Word Spotting for Historical Documents

T. M. Rath and R. Manmatha*

Multimedia Indexing and Retrieval Group

Center for Intelligent Information Retrieval

Dept. of Computer Science

University of Massachusetts Amherst

Amherst, MA 01003

Abstract

Searching and indexing historical handwritten collections is a very challenging problem. We describe an approach called word spotting which involves grouping word images into clusters of similar words by using image matching to find similarity. By annotating "interesting" clusters, an index that links words to the locations where they occur can be built automatically.

Image similarities computed using a number of different techniques including dynamic time warping are compared. The word similarities are then used for clustering

*[trath,manmatha]@cs.umass.edu

using both K-means and agglomerative clustering techniques. It is shown on a subset of the George Washington collection that such a word spotting technique can outperform an HMM word-based recognition technique in terms of word error rates.

Traditional libraries contain an enormous amount of handwritten historical documents that they would like to make available electronically on the Internet or on digital media. Examples include the collections of Presidents like George Washington at the Library of Congress and scientists like Isaac Newton at the University of Cambridge Library. Such large collections (George Washington's handwritten papers exceed 140,000 pages). can only be accessed efficiently if a searchable or browsable index exists, just like in the back of a book. The current state-of-the-art approach to this task is to manually create an index for the collection. Since manual indexing is expensive, automation is desirable in order to reduce costs.

Automatic handwriting recognition has seen great improvements over the last decades and the error rates have dropped to a level that makes commercial applications feasible. *Offline* handwriting recognition, where only an image of the produced writing is available, has only been successful in domains with very limited vocabularies, such as automatic mail sorting and check processing. In addition, these domains usually provide good quality images, while the quality of historical documents is often significantly degraded due to faded ink, stained paper, and other adverse factors (see Figure 1). Consequently, traditional Optical Character Recognition (OCR) techniques that usually recognize words character-by-character, fail when applied to historical manuscripts.

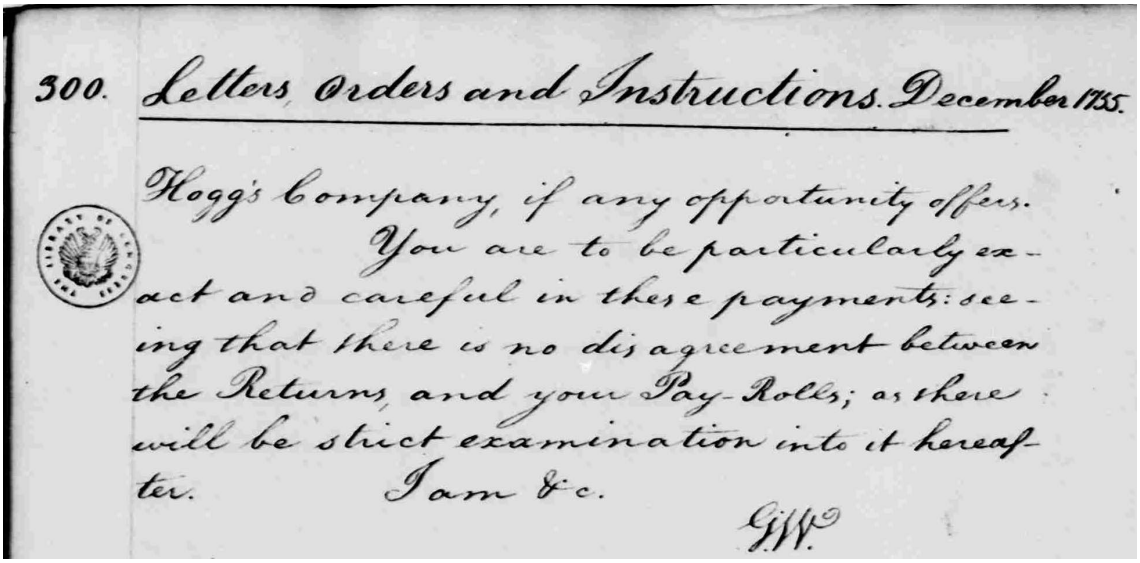


Figure 1: Part of a scanned document from the George Washington collection.

For collections of handwritten manuscripts written by a single author (or a few authors), (for example the George Washington collection used in this paper), the images of multiple instances of the same word are likely to look similar. Wordspotting, which was initially proposed by Manmatha et al. [15], treats a collection of documents as a collection of word images. First the document is segmented into word images. The idea of wordspotting (see Figure 2) is to use image matching for calculating pairwise "distances" between word images, which can be used to cluster all words occuring in a collection of handwritten documents. Ideally, each cluster would contain all the words with a particular annotation. Clusters that contain terms which are "interesting" for an index for the document collection, are selected and labeled manually. By assigning the cluster labels to all word images contained in a cluster, we get a partial transcription of the document collection. This in turn allows us to create a partial index for the collection, which allows us to retrieve text

portions that contain only the manually assigned labels.

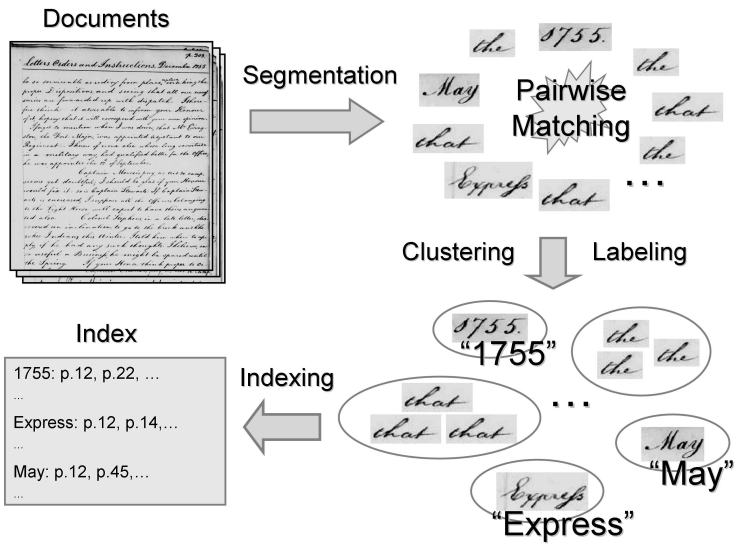


Figure 2: An illustration of the Wordspotting process. Documents are segmented, and distances between word images are calculated. After clustering the word images, some clusters are manually labeled and can be used as index terms.

Historical handwritten documents are often of poor quality and unlike printed documents, there is variation in the way the words are written. The George Washington collection, for which we present results here, was not scanned from the originals, but from microfilms which further decreases the quality. Thus, both segmentation of a page into words and the matching of word images (i.e., finding similarity) are challenging problems for such documents. Previous work by [17, 16] has dealt with the problem of segmenting such images of historical documents. We, therefore, use the output of their algorithm as input to our system. The image matching problem is difficult and has prompted a number of publications that propose algorithms and features for the approach [14, 13, 24, 26, 25, 27].

In this paper, we discuss an approach to matching images using Dynamic Time Warping (DTW) of profiles of the word images. DTW has been widely used to match 1-D signals in

the speech processing, bio-informatics, and also the online handwriting communities. DTW can handle local distortions in word images and is not restricted to a single global transform. DTW is compared with a number of other techniques including XOR, affine-corrected Euclidean Distance Matching, shape context [2], intensity correlation using sum-of-squared differences, an affine matching point matching algorithm due to Scott and Longuet-Higgins [30] and a point correlation voting algorithm [27]. We show that for image matching DTW outperforms all other algorithms (the point correlation voting algorithm comes close).

Previous work on word spotting has focused entirely on the matching process. We go further in this paper and cluster the results of the matching algorithm. K-means and a number of different agglomerative clustering algorithms are compared. We simulate the cluster annotation process and show that in terms of word error rates this approach outperforms one previously published HMM word recognizer on a standard subset of the George Washington collection. Dynamic time warping, while faster than many of the other algorithms suggested, is computationally expensive. An alternative approach, which computes a discrete Fourier representation from the profile features, overcomes this disadvantage. The clustering is done directly using these DFT features. The results show that this approach can outperform even dynamic time warping while still being relatively fast. We believe that this paper shows that the word spotting approach is practical and may be used to create indices or even retrieve handwritten pages given text queries.

The rest of this paper is organized as follows. Section 1 discusses related work in this area. This is followed by a discussion of which clusters are "interesting" to index based

on Luhn's ideas [12]. Profile features are introduced in section 3, followed by section 4 on word image matching. The dynamic time warping algorithm is explained and its results compared with other techniques for matching word images. Clustering is discussed in section 5 followed by the conclusion.

1 Related Work

There are at least three different ways of approach the problem of indexing/retrieving historical documents.

As mentioned in the introduction, the word spotting approach for historical handwritten documents was first proposed by Manmatha et al. [15], and a number of different word matching algorithms were investigated in [14, 13, 24, 26, 25, 27]. Given the difficulty of the image matching problem, it is understandable that none of these papers went beyond and investigated the clustering of these word images. Segmentation for historical documents was investigated in [17, 16].

Rath et al. [32] proposed the first automatic retrieval system for historical handwritten documents using relevance models. Their dataset consisted of 1000 manuscript pages from the George Washington collection. The word spotting approach would be an alternative to building such a system. Results are not directly comparable since the aim of the paper [32] was to do retrieval while the results reported here focus on word error rates (WER).

Another approach involves the use of handwriting recognition, followed by – say – a text search engine. However, handwriting recognition of large vocabulary historical documents

is still a very challenging task. Nagy [19] discusses papers published in PAMI on document analysis during the last 20 years. In recent years, research in handwriting recognition [22] has advanced to a level that makes commercial applications (e.g. Tablet PCs) feasible. However, this success has been mostly limited to the *online* handwriting recognition case, where the pen movements are recorded as a user writes. *Offline* handwriting recognition, that is, recognition of text from an image of the writing, has only been successful in small-vocabulary and highly redundant domains such as automatic check processing and mail sorting (e.g. [9]). Srihari and Kim [31] described an early system for reading unconstrained handwriting. More recently, the community has started to look at large-vocabulary tasks [36].

In [18], the authors discuss the application of a Hidden Markov model for recognizing handwritten material that was produced specifically for this purpose. First, they asked a number of subjects to write out a set of pages. To improve the quality of the writing, the subjects were asked to use rulers and not to split words across lines. Recognition was performed with a Hidden Markov model with 14 states for each character. These Markov models were concatenated to produce word and line models. A statistical bigram language model was used, and the authors showed that this improved the results by about 10%. The authors showed a recognition rate of about 60% for vocabulary sizes ranging from 2703 to 7719 words. The paper also contains a discussion of recognition rates obtained by other researchers - these varied from a recognition rate of 42.5% for a 525 word vocabulary and 37% for a 966 word vocabulary reported in [21], to a recognition rate of 55.6% in a 1600

word vocabulary reported by [8].

There is much less work on historical handwritten documents which are much more challenging. Tomai et al. [33] have shown how difficult this can be: the authors aligned a page of Thomas Jefferson with its manually generated transcript using recognition. Even after restricting the lexicon to about 13 words, their alignment accuracy on this single page was 83% (given a perfect transcript!). Govindaraju and Xie [3] also investigated the problem of handwriting recognition in historical documents. Lavrenko et al. [10] trained an HMM model on 19 pages of the dataset used in this paper and tested on the remaining page. With 20-fold cross validation they obtained a word error rate of 41% (excluding out of vocabulary terms) and 50% (when including out of vocabulary terms). By including bigrams from a Jefferson corpus and a Washington corpus (exluding the test set) they reduced the WER to 35% and 45% respectively. We note that the best approach in this paper has an even lower WER, showing that the word spotting approach is quite competitive.

2 Interesting Clusters

Early work in information retrieval by Luhn [12] lets us concretize the notion of "interesting" clusters. A plot of term frequencies, where terms are ordered by decreasing frequency of occurence, exhibits a distribution that is known as *Zipf's law* [37]. That is, the frequency of the k-th most frequent term has a frequency that is f_0/k, where f_0 is the frequency of the most frequent term. Luhn argued that index terms should be taken from the middle of that distribution. Figure 3 shows an example of the actual distribution of term frequencies

and the distribution predicted by Zipf. Note the large amount of mass that is concentrated in high-frequency terms and the long tail of the distribution to the right, which continues beyond the shown range.

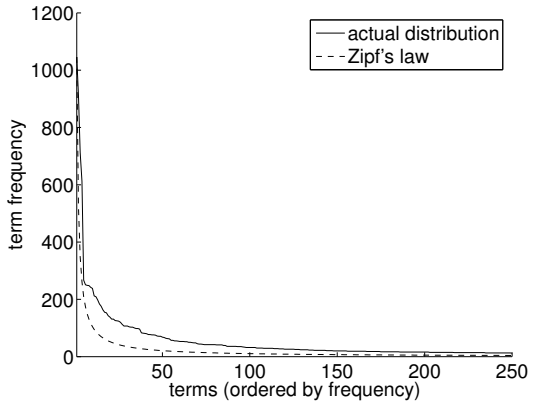


Figure 3: Zipf's law: The plots show the actual distribution of term frequencies and the prediction made with Zipf's law based on the actual frequency of the most frequent term. The collection size is 21324 words; only the left-hand portion of the graph is shown.

The reason is that terms with frequencies that are high (left side of the plot) are often *stop words*, such as *and/the/...*, which do not carry any meaning. Terms with very low frequencies are often sporadic, and are not descriptive of the content in the collection. Terms that are descriptive of the content can often be found in the middle of the plot. Their repeated, but not excessive use suggests that they are essential to describing the content of the collection and should consequently be part of the index.

In the following sections we assume the output of a page segmentation algorithm and describe approaches to matching pairs of word images and clustering experiments.

3 Features

The images we operate on are all grayscale with 256 levels of intensity [0..255]. Before column features can be extracted from an image, inter-word variations such as skew and slant angle have to be detected and normalized. All of the column features we describe in the following are normalized to the range [0..1]. Specific pixel intensity values in an image $I \in \mathbf{R}^{h \times w}$ are referred to as $I(r, c)$, where r and c indicate the row and column index of the pixel. Our goal was to choose a variety of features presented in handwriting recognition literature (e.g. [35]), such that an approximate reconstruction of a word from its features would be possible.

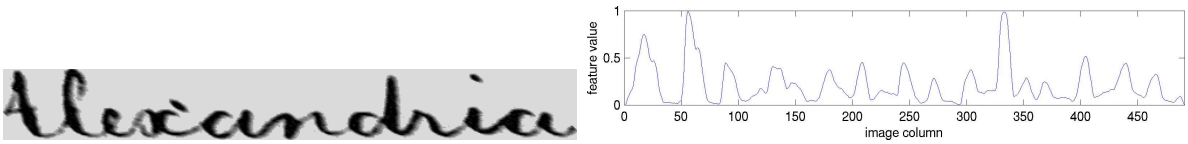
3.1 Projection Profile

Projection profiles capture the distribution of ink along one of the two dimensions in a word image. A vertical projection profile is computed by summing the intensity values[1] in each image column separately:

$$pp(I, c) = \sum_{r=1}^{h} (255 - I(r, c)). \tag{1}$$

Due to the variations in quality (e.g. contrast, faded ink) of the scanned images, different projection profiles do not generally vary in the same range. To make them comparable, the range of the projection profiles is normalized to the range [0..1] which gives $f_1(I, c)$.

[1] We chose to invert the pixel intensities, because the result is visually more intuitive (peaks for pronounced vertical components in input word image). The descriptive capabilities of the feature however remain the same.



(a) original image: slant/skew-normalized, cleaned.

(b) normalized projection profile.

Figure 4: Original image and projection profile feature.

Figure 4 shows an example projection profile and the original image it was extracted from.

3.2 Word Profiles

Word profiles capture part of the outlining shape of a word. The current word matching algorithm uses upper and lower word profiles: let $is\_ink(I, r, c)$ be a function that returns 1 if the pixel $I(r, c)$ is an "ink pixel", and 0 if the pixel is a background pixel. This function is currently realized using a thresholding technique which we have found to be sufficient for our purposes. For more sophisticated foreground/background separation, see [11]. Using is_ink, the upper and lower word profiles can be calculated as follows:

$$up(I,c) = \begin{cases} \text{undefined}, & \text{if } \forall r \, (is\_ink(I,r,c) = 0) \\ \underset{r=1,\ldots,h}{argmin} \, (is\_ink(I,r,c) = 1), & \text{otherwise} \end{cases} \tag{2}$$

$$lp(I,c) = \begin{cases} \text{undefined}, & \text{if } \forall r \, (is\_ink(I,r,c) = 0) \\ \underset{r=1,\ldots,h}{argmax} \, (is\_ink(I,r,c) = 1), & \text{otherwise} \end{cases} \tag{3}$$

If a column does not contain ink pixels, up and lp will be undefined (no distance to the

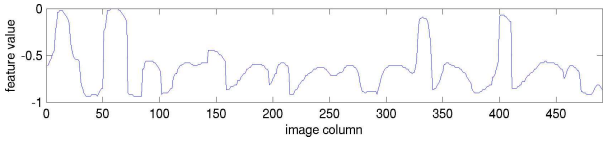


Figure 5: Normalized upper word profile (original in Figure 4(a), negative feature value displayed).

nearest ink pixel from top or bottom of word image bounding box). A number of factors, such as pressure on the writing instrument and fading ink affect the occurrence of such gaps, which is not consistent for multiple instances of the same word. Therefore, gaps where up and lp are undefined were closed by linearly interpolating between the nearest defined values:

$$up'(I,c) = \begin{cases} \text{interpolated value}, & \text{if } up(I,c) \text{ undefined} \\ up(I,c), & \text{otherwise} \end{cases} \tag{4}$$

f_2 (and similarly f_3 from lp') can be obtained by normalizing up' to the range [0..1]. Figure 5 shows an upper word profile feature, generated from the original in Figure 4(a).

3.3 Background/Ink Transitions

So far, the above features represent the distribution of ink in the columns of a word image and the outlining shape of the word. To capture part of the "inner" structure of a word, we chose to record the number of background to ink transitions $nbit(I,c)$ in an image column as the last feature. The range of this feature is normalized with a (conservatively

determined) constant that ensures a range of [0..1]:

$$f_4(I, c) = nbit(I, c)/6. \tag{5}$$

With this feature set at hand, we will now demonstrate its effectiveness when used within the proposed DTW matching algorithm.

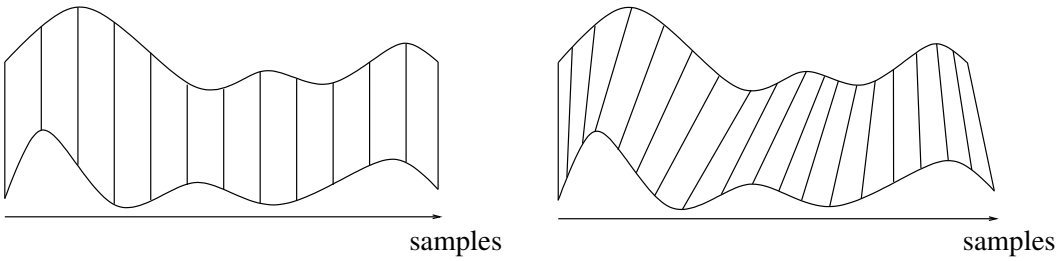
We tried other features, including Gaussian filter responses [25], but the above set seemed to work the best.

4 Word Image Matching

One of the key parts of the wordspotting approach is the image matching technique for comparing word images. Several techniques have been investigated [24, 26, 27], with the best performing being *Dynamic Time Warping* matching [26], which we explain here in detail.

For DTW matching, word images are represented by multidimensional profile features (see section 3). These profiles are then matched using DTW, a dynamic programming algorithm that is able to account for writing variations, which cause the profile features to be compressed and stretched nonlinearly with respect to one another.

The advantage of DTW over simple distance measures such as linear scaling followed by a Euclidean distance calculation is that it determines a common "time axis" (hence the term *time* warping) for the compared signals, on which corresponding profile locations



(a) Linear scaling alignment, (b) Dynamic Time Warping alignment.

Figure 6: Two profiles, aligned using linear scaling and Dynamic Time Warping. DTW ensures that only corresponding locations will be compared.

appear at the same time. Due to the variations in handwriting, two profiles of the same word do not generally line up very well if they are just scaled linearly (see Figure 6).

4.1 Dynamic Time Warping

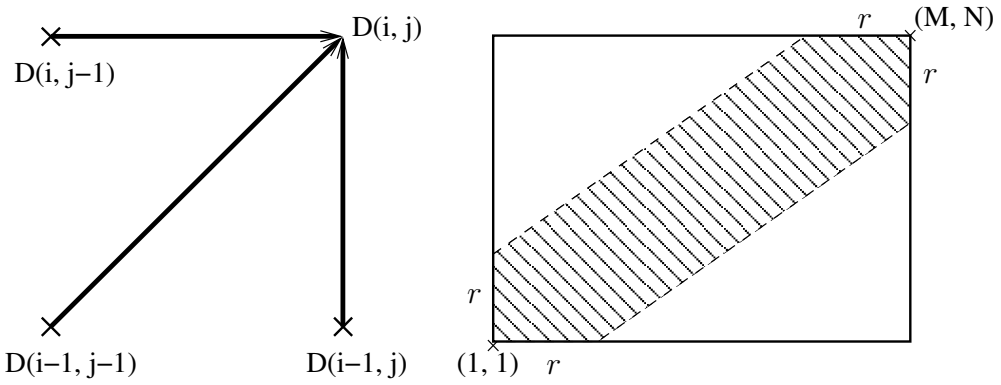
When determining the DTW-distance[2] $dist(\mathbf{X}, \mathbf{Y})$ between two time series $\mathbf{X} = (x_1, \ldots, x_M)$ and $\mathbf{Y} = (y_1, \ldots, y_N)$, a matrix $D \in I\!R^{M \times N}$ is built, where each entry $D(i, j)(1 \leq i \leq M, 1 \leq j \leq N)$ is the cost of aligning the subsequences $\mathbf{X}_{1:i}$ and $\mathbf{Y}_{1:j}$.

Each entry $D(i, j)$ is calculated from some $D(i', j')$ plus an additional cost d, which is usually some distance between the samples x_i and y_i. For instance, our implementation of the algorithm uses

$$D(i,j) = min \left\{ \begin{array}{l} D(i, j-1) \\ D(i-1, j) \\ D(i-1, j-1) \end{array} \right\} + d(x_i, y_i). \tag{6}$$

[2] $dist(\cdot, \cdot)$ does not satisfy all metric axioms.

The recursive definition of $D(i, j)$ based on the given three values is a *local continuity constraint* (cf. Figure 7(a)). It ensures smoothness of the recovered warping, e.g. no sample can be left out in a warped sequence. For a more detailed discussion of continuity constraints and alternatives to the one used in this work, we refer the reader to [29].



(a) local continuity constraint. (b) global path constraint.

Figure 7: Constraints used in the current dynamic time warping implementation.

Table 1 contains pseudo-code of the DTW algorithm (adapted from [34]) using the local continuity constraint from Figure 7(a). The algorithm determines a *warping path* composed of index pairs $((i_1, j_1), (i_2, j_2), \ldots, (i_K, j_K))$, which aligns corresponding samples in the input sequences $\mathbf{X}$ and $\mathbf{Y}$. To prevent DTW from recovering pathological warpings that match a small portion of one series to a large portion in the other, *global path constraints* are used. They force the recovered paths to stay close to the diagonal of the DTW matrix. Our implementation of DTW uses the *Sakoe-Chiba* band [28] (see Figure 7(b); the warping path must lie in the shaded region), but the *Itakura* parallelogram [6] is also a popular choice. As a side effect, the constraint speeds up the computation of the DTW matrix, since it does not have to be entirely evaluated. Our implementation of the word matching

algorithm uses $r = 15$ samples. Recent work [23] shows that the size and shape of the global path constraint can be adapted, leading to faster DTW computations and better matching performance.

Input: $\mathbf{X} = (x_1, \ldots, x_M)$ and $\mathbf{Y} = (y_1, \ldots, y_N)$, distance function $d(\cdot, \cdot)$
Output: DTW matrix D
Algorithm:
1. $D(1,1) = d(x_1, y_1)$;
2. for $m = 1 : M$
3. $\quad D(m,1) = D(m-1,1) + d(x_m, y_1)$;
4. for $n = 1 : N$
5. $\quad D(1,n) = D(1,n-1) + d(x_1, y_n)$;
6. for $m = 2 : M$
7. $\quad$ for $n = 2 : N$
8. $\quad D(m,n) = min\begin{Bmatrix} D(i,j-1) \\ D(i-1,j) \\ D(i-1,j-1) \end{Bmatrix} + d(x_m, y_n)$;

Table 1: Pseudo code for the DTW algorithm.

Once all necessary values of D have been calculated, the warping path can be determined by backtracking the minimum cost path starting from (M, N). We are just interested in the accumulated cost along the warping path, which is stored in $D(M, N)$. As it is, this matching cost would be lower for shorter sequences, so we offset this bias by dividing the total matching cost by the length K of the warping path, yielding

$$dist(\mathbf{X}, \mathbf{Y}) = D(M, N)/K. \tag{7}$$

4.2 Matching Word Images with DTW

We represent word images with single- or multi-dimensional profile features. Single-dimensional profiles that were extracted from the same word have the same length, and can be "stacked" to create multidimensional profiles. Hence, when matching word images, the sequences $\mathbf{X}$ and $\mathbf{Y}$ consist of samples of dimensionality $d \geq 1$, i.e. $x_i, y_i \in \mathbb{R}^d$.

In order to use DTW to match such profiles, we need to define a distance measure $d(\cdot, \cdot)$ that determines the (local) distance of two samples in a profile. Our implementation uses the square of the Euclidean distance

$$d(x_i, y_j) = \sum_{k=1}^{d} (x_{i,k} - y_{j,k})^2, \tag{8}$$

where the index k is used to refer to the k-th dimension of x_i and y_j. With this distance measure defined, we can now calculate the matching distance between two word images by comparing their profile features using DTW and equation (7).

4.3 Experimental Setup

The performance of the DTW word image matching algorithm was evaluated in a retrieval-by-example setup, where word images in a collection are ranked by decreasing similarity to a given template. Experiments were conducted on two test sets of different quality, both 10 pages in size (2381 and 3370 word images). The first set is of acceptable quality, see Figure 8(a)). The second set is very degraded (see Figure 8(b)) - even people have difficulties reading these documents. We used this data set to test how badly matching

algorithms perform on manuscripts of such poor quality. Each page in the two test sets was segmented into words with an automatic page segmentation procedure. While the quality of the segmentation algorithm has been improved in the meantime, we used the same segmentation results as in [7], for comparability.

We conducted four experiments on the test sets and compared the performance of various matching approaches. Each experiment involves selecting one of the above two data sets and identifying a subset that will be used for querying. Each of the queries is used to rank the images in the dataset according to their similarity to the query. The similarity scores are determined by a matching algorithm. Four experiments were conducted (A and C were initially proposed in [7]):

Experiment A: 15 images from test set 1 were selected as queries.

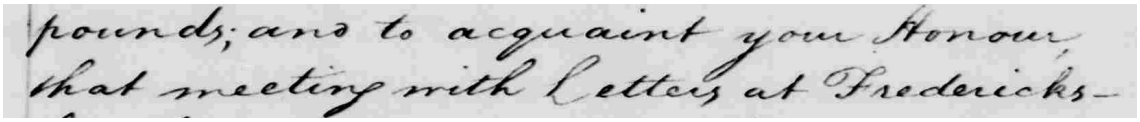
Experiment B: All images in test set 1 were used as queries. This yields a total of 2381 query images, 9 of which do not contain any letters.[3]

Experiment C: 32 images from test set 2 were selected as queries. 13 of these images contain words that occur only once in the collection.

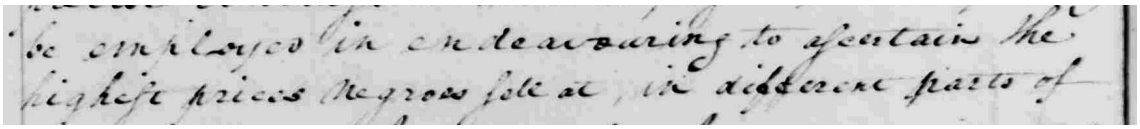
Experiment D: All images in test set 2 were used as queries. This yields a total of 3370 query images total, 108 of which do not contain any letters.[3]

Experiments A and C allow us to test algorithms which would otherwise take too long to run on the entire dataset.

[3]These images are the result of segmentation errors.



(a) acceptable quality (set 1).



(b) significantly degraded (set 2).

Figure 8: Document samples from the Dynamic Time Warping testbeds, showing the differing quality.

In order to reduce the number of pairwise comparisons that have to be made, we pruned the total set of matches by comparing scalar features. If thresholding applied to the features extracted from the query and a candidate image determines that the images are dissimilar, we do not assign a similarity score to the candidate image (see [26] for details of this process). Table 2 shows the reduction of candidates through pruning, and how many true positives remain in the pruned candidate set.

Experiment	Total #queries	Pruned candidates	Recall
A	15	87.29%	90.72%
B	2372	86.43%	71.11%
C	32	86.99%	56.49%
D	3262	85.74%	55.05%

Table 2: Pruning statistics, showing the reduction in the total number of matches to be made and the percentage of remaining true positives (recall).

Each word in the data sets was labeled with its ASCII equivalent. In case of segmentation errors, a tag corresponding to all visible characters in the segmented word image

was assigned. Based on this annotation, relevance judgments were produced for the data sets. Two word images were considered relevant, if they have the same tags. For the evaluation, we used the `trec_eval` program to compute mean average precision scores for all experiments.

4.4 Experimental Results

Table 3 shows mean average precision results for all data sets obtained with a range of different matching techniques:

1. XOR[7]: The images are aligned to compensate for shear and scale changes, binarized and then a difference image is computed. The number of difference pixels is used as the matching cost.

2. SSD[7]: This approach translates the query and candidate images relative to one another to find the minimum cost based on the sum of squared differences of the pixel intensities. This cost is used as the matching cost.

3. SLH[7]: Scott and Longuet-Higgins algorithm [30]. This algorithm recovers an affine warping transform between sample points taken from the boundaries of the query and candidate images. The residual between query points and the warped candidate points is the matching cost.

4. SC [2]: Shape context matching. Sample points are taken from the outlines of the query and candidate image. Each point is assigned a *shape context* histogram, which

is used to recover corresponding sample points in the query and the candidate image. The matching is done in an iterative process, which successively warps the candidate image. The matching cost is determined from the cost that is associated with the chosen correspondences.

5. EDM[7]: Euclidean distance mapping. In the XOR image, difference pixels in larger regions are weighted more heavily, because they are likely to result from structural differences between the template and the candidate image, not from noise. The matching cost is the cumulative weight of the difference pixels in the XOR image.

6. CORR[27]: This technique recovers similarities between a small number of corner points in the query and candidate images. Points of interest are determined by a corner detector. The matching cost is calculated from the number of recovered correspondences and the relative location of corresponding points in the two images.

7. DTW[26]: Dynamic Time Warping word image matching as described above.

The obtained mean average precision scores for experiments A and B had to be corrected, because of an evaluation problem in [7]. The reason is that Kane et al. ranked all the images in the dataset, including the query image. Queries with only one single relevant item (the query itself) produce average precision values of 1 (because the query image is retrieved at rank 1), which artificially inflates the retrieval scores. To solve this problem we have chosen to disregard 13 of the queries in set C and 960 queries in set D. Table 3 reflects the values after this correction.

Despite this correction, the remaining queries continue to retrieve the query image at rank 1, still inflating the scores. Hence, in order to give the most accurate picture of the actual matching performance, we have re-calculated the mean average precision scores for test-runs that were available to us in ranked-list format. Table 4 shows the average precision scores of four runs, where the top ranked image is removed from all ranked result lists, effectively discarding each query image from their respective candidate set.

Algorithm	**Exper. A**	**Exper. B**	**Exper. C**	**Exper. D**
XOR	54.14%	n/a	n/a	n/a
SSD	52.66%	n/a	n/a	n/a
SLH	42.43%	n/a	n/a	n/a
SC	48.67%	n/a	48.11%	n/a
EDM	72.61%	n/a	15.05%	n/a
CORR	73.95%	62.57%	59.96%	51.08%
DTW	73.71%	65.34%	58.81%	51.81%

Table 3: Mean average precision scores on all data sets (results for test set A and B have been corrected, XOR: matching using difference images, SSD: sum of squared differences technique, SLH: technique by Scott & Longuet-Higgins [30], SC: shape context matching [2], EDM: euclidean distance mapping, CORR: corner-point correlation, DTW: dynamic time warping matching).

For experiment A, results are available with all matching algorithms. EDM, DTW and CORR clearly outperform any of the other techniques. SC was run with a number of sample points proportional to the width of the words being matched, with about 100 sample points for a word like *Alexandria*. More sample points would probably improve the effectiveness of the technique, but at the cost of further increasing the matching time (for 100 sample points it is already about 50 seconds, see Table 5).

The DTW and CORR algorithms were also used in experiment B (all images used as templates). The other algorithms were too slow to realistically run on this dataset. On query set B, the average precision scores for DTW and CORR are lower than on the smaller subset A. We attribute this effect mostly to the pruning method, which works much better on the smaller set A: while the pruning preserves about 91% of the relevant documents for data set A, it only produces 71% recall on data set B. The lower recall on set B (due to the pruning) then results in a lower average precision score after matching. While the performance of DTW was slightly worse than CORR's on the smaller query set A, DTW outperforms CORR on query set B, which is much larger and makes for a better comparison.

We compared the results of the SC, CORR, EDM and DTW techniques on data set C. While the performance of all approaches is generally low on data set C, DTW's and CORR's performance is almost four times better than that of EDM (58.81% and 59.96% vs. 15.05%). DTW also performs similarly on the rest of the data set (51.81% average precision on data set D). This shows that the DTW and CORR matching techniques are more robust to document degradation than EDM, with DTW - again - showing superior performance to CORR on the exhaustive query set. We would expect the results to be better, if a more careful pruning was applied: after pruning, the recall percentages have already dropped to about 56% for sets C and D (see Table 2). This limits the maximum average precision achievable with the matching algorithms.

These results show that DTW performs best amongst the set of algorithms tried. However, a look at Table 4 shows that significant efforts need to be made in order to perform

Algorithm	**Exper. A**	**Exper. B**	**Exper. C**	**Exper. D**
SC	40.58%	n/a	9.46%	n/a
EDM	67.67%	n/a	n/a	n/a
CORR	69.69%	36.23%	14.84%	15.49%
DTW	67.92%	40.98%	13.04%	16.50%

Table 4: Mean average precision scores using the alternative evaluation measure for all test results that were available to us in ranked list format. SC: shape context matching [2], EDM: euclidean distance mapping, CORR: corner-point correlation, DTW: dynamic time warping matching).

well on such challenging datasets as C and D. Whether this improvement will come from the preprocessing or from the matching algorithm itself remains to be seen.

Algorithm:	XOR	SSD	SLH	SC	EDM	CORR	DTW
Running time [s]:	13	72	121	$\sim$50	14	$\sim$1	$\sim$2

Table 5: Run times for the compared algorithms in *Matlab* on a 400MHz machine. The values include the time that is spent on normalization (e.g. relative scaling), feature extraction, and similar processing steps.

Comparing the running times of the investigated algorithms (see Table 5) shows CORR in the lead. CORR's superior execution is a result of the very few corner points that are considered for establishing correspondences between the query and a candidate image. DTW is second in execution time, but we believe its performance can be improved substantially with optimization. The other algorithms (including our implementation of SLH) all use the actual images (rather than the 1D profiles used by DTW) and hence are much slower.

5 Word Image Clustering Experiments

All of the previous work on wordspotting has concentrated mostly on finding effective similarity measures for word image matching, but the clustering of word images has not been tackled. In this section, we perfom word image clustering experiments, followed by simulated cluster annotations that are designed to imitate a human annotator.

Before we start clustering, we need to get a good estimate of the number of clusters that our data will form. *Heaps' law*, an empirical rule, provides the tool for the estimation, which is discussed in the following section. With an accurate cluster estimate we then move on to various clustering techniques that we apply to group word images.

5.1 Heaps' Law

Many clustering algorithms often require that the number of clusters to be created is known. In fact, all of the clustering algorithms that were used in our experiments have the target number of clusters as an input parameter. In the ideal case, each cluster contains all instances of a particular word, so there are as many clusters as there are distinct words in the collection at hand. In other words, the number of clusters is equivalent to the vocabulary size.

Early work in information retrieval by Heaps [5] provides an empirical estimate for the *vocabulary* size of a collection from the size of the collection in words. The rule, which is known to be quite effective [1], has become known as *Heaps' law*. It predicts that the

vocabulary size of a collection of n words can be estimated to be

$$V(n) = K \cdot n^{\beta}, \tag{9}$$

where K and β are parameters that depend on the language of the collection.

We estimated K and β by fitting Heaps' law to the ground truth transcription of a collection of 100 pages (21324 word images) from George Washington's letters, which does not include our testing set on which we performed clustering experiments. We simulated documents of sizes $n = 1$ to $n = 21324$ by only considering the first n transcription words. For each n, we determined the vocabulary size and then fitted Heaps' law to the resulting curve. Figure 9 shows a plot of the vocabulary size V as a function of n and the fitted curve.

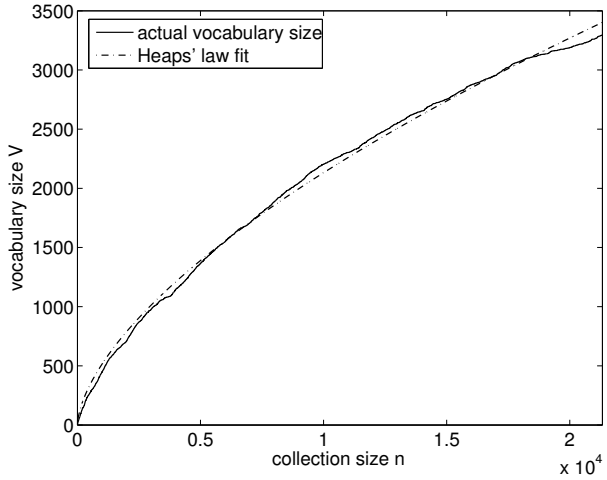


Figure 9: Actual vocabulary size as a function of the collection size and a fit of Heaps' law shown for collection sizes of up to 21324 words.

The fitting was performed with the "Nelder-Mead" optimization procedure [20], which minimizes the sum of squared differences between the actual vocabulary sizes and the ones

predicted by Heaps' law. For the collection at hand, we estimated the optimal parameter settings to be $K = 7.2416$ and $\beta = .6172$, resulting in a tight fit.

We used these parameters to estimate the vocabulary size of a collection of 20 pages, our testbed for the clustering experiments in the following section. The annotated data set is publicly available and was originally used in [10] for doing recognition experiments. It consists of 4860 word images. [4]

#Images	**Voc. size**	**Predicted Voc. size**	**Est. error**
4860	1187	1365	15%

Table 6: Accuracy of the vocabulary size prediction (with Heaps' law), shown for two data sets.

Table 6 shows the accuracy that Heaps' law achieves when predicting the vocabulary size of the data set. The vocabulary size of the collection is overestimated by 15%. This appears acceptable, given the small size of the collection. It is also possible that a larger text source for the parameter estimation could yield better prediction results.

5.2 Clustering

With the desired number of clusters at hand, we can now turn to grouping word images based on pairwise similarity and then determine the accuracy of the generated clusters. Two sets of experiments were performed using 2 different image similarity metrics. The main difference between them lies in the features that are used for the representation:

[4]The original dataset had 4856 word images. Based on feedback, a small number of segmentation and annotation mistakes were corrected.

α: The first experimental set consists of 4860 word images. This set does not consist of feature vectors, but rather of a 4860×4860 sparse matrix with pairwise distances. The distances were calculated using the Dynamic Time Warping word matching algorithm described in section 4.2 above. The matrix is sparse, because pairwise distances were only calculated if a word image pair was not ruled out by pruning. 76% of the matrix entries were not computed and were filled with the default distance value ∞. The calculation of the distance matrix required roughly one week on a multiprocessor machine with 4 CPUs and 500MHz.

β: This set consists of all the 4860 word images in the dataset, solely represented with DFT coefficients extracted from the upper-, lower-, and projection-profiles. 33 DFT coefficients were used to represent each word image, which is the optimal number that was determined for this dataset.

We experimented with both the K-means clustering algorithm and various agglomerative clustering approaches on these data sets with one exception: Since dataset α is not represented in feature space, but rather in terms of pairwise distances, K-means clustering cannot be applied. K-means keeps track of cluster centers, a notion that does not exist in pairwise distance space.

Numerous clustering techniques are described in the literature. The following is a brief overview of the clustering approaches that were used in our experiments. More detailed descriptions of clustering techniques can be found in relevant literature, e.g. [4]. Except for K-means clustering, all others techniques are *agglomerative* bottom-up procedures, which

build a hierarchical cluster tree by successively linking clusters. For such clustering techniques, we only list how inter-cluster dissimilarity is determined:

K-means: The algorithm is initialized with K randomly selected cluster centers. Then each feature vector is assigned to the nearest cluster, and the cluster centers are recalculated. This procedure is repeated until convergence.

Single linkage: The inter-cluster dissimilarity between two clusters is the distance between the closest items within the two clusters.

Complete linkage: The distance between the two furthest items in the clusters is used as the cluster dissimilarity.

Average linkage: Here the distance between two clusters is the average distance between all item pairs in the clusters.

Weighted linkage: A slight variation of the *Average Linkage* technique, which uses a weighted average for the cluster distance calculation.

Ward's linkage: This linkage uses the sum of squares measure to assess the similarity between clusters. The sum of squares is the total squared distance of all items in a cluster relative to the cluster centroid. The distance of two clusters is then taken to be the increase in the sum of squares measure, before joining them and taken together.

Each of our experiments involves selecting one of α and β and a clustering method. First, the desired number of clusters is estimated using Heaps' law. Then, we start the

clustering of the data. In the case of K-means clustering, the feature vectors form the input for the clustering algorithm. All other clustering routines use a dendrogram as input, which can be constructed from pairwise distances between word images or their feature vectors (we used the Euclidean distance measure to calculate distances between feature vectors). The output of the clustering is a vector of cluster labels, which assigns each word image to a single cluster.

The accuracy of a particular clustering output is evaluated by simulating the task of labeling clusters, which would be performed by a human annotator if we were to perform wordspotting. For the purpose of the simulation, it is assumed that a human annotator would label a cluster with the vocabulary term that occurs most frequently in a cluster. This strategy is sound, because it minimizes the total number of wrong annotations, when cluster labels are spread over all word images within a cluster. Ground truth data is available for all word images, so this process can be easily simulated. Once all clusters have been annotated in this fashion, we assign each cluster label to all word images within the cluster, essentially transcribing the entire collection. In practice, one could imagine a scenario where the user very quickly browses (overviews) a set of word images in a cluster and then assigns a label to the cluster. Table 7 shows the word error rates of such transcriptions obtained from various clustering approaches.

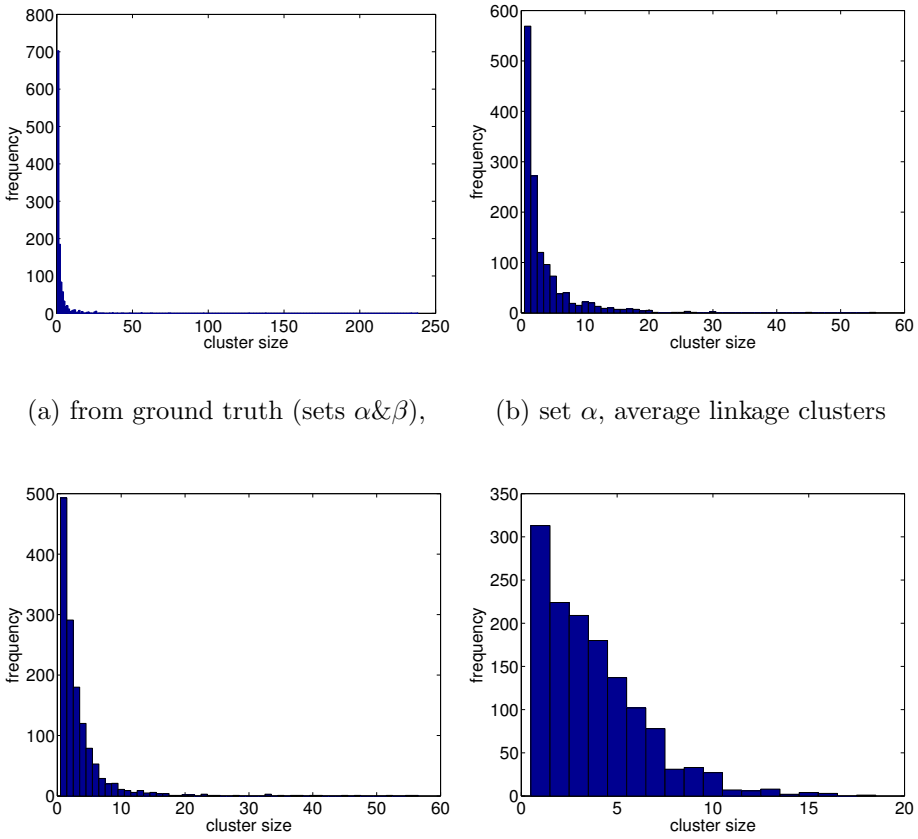
The clustering algorithms tend to perform quite well. Data set β yielded the best overall result, with *Ward* linkage clustering. Interestingly, the DTW dissimilarity data (set α) performs slightly worse using the Ward linkage, but otherwise consistently better than

Clustering algorithm	**α: WER**	**β: WER**
K-means	n/a	41.58%
Single linkage	65.00%	65.10%
Complete linkage	36.11%	37.24%
Average linkage	34.12%	44.47%
Weighted linkage	34.77%	41.03%
Ward linkage	34.47%	**31.50%**

Table 7: Performance of the clustering algorithms in terms of word error rate (WER), after simulated annotation of the entire collection.

set β, and consistently well with word error rates between 34% and 36% (except for the single linkage algorithm). This suggests that the DTW distance measure captures different aspects of word image similarity than the features used in set β. The matrix with DTW distances has not been entirely computed due to pruning, which should have an adverse effect on performance (the pruning assigns a distance of ∞ to a large portion of all possible word image pairs). Without the pruning, it would be impossible to run the DTW algorithm in a realistic amount of time on this dataset.

Figure 10 shows histograms of the sizes of clusters that have been generated with the best performing methods on sets α and β (average linkage and Ward linkage), as well as the output of a clustering technique with higher word error rate (K-means). The clustering techniques with lower word error rates generate better matches for the actual distribution of cluster sizes. This is also true of techniques for which no plots are provided. It is important for a good clustering approach to produce clusters of a variety of sizes. The output of the K-means clustering in Figure 10(d) shows that clusters which should have been large, were broken down into smaller pieces.



(a) from ground truth (sets $\alpha\&\beta$),

(b) set α, average linkage clusters

(c) set β, Ward linkage clusters

(d) set β, K-means clusters

Figure 10: Histograms of actual and automatically determined cluster sizes. Some clustering algorithms achieve a good match of the actual cluster size distribution (b),(c), while others tend to produce clusters with a limited range of sizes (d).

Of course our goal is not to obtain labels for all word images in the collection. Following Luhn's line of thought, we can identify clusters that should make good candidates for an index. We constrained the simulated annotation to clusters with at least 3 members, but not more than 50, and calculated the word error rate for the simulated annotation that is restricted only to the selected clusters. Table 8 shows the word error rates that were achieved for such clusters and the number of word images in the collection that were assigned a label.

Clustering alg.	α**: WER/#Img**	β**: WER/#Img**
K-means	n/a	50.80%/2941
Single linkage	4.90%/715	3.83%/758
Complete linkage	42.58%/2790	39.64%/2422
Average linkage	41.66%/2787	37.92%/2070
Weighted linkage	41.72%/2656	40.87%/2273
Ward linkage	41.88%/2827	38.12%/2867
Actual #images in sel. clusters	**#Img Set** α	**#Img Set** β
	2567	2567

Table 8: Performance of the clustering algorithms, computed for clusters with a moderate number of members. The word error rate was calculated for annotations from clusters with at least 3 members, but not more than 50. *#Img* refers to the total number of images that fall into such clusters. The last row of the table shows the correct value for *#Img*, according to the ground truth annotations.

The results show increased word error rates (not including clusterings that significantly underestimate #Img), indicating that the clustering performs slightly better on words that were excluded from the word error rate calculation. Generally, the clusterings of sets α and β come close to the desired number #Img. Further testing would have to be done to reveal which words the clusters in these experiments correspond to.

6 Conclusion

Wordspotting appears as an attractive alternative to the seemingly obvious *recognize-then-retrieve* approach to historical manuscript retrieval. With the capability to match word images quickly and accurately, partial transcