,1]$ and it produces a higher score if the two words are similar. It is equal to one if the input words are identical. The normalized q-gram similarity score is linearly combined with the image search score:

$$Comb(I,Q) = \lambda_k \cdot P_\Lambda(I|Q) + (1-\lambda_k) \cdot S_q(I_{OCR}, Q) \quad (26)$$

where Q is the text query, I_{OCR} corresponds to the OCR output for the word image I and λ_k is a parameter in the range $[0,1]$ determined using the training and used for combining the normalized q-gram similarity and the word image relevance scores. The word images are finally sorted in descending order of their combined scores $Comb(I,Q)$.

5 Experiments

The aim is to investigate the effectiveness of our image search engine given a particular text query. In order to make the evaluation more fair, we only focus on single-word search. The OCR text search baseline is also case-sensitive. Punctuation is ignored at all stages. For simplification purposes, we do not employ any advanced query evaluation techniques for both text and image search, such as query expansion, stemming etc.

The proposed approach is evaluated using printed books written in three different languages and scripts: English (Latin script), Telugu (an Indian language) and Ottoman (a mixture of Arabic, Persian and Turkish). English includes a fixed set of characters which do not change their shape based on their context in the text and is, therefore, relatively easy to recognize and there are several high accuracy commercial OCR engines available for this purpose. Telugu and Ottoman have no commercial OCR engine available due to their complexities as discussed in the following subsection. For English we demonstrate that OCR text search accuracy can be significantly improved if it is combined with the proposed word image search based approach. For Telugu and Ottoman we show that our approach is effective in searching text in noisy document images. Detailed information about the datasets, training the proposed model and evaluations are given in the following subsections.

5.1 Datasets

5.1.1 The English (Latin Script) Dataset

The English experiments consist of two publicly available books printed in Latin script."Adventures of Sherlock

TABLE 1
Frequency distribution of the words in the English, Telugu and Ottoman books after ignoring punctuation.

Dataset	Total# words	Voc. size	#words per frequency 1	2	3	≥ 4
LAT-S.H.	103375	9080	4552	1408	713	2407
LAT-W.H.	119275	10530	5025	1701	904	2900
TEL-1716	21142	12752	10556	1248	356	592
TEL-1718	4294	2951	2812	89	14	36
OTTO-1	9879	4997	3785	653	205	354
OTTO-2	3548	2498	2064	275	88	71

Holmes" by Arthur Conan Doyle is used for training the hyper parameters of the proposed model. The test book (299 pages) is titled "Wuthering Heights" by Emily Brontë. The ground truth is automatically generated for the 286 pages by aligning the main text of the book (downloaded from the Project Gutenberg website) with the Internet Archive's provided OCR text output using the Recursive Text Alignment Scheme (RETAS) [36]. The text alignment output itself is used for estimating OCR word and character accuracies (88.67% and 97.01%, respectively) as described in [36]. The last 50 pages of the book (89.35% OCR word acc.) are used for testing purposes and the rest used for training. The query test set contains all 3898 vocabulary words which appear in the test pages. Detailed statistics are given in Table 1 after removing the pages which could not be automatically auto-annotated because of missing or duplicate pages. Roughly half of the words in the vocabulary of each book appear only once in the context.

Latin characters typically have straight ink pieces and/or round curves. The corner detector locates relatively fewer number of corner points. A dense sampling approach is therefore adopted to address the sparse keypoint problem. The page images (12 megapixels) are first downsampled by a scaling factor (0.27). Provided OCR word bounding boxes are used for evaluation. No other page segmentation method is used. All the ink (foreground) pixels in the downscaled image are defined as keypoints. and features are extracted at each of these keypoints (see Section 3).

5.1.2 The Telugu Script Dataset

Telugu is a widely spoken language in India (>75 million speakers) and it has its own script. The primary complexity of the Telugu script is the spatial distribution of the connected components that make up the characters. Although the individual characters are lined up from left to right, the connected components of a particular character might be positioned not only in the horizontal order, but also they might be above, below or even inside other connected components. Another complexity is that a word in Telugu might have a slightly different appearance in different contexts although the semantics of the word is the same. Due to the complexities of this script, the character recognition accuracies are typically quite low [37], [38]. No commercial OCR engine is available for recognizing Telugu.

The Telugu experiments consist of two books Telugu-1716 and Telugu-1718 used for training and testing the proposed model respectively [27] (see Table 1). Notice that the majority of the words in these books (82.7% and 95.3% of the vocabulary words, respectively) appear only once in their respective context making conventional word spotting approaches not applicable for searching text in these books. Traditional word spotting approaches need at least one word image example to search for other instances of the query word using visual similarities.

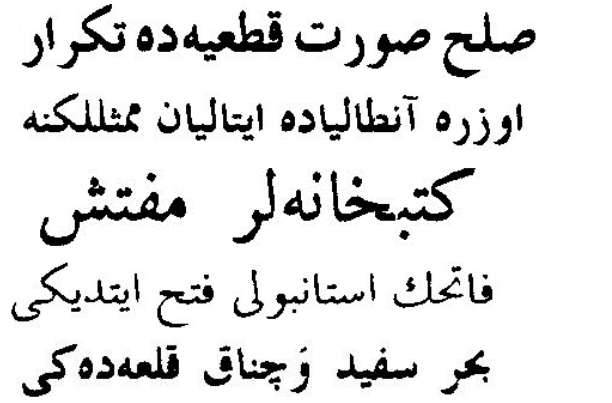


Fig. 8. Example text lines from the Ottoman dataset. The Ottoman script is quite similar to the Arabic script with some additional letters and missing diacritics.

These books were annotated manually using an ASCII coding scheme (lower case Latin characters). Each character is encoded by at least one but typically multiple ASCII characters. Therefore the mapping between each character glyph and the ASCII characters are not one to one. This type of annotation is actually not desirable for training the proposed model. It violates the assumption of one-to-one mapping between each character class and their corresponding visuals in the word images. However, the experiments demonstrate that the proposed model tolerates one-to-many and many-to-one character mappings as well. Figure 11 shows some example word images written in Telugu along with their ASCII encodings. All the words in the vocabulary of the test book are used for evaluation purposes.

5.1.3 The Ottoman Script Dataset

The Ottoman script is quite similar to the Arabic script with some additional characters and missing diacritics. An existing Ottoman dataset is used for evaluation purposes [39]. It consists of 100 document images (300 dpi - binary images) scanned from two different books teaching Ottoman script. Each book contains a number of short readings written in Ottoman language. Each reading published in these books is originally scanned from different sources and the font type and/or the size of text therefore varies for each article. The articles scanned from the first book contain 60 document images and they are used for training the proposed dependence model. The remaining 40 pages scanned from the second book are used for testing purposes. Table 1 shows word frequency statistics for both sets.

The training and test sets (OTTO-1 and OTTO-2) consist of 9879 and 3548 word images respectively. As for Telugu most words appear only once in the respective context for both sets 75.7% and 82.6% of the words in the vocabulary appear only once in the training and test sets respectively. Conventional word spotting techniques which require whole word images are therefore not applicable for searching text in these collections as well.

The ground truth contains locations of each connected component in the document images along with the associated letter symbols. There are 48 integer coded ink shapes used for annotating the connected components. The entire dataset contains a total of 70K shape-coded characters. Line and word boundaries are used for word annotations and they are determined using projection profiles. The best recognition accuracy reported for this Ottoman dataset is 93% [39]. This dataset is the most challenging one among the others because the font type, size and document noise vary across different articles (Figure 8).

5.2 Training

First, the visual vocabulary is trained by selecting a number of document images at random and clustering their visual features using the Hierarchical K-means algorithm (branching factor 64, depth 2), as described in Section 3. Next, the visual term distributions of the letter bigram classes are estimated. Including the space character, there are $63 \times 63 = 3969$, $49 \times 49 = 2401$ and $27 \times 27 = 729$ letter (or character primitive) bigram classes are identified for English, Ottoman and Telugu datasets, respectively.

One way to estimate the prior $P(v_i)$ and posterior $P(v_i|q_j)$ probabilities is to use scanned page images with annotated word bounding boxes. However, it is not possible to estimate visterm distributions for all letter bigram classes. Only about 300 of the 4K letter bigram classes have more than 20 occurrences in a single book. These letter bigrams are actually sufficient to generate the majority of words in the English language. In most cases only a few letter bigrams might be sufficient to resolve a given query. For example, for a query word "Holmes", the letter bigrams (space,H) and (s,space) are sufficient to filter out the majority of the words in the vocabulary of the whole book for being a match. However, the more letter bigrams used for search, the more precise the retrieval.

Another option is to estimate visual distributions using synthetic word images. However, learning image features from synthetic images has its own challenges. It is not easy to model document noise and different fonts [40]. Experiments with synthetic word images using 100 different fonts performed much worse (MAP 0.47 at best) for English and are not discussed further.

The hyper parameters of the MRF model (such as λ_M, λ_S and λ_k) are learned using grid search. In the case of English, they are learned from another book (Sherlock Holmes, S.H.) and applied on Wuthering Heights (W. H.). The last 50 pages of W.H. are used for testing and the visual terms and their distributions are trained on the rest. The training process for English is fully automatic - using OCR output as labels. In the case of Telugu and Ottoman there is no OCR output available. In these cases, the annotated book is necessary for training the model parameters, visual vocabulary and visterm distributions. OTTO-1 and TELUGU-1716 books are used for training all the parameters of the proposed model. It should be noted that, the proposed approach can search for arbitrary text in document images unlike the word spotting approaches. Therefore all the words in the vocabulary of the test set are used for evaluation purposes.

The overall effectiveness of the proposed approach is not quite sensitive to the Gaussian parameter σ and therefore it is set to 0.5 in all experiments. Yet another parameter is the sliding interval for the Gaussian window. Smaller intervals yield better results however processing time may get very large. The point is to ensure that we do not skip over any letter while sliding the window. Therefore this value is set to 0.5 times the height of the word bounding box. Assuming that all the letters have equal width, this corresponds to $2n + 1$ window positions for n letters in a word image.

5.3 Baselines

Ukkonen's q-gram based distance approach described in Section 4.4 serves as a natural baseline for Latin script experiments since the OCR output is available. The other baseline is a multi-layer Bidirectional Long Short-term Memory (BLSTM) network trained with Connectionist Temporal Classification (CTC) as described in [23], [24]. The BLSTM baseline produces a text string for the given word images. The $\text{BLSTM}_{\text{q-gram}}$ baseline uses q-gram similarity scoring function for ranking word images. The Latin experiments have also a third baseline ($\text{BLSTM}_{\text{q-gram}}$+$\text{OCR}_{\text{q-gram}}$), which uses BLSTM and OCR text outputs to generate the final ranked list as described in Section 4.4.

The open source TensorFlow [41] implementation is used for training BLSTM models. The input images are first height normalized (32 pixels) and randomly grouped into batches of 200 images. Three columns of raw gray-scale pixel values (32x3=96 features) are fed into the network at each time step. Using a single or two columns of pixel values performed worse. The momentum and learning rate parameters are set to be 0.9 and 0.001, respectively. After each training epoch, the batches are randomly reshuffled and the learning rate is updated using an exponential decay function with parameter 0.9.

A number of experiments had to be performed to find the best network topology and training parameters. Mathew et. al. [42] uses a BLSTM network with three hidden layers of size 50 nodes each to recognize a variety of Indic scripts. Qaralleh et. al. [43] demonstrates that a BLSTM model with three hidden layers provides the best results for recognizing Arabic script. In our case, the number of hidden layers (2, 3 and 4) and the total number nodes in each layer (30, 50 and 70) are varied for English, Ottoman and Telugu scripts independently. The final (best) network uses three layers of BLSTM layers with 50 units each for all scripts. After 100 training epochs, the best word recognition error rates and corresponding epoch numbers are (7.1%,65), (84.9%,21) and (83.8%,50) for English, Telugu and Ottoman, respectively.

It should be noted that there are a large number of shape occurrences to be learned for Telugu and Ottoman scripts as compared to Latin. The majority of the input words and shape occurrences appear only once in the entire training and test sets. Each connected component or character i) may have varying size, ii) might be connected and/or positioned on top of each other and iii) hand coded with an arbitrary number of ASCII characters. Much larger manually labeled datasets are therefore desirable for training better models. In this work, Telugu, Ottoman and Latin training sets contain 21K, 10K and 100K word images, respectively. Trained BLSTM models performed quite poorly on the Telugu and Ottoman test sets and are not discussed further.

TABLE 2
MAP scores for the test book titled "Wuthering Heights". "MRF" stands for the proposed scoring function. "I" and "U" are short for "Intersection" and "Union" respectively.

Search method	Learning Model	λ_M	λ_S	λ_k	MAP
MRF + $OCR_{q\text{-}gram}$	I	0.19	0.01	0.50	**0.958**
MRF + $OCR_{q\text{-}gram}$	U	0.53	0.01	0.47	0.957
$BLSTM_{q\text{-}gram}$ + $OCR_{q\text{-}gram}$	-	-	-	0.50	0.952
$OCR_{q\text{-}gram}$	-	-	-	-	0.930
$BLSTM_{q\text{-}gram}$	-	-	-	-	0.866
MRF	I	0.19	0.01	-	0.854
MRF	U	0.53	0.01	-	0.817
MRF	I	1.0	0.01	-	0.792
MRF	U	1.0	0.01	-	0.497

5.4 Evaluation

The evaluations are performed over all vocabulary words that appear in the test collection. The query words with one or two characters are excluded from the query test set. These words typically correspond to function or stop words (for ex. "in", "at") and are usually ignored for evaluation purposes as they are unlikely to appear as query words. The evaluation measure is "Mean Average Precision" (MAP) which is computed over the entire ranked list.

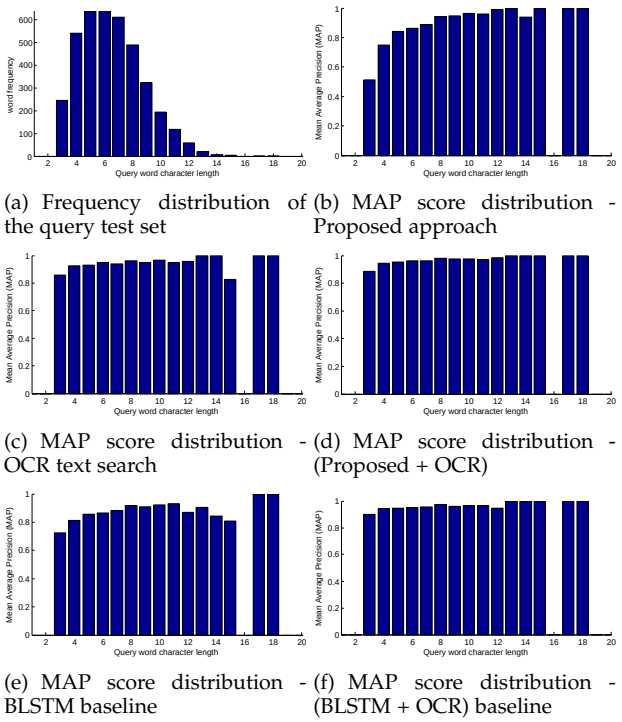
Image search treats each visually distinct pattern as a different visual class and performs the search accordingly. The same character may have visually different forms in another context (such as 'A' and 'a'). In Arabic, characters might have up to four different shapes based on their position in the word. For simplification purposes, the search task is therefore assumed to be case-sensitive. Advanced query evaluation techniques may also be used to retrieve different forms and/or inflections of the query word. They are not discussed further due to space limitations.

5.4.1 English (Latin Script) Experiments

The ABBYY FineReader OCR text output and corresponding word bounding boxes of the training book "Sherlock Holmes" are used for training the model's hyper parameters λ_M, λ_S and λ_k. Two hundred query words are randomly selected from the vocabulary of the training book to determine the parameters which maximize MAP score. Estimated model parameters are later used for the test book.

The first 236 pages of the test book "Wuthering heights" are used for learning the visual vocabulary and visterm distributions for each letter bigram. The noisy OCR output (ABBYY) of the test book is used as the ground truth for this purpose. Estimated OCR letter bigram accuracy is 94.09% for the test book. All the 3838 words that appear at least once in the last 50 pages of the book are used for querying.

Table 2 shows the retrieval scores of the proposed and baseline approaches for the test book "Wuthering Heights". The baseline OCR text search baseline gives a MAP score of 0.930 compared to the scores of 0.854 and 0.817 using the Intersection and Union models respectively. When the letter bigram positional dependencies are not used ($\lambda_M = 1.0$) the scores are much lower for both the Intersection and Union models showing that bigram order matters. Combining the OCR text search baseline with the proposed dependence model provides the highest retrieval scores - 0.958 and 0.957



(a) Frequency distribution of the query test set
(b) MAP score distribution - Proposed approach
(c) MAP score distribution - OCR text search
(d) MAP score distribution - (Proposed + OCR)
(e) MAP score distribution - BLSTM baseline
(f) MAP score distribution - (BLSTM + OCR) baseline

Fig. 9. Distribution of query words as a function of their length is given in a). MAP score distribution as a function of the query word length is given for b) the proposed approach (Intersection model), c) OCR text search baseline, d) Proposed approach + OCR, e) BLSTM baseline, and, f) (BLSTM+OCR) baselines.

respectively versus the OCR baseline of 0.93. Both the OCR text search baseline and the combined approach provide the same AP scores for 3220 out of 3898 test query words in total. The combined approach provides better AP results for 573 queries out of the remaining 678 queries.

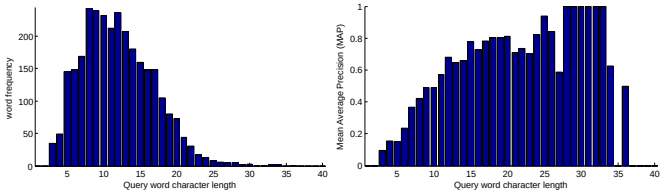
A BLSTM baseline trained with the ground truth text provides a MAP score of 0.866. Without combining with OCR text search, BLSTM baseline performs slightly better than the proposed approach (0.854 MAP), which is trained with noisy OCR output itself. When combined with OCR text search, the proposed approach outperforms the BLSTM baseline and provides the best results (MAP 0.958). Further investigation shows that the BLSTM baseline tends to combine repeating characters ("room" maps to "rom") or misrecognize larger individual characters with multiple letters ("m" maps to "in" or "hn").

In the case of English, these results demonstrate that the noisy OCR text output can be effectively used for improving OCR text search. Given the hyper-parameters, the training phase is fully automated. The visual term distributions for each letter bigram are trained from the target book itself. It automatically captures the font characteristics and adapts to the specifics document noise inherent in the book.

Figure 9 shows that MAP scores increase as the query gets longer. This effect is more evident for the image search approach. The lowest MAP scores are obtained for query words which include only three letters. In the case of image search, the query word "the" is commonly confused with "there" and "therefore" since these words include all the letter bigrams of the query word exactly in the same order. The proposed approach only accounts for the existence and

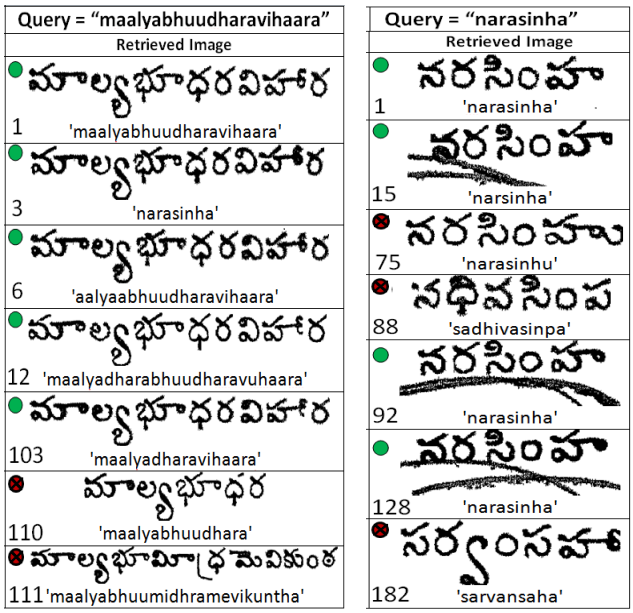
TABLE 3
Experimental results for the Telugu dataset (TELUGU-1716 for training, TELUGU-1718 for testing).

Training Model	Dataset	λ_S	λ_M	MAP
Intersection	Training	0.002	0.46	0.563
	Test	0.002	0.46	**0.562**
Union	Training	0.016	0.65	0.488
	Test	0.016	0.65	0.436



(a) Word frequency distribution (b) MAP score distribution

Fig. 10. Intersection model's MAP score distribution on the TELUGU-1718 test set as a function of the query word length.



(a) Long query example (b) Retrieving noisy word images

Fig. 11. Successfully retrieved Telugu word images.

the global order of letter bigrams in the word image. A test image therefore obtains a high matching score if it subsumes the letter bigrams of the query word and the letter bigrams also follow the same order.

5.4.2 *Telugu Script Experiments*

The model parameters, the visual vocabulary and visterm distributions for each letter bigram are trained on the training set (TELUGU-1716) and directly applied on the test set. The MAP scores produced by the proposed approach using the Union and Intersection training models are shown for the training and test sets in Table 3. For both test sets the intersection model does much better than the union model.

In the case of the TELUGU-1718 test set, the relevant word images are typically found at the top positions in the rank lists. TOP-K recall rates for the first correct match are 0.46, 0.602, 0.663, 0.728, 0.789, 0.857 when K is set to 1, 3, 5, 10, 20 and 50, respectively. The rank lists contain 4294 word images in total and 95% of the query words appear only once in the test book TELUGU-1718 (Table 1). It should be noted that a significant amount of annotation errors exist in the provided ground truth of the Telugu books as discussed later. The MAP scores are expected to be higher than 0.56.

Unlike OCR and conventional word spotting approaches, the proposed approach is able to find long and rare query terms effectively without requiring any dictionary or explicit language model. Figure 10 shows word frequency and MAP scores as a function of word length.

Figure 11 shows results for two example queries (query at top position). The retrieved word images are listed according to their ranks and the ground truth label is given under each word image. Only a subset of the retrieved (both relevant and non-relevant) words are shown due to space considerations. Green (light) and red (dark) circles indicate relevant and non-relevant word images respectively.

The first example query word is "maalyabhuudharavihaara" (Figure 11a)). All 109 examples of the query word are retrieved at top ranks without any false positives. The top five correctly matched word images are identical but their (Romanized) groundtruth are sometimes not the same. Inconsistent annotation errors are estimated to have impacted 10% of the words in the dataset. Using the provided ground truth labels, the AP score for this query is 0.756 while manual evaluation of the ranked list yields an AP score of 1.0. Despite the noisy annotations, the proposed model effectively learns the dependencies between the visual terms and character letter bigrams.

The second example is a medium length query word: "narasinha", for which there are noisy matching instances (note the lines through several of the images). According to the ground truth, there are 113 samples of the query word and the AP score is 0.774. Manual investigation indicates that there are actually 120 examples of the query word. The true AP score is 0.85. Notice that the noisy examples of the query word are successfully retrieved by the proposed approach. Commercial OCR systems typically fail to recognize character glyphs which are underlined or connected to other characters. The false positive examples (word images with rank 75 and 88) have a number of common characters with the original query word. These words obtained a high rank since they are visually quite similar to the query word.

Robustness to font variations and document noise might co-exist if the word bounding box estimation is successful within the scale estimation error tolerance of the descriptor (in our case SIFT). Fig 11b) shows examples where patch scale estimation using bounding box height (as explored for Ottoman script) would not help match the noisy samples. In the case of Telugu, setting the patch scale according to the known font size resolved the problem. Another option is to estimate the font size and used it for each word image and it is not explored in this work.

The experiments demonstrate the effectiveness of the proposed approach for searching noisy Telugu documents for which OCR and word spotting techniques are not applicable. The query response time is 4 ms/query.

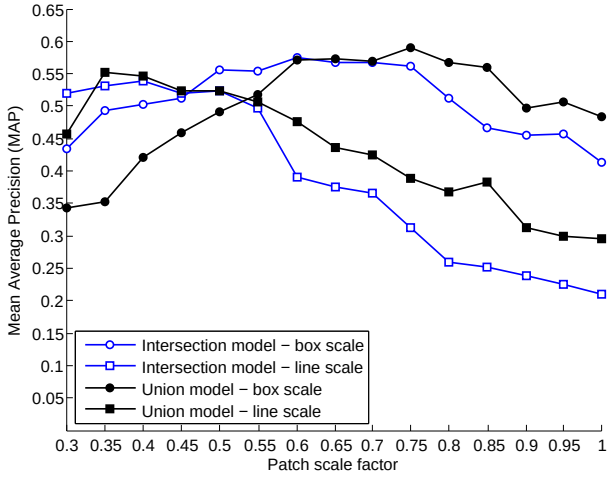


Fig. 12. MAP scores as a function of the patch scale factor for different configurations of the proposed model on the Ottoman training set.

5.4.3 Ottoman Script Experiments

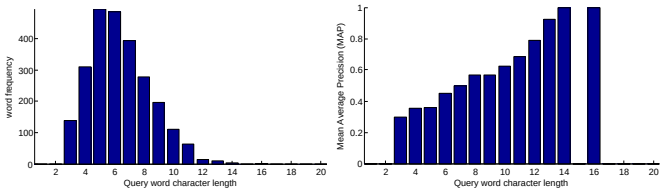
The Telugu and English (Latin script) experiments used printed documents in almost entirely the same font type and size. In these cases, fixing the size and the orientation of the image patch across the document images is sufficient to find visual correspondences across the word images. However, this is not the case for the Ottoman dataset. It includes several articles scanned from several sources and the font varies across document images. The visual features extracted from the same word images printed in different font type and size do not match especially if the patch size is fixed. Three different approaches are investigated to address this problem. The first approach is to use the scale-invariant SIFT keypoint detector which automatically determines the coordinate, scale and orientation of each keypoint. The scale-invariant nature of SIFT helps match visual features across different scales. However, SIFT features are known to be sensitive to certain types of document noise such as text underlining and ink bleeding [27]. The second approach is to use the fast-corner-detector to find the location of the visual terms. Each word image is assigned a uniform image patch scale relative to the height of the line it belongs to. The patch scale of a word image is its line height multiplied with a constant called "patch scale factor". The third approach is similar to the previous one except that the height of the word bounding box is used for determining the patch scale. The second and third approaches assume that the page skew is corrected and the image patch orientation is set to zero.

The effectiveness of line and box height based patch scale estimation approaches are tested on the OTTO-1 training set by varying the patch scale factor. The model parameters λ_S and λ_M are trained using the visual features extracted by the scale-invariant SIFT keypoint detector. The same model parameters are used for the other settings. The experiments are repeated for both union and intersection learning models (Figure 12). On the training set, patch size estimation using the box height gives the best results for both learning models. The union model provides a slightly higher MAP score on the training set with a patch scale factor 0.75.

The best patch scale factor is determined for each configuration on the training set using the MAP scores plotted in Figure 12. Estimated patch scale factors are then applied to the test set for the corresponding configuration and the obtained MAP scores are given in Table 4. The MRF model parameters are estimated using the scale-invariant SIFT features and the same parameters are used for testing the configurations as well. The best test configuration uses the box scale approach coupled with the intersection model and provides a MAP score of 0.473 followed by using the intersection model with the line scale approach (MAP 0.392). Note that the union model does better on the training set but the intersection model does better on the test set.

TABLE 4
Experimental results for the Ottoman dataset (training on OTTO-1, test on OTTO-2) for three different patch size selection approaches.

Patch Scale	Training Model	Dataset	Scale factor	λ_S	λ_M	MAP
Box	Intersec.	Training	0.60	0.002	0.91	0.574
		Test	0.60	0.002	0.91	**0.473**
	Union	Training	0.75	0.005	0.99	0.590
		Test	0.75	0.005	0.99	0.365
Line	Intersec.	Training	0.40	0.002	0.91	0.539
		Test	0.40	0.002	0.91	0.392
	Union	Training	0.35	0.005	0.99	0.552
		Test	0.35	0.005	0.99	0.275
SIFT	Intersec.	Training	-	0.002	0.91	0.579
		Test	-	0.002	0.91	0.385
	Union	Training	-	0.005	0.99	0.604
		Test	-	0.005	0.99	0.363



(a) Word frequency distribution (b) MAP score distribution

Fig. 13. MAP score distribution of the best configuration on the OTTO-2 test set as a function of the query word length.

The results in Table 4 indicate that the best values λ_S is small as in the case of the Telugu experiments. This implies that smoothing is not of much help even though the number of training instances is not large (<10K labeled word images). The estimated values for the λ_M are 0.99 and 0.91 for the union and intersection models respectively. This indicates that the existence of letter bigrams is more important than their relative positions. Further analysis indicate that some of the basic shapes used for annotating the script are quite small. Some of them simply correspond to simple loops and pieces of ink which might be a part of many different characters in the alphabet. The width of a shape might be as small as 5-10% of the line height. It should be noted the sliding interval for the Gaussian window is set to 0.5 times the bounding box height in all experiments for simplification purposes. Estimation of the exact position of shape code pairs is therefore not quite reliable compared to Telugu and English. Smaller values for the sliding window interval are expected to give better localization performance with additional computational cost.

Figure 13 shows the distribution of MAP scores as a

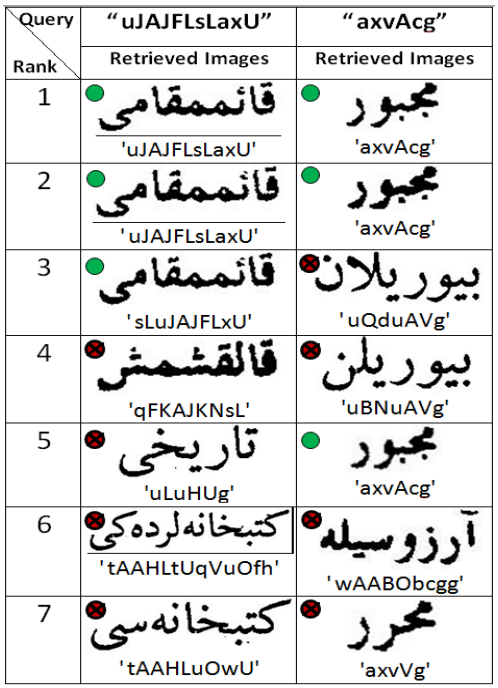


Fig. 14. Two example queries on the OTTO-2 test set.

function of word length. Average word length in the Ottoman dataset is not as high as in the Telugu dataset. It is clear that the MAP scores increase as the total number of characters increase in the query word. As discussed in the case of Telugu, it is desirable to have high accuracy for more complex queries which are typically longer words.

As in the case of Telugu, the relevant word images are typically found at the top positions in the rank lists. TOP-K recall for the first correct match are 0.20, 0.356, 0.398, 0.528, 0.624, 0.776 for values of K 1, 3, 5, 10, 20 and 50, respectively. The ranked lists contain 3548 word images in total. 82.6% of the words in the OTTO-2 query test set appear only once. 97.2% of the vocabulary words appear no more than three times in the entire test collection.

Figure 14 shows a long (left) and a short (right) query example for the Ottoman test set. The long query word has three matching words in the ground truth. The proposed approach retrieved all of them at the top of the ranked list with an AP score of 1.0. As in the case of Telugu experiments, the proposed approach was able to find annotation errors in the dataset. The word image ranked 3rd is retrieved correctly by the proposed approach but it has an incorrect label. The 4th retrieved image is not a match but it is visually quite similar to the true positive examples. The short query example has an AP score of 0.867 with three relevant word images. The first two true positives are top ranked, however, the last one is ranked 5th after two false positives. Notice the word images in rows 3 and 4 have partial visual similarity with the query word - the first four letters from the right partially match the shapes and characters of the query word. Partial matches might be useful for users especially if the search term does not appear in the test set. Partially matching words are typically inflections or morphological variations of the query word and they might also be considered to be relevant depending on the search task.

5.4.4 Computational analysis

The off-line processing has three main stages: i) visual feature extraction, ii) learning visual term distributions for each letter bigram, and, iii) creating a letter bigram index for resolving queries. In the case of Latin script, extracting visual features takes 30 sec/page using a single thread MATLAB CPU implementation on a i5 Intel CPU machine with 8GB RAM (with GPU, 1-2 sec/page, less than 10 min for 300 page book). Learning visual term distributions takes less than 10 seconds for 100K training word images. Creating the letter bigram index takes about one hour. Each stage is inherently parallelizable and further speed improvements are possible. The online query resolution takes about 5 ms for searching an entire book. On the same dataset, the BLSTM baseline took about a day to train using TensorFlow implementation (Hardware: i7 Intel CPU, 96GB RAM, 12GB GTX Titan X GPU). The BLSTM baseline also required an explicit network construction and parameter tuning. It took several weeks of computation to find the right parameters and produce the best results.

6 Conclusion

A dependence model is proposed for searching arbitrary text in noisy document images for which high quality OCR does not exist. The proposed approach relies on efficient processing of local visual features to robustly match word images in real time. The effectiveness of the proposed approach is demonstrated by improving high quality OCR text search in the case of English. Experiments on Ottoman and Telugu documents demonstrate that the proposed framework is effective in searching arbitrary text in noisy document images for which OCR is not applicable or available. The proposed dependence model outperforms a state-of-the-art BLSTM solution and it does not need to be retrained for each book separately.

Future work includes (i) speeding up the offline processing stage using more efficient and robust visual features, (ii) improving the retrieval model by incorporating additional features and dependencies, (iii) investigating the effects of document noise in the retrieval accuracy, (iv) investigating the use of synthetic images to accurately train the dependencies for the letter bigrams, (v) extending the proposed approach for searching arbitrary texts in handwritten documents and natural scene images, and, (vi) generalizing the proposed approach for searching text large scanned book collections using a collection-wide letter bigram index with clique features.

Acknowledgments

Thanks to i) Pramod Sankar & C. V. Jawahar for providing us the Telugu datasets and ii) Bilkent University Multimedia Databases Group for the Ottoman datasets. Special thanks to W. Bruce Croft for his valuable discussions and feedback. This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #IIS-0910884. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

REFERENCES

[1] K. Kukich, "Technique for automatically correcting words in text," *ACM Computing Surveys*, vol. 24, no. 4, pp. 377–439, Dec. 1992.

[2] S. M. Beitzel, E. C. Jensen, and D. A. Grossman, "Retrieving OCR text: A survey of current approaches," in *SDUIT*, 2003.

[3] X. Tong and D. A. Evans, "A statistical approach to automatic OCR error correction in context," in *WVLC*, 1996, pp. 88–100.

[4] I. Z. Yalniz, I. S. Altingövde, U. Güdükbay, and Ö. Ulusoy, "Ottoman archives explorer: A retrieval system for digital Ottoman archives," *JOCCH*, vol. 2, no. 3, p. 8, 2009.

[5] S. Harding, W. B. Croft, and C. Weir, "Probabilistic retrieval of OCR degraded text using n-grams," in *ECDL*, 1997, pp. 345–359.

[6] J. Parapar, A. Freire, and A. Barreiro, "Revisiting n-gram based models for retrieval in degraded large collections," in *ECIR*, 2009, pp. 680–684.

[7] "The Internet Archive," http://www.archive.org, 2012.

[8] T. M. Rath and R. Manmatha, "Word spotting for historical documents," *IJDAR*, vol. 9, no. (2-4), pp. 139–152, 2007.

[9] E. Ataer and P. Duygulu, "Matching Ottoman words: an image retrieval approach to historical document indexing," in *CIVR*, 2007, pp. 341–347.

[10] T. Rath and R. Manmatha, "Word image matching using dynamic time warping," in *CVPR*, vol. 2, 2003, pp. 521–527.

[11] T.Rath, R. Manmatha, and V. Lavrenko., "A search engine for historical manuscript images," in *SIGIR, pp. 297 -304*, 2004.

[12] T. M. Rath and R. Manmatha, "Features for word spotting in historical manuscripts," in *ICDAR*, 2003, pp. 218–222.

[13] K. P. Sankar and C. V. Jawahar, "Probabilistic reverse annotation for large scale image retrieval," in *CVPR*, 2007.

[14] P. Sankar K., C. V. Jawahar, and R. Manmatha, "Nearest neighbor based collection OCR," in *DAS*, 2010, pp. 207–214.

[15] S. Wshah, G. Kumar, and V. Govindaraju, "Multilingual word spotting in offline handwritten documents," in *ICPR*, 2012, pp. 310–313.

[16] G. R. Ball, S. N. Srihari, and H. Srinivasan, "Segmentation-based and segmentation-free methods for spotting handwritten arabic words," in *ICFHR*, Oct. 2006.

[17] J. Almazan, A. Gordo, A. Fornes, and E. Valveny, "Handwritten word spotting with corrected attributes," in *ICCV*, 2013.

[18] T. M. Rath and R. Manmatha, "Features for word spotting in historical manuscripts," in *ICDAR*, 2003, pp. 218–222.

[19] R. Manmatha and J. L. Rothfeder, "A scale space approach for automatically segmenting words from historical handwritten documents," *IEEE Trans. on PAMI*, vol. 27, no. 8, pp. 1212–1225, 2005.

[20] G. Louloudis, B. Gatos, and C. Halatsis, "Text line detection in unconstrained handwritten documents using a block-based hough transform approach," in *ICDAR '07*, 2007, pp. 599–603.

[21] U.-V. Marti and H. Bunke, "Text line segmentation and word recognition in a system for general writer independent handwriting recognition," in *ICDAR*, 2001, pp. 159–163.

[22] J. A. Rodriguez and F. Perronnin, "Local gradient histogram features for word spotting in unconstrained handwritten documents," in *Proc. ICFHR*, 2008, pp. 7–12.

[23] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE PAMI*, vol. 31, pp. 855–868, 2009.

[24] V. Frinken, A. Fischer, H. Bunke, and R. Manmatha, "Adapting BLSTM neural network based keyword spotting trained on modern data to historical documents." in *ICFHR*, 2010, pp. 352–357.

[25] S. Marinai, E. Marino, and G. Soda, "A comparison of clustering methods for word image indexing," in *DAS*, 2008, pp. 671–676.

[26] S. Lu, L. Li, and C. Tan, "Document image retrieval through word shape coding," *IEEE PAMI*, vol. 30, pp. 1913–1918, 2008.

[27] I. Z. Yalniz and R. Manmatha, "An efficient framework for searching text in noisy document images," in *DAS*, 2012, pp. 48–52.

[28] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *ECCV*, vol. 1, May 2006, pp. 430–443.

[29] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, pp. 91–110, November 2004.

[30] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proceedings of CVPR*, 2006, pp. 2161–2168.

[31] D. Metzler and W. B. Croft, "A markov random field model for term dependencies," in *SIGIR*, 2005, pp. 472–479.

[32] S. Feng and R. Manmatha, "A discrete direct retrieval model for image and video retrieval," in *CIVR*, 2008, pp. 427–436.

[33] S. Feng, R. Manmatha, and V. Lavrenko, "Multiple bernoulli relevance models for image and video annotation." in *CVPR*, 2004, pp. 1002–1009.

[34] E. Ukkonen, "Approximate string-matching with q-grams and maximal matches," *Theor. Comput. Sci.*, vol. 92, pp. 191–211, 1992.

[35] S. Harding, W. B. Croft, and C. Weir, "Probabilistic retrieval of OCR degraded text using n-grams," in *ECDL*, 1997, pp. 345–359.

[36] I. Z. Yalniz and R. Manmatha, "A fast alignment scheme for automatic OCR evaluation of books," in *ICDAR*, 2011, pp. 754–758.

[37] U. Pal and B. Chaudhuri, "Indian script character recognition: a survey," *Pattern Recognition*, vol. 37, no. 9, pp. 1887 – 1899, 2004.

[38] V. Govindaraju and S. Setlur, *Guide to OCR for Indic Scripts - Document Recognition and Retrieval*, 1st ed. Springer, 2009.

[39] I. Z. Yalniz, I. S. Altingovde, U. Gudukbay, and O. Ulusoy, "Integrated segmentation and recognition of connected Ottoman script," *Optical Engineering*, vol. 48, no. 11, pp. 117 205–12, 2009.

[40] T. Konidaris et. al., "Keyword-guided word spotting in historical printed documents using synthetic data and user feedback," *IJDAR*, vol. 9, no. 2-4, pp. 167–177, 2007.

[41] M. Abadi and et. al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: http://tensorflow.org/

[42] M. Mathew, A. K. Singh, and C. V. Jawahar, "Multilingual OCR for indic scripts," in *DAS*, 2016, pp. 186–191.

[43] E. Qaralleh, G. Abandah, and F. Jamour, "Tuning recurrent neural networks for recognizing handwritten arabic words," *Journal of Software Engineering and Applications*, vol. 6, no. 10, p. 533, 2013.

I. Zeki Yalniz received his PhD in computer science from University of Massachusetts at Amherst (2013) and B.S. & M.S. degrees in computer engineering from Bilkent University, Ankara, Turkey. He is broadly interested in combining computer vision and information retrieval concepts to offer practical solutions for data and/or computation intensive problems. On the computer vision side, he worked on texture analysis and segmentation, video content analysis, document image analysis, image/video fingerprinting, image matching and recognition. On the information retrieval side, he worked on the retrieval of noisy (OCRed) text documents focusing on text alignment, OCR evaluation and error correction, duplicate document detection, and translation identification. His current research focuses on the development of effective and fast visual search techniques at scale. He is a reviewer for major IEEE and ACM journals and conferences. He has also served as a program committee member for CIKM, ICDAR and OAIR. He is currently a research scientist at Facebook Inc.

R. Manmatha is an adjunct professor at the School of Computer Science, University of Massachusetts at Amherst. His research is in the areas of retrieving handwritten documents, printed documents and image/video retrieval. He proposed the idea of word spotting for handwritten documents. He and his students built the first automatic demonstration system on a portion of George Washington's handwritten documents. In addition he has worked on large collections of scanned printed books including algorithms to automatically compute the error rate of optical character recognition and detecting duplicates and translations in large printed collections. He worked on the automatic annotation and retrieval of images and videos and action and event detection in videos. He was a co-founder of SnapTell, a mobile image search company (acquired by A9/Amazon). At SnapTell he worked on large scale instance matching. He was a consultant to A9/Amazon and Google. He is an associate editor for IEEE PAMI and was previously an associate editor for ACM TOIS and Pattern Recognition Letters. He is a general chair for the Intl. Conf. on Frontiers of Handwriting Recognition (ICFHR) in 2018, and was a program chair for ACM Intl. Conf. for Multimedia Retrieval (ICMR) 2014 and ICFHR 2014 and has been on the program committees of a number of conferences such as ICDAR, ICFHR, DAS, SIGIR, CIKM, CVPR, ECCV and ICCV.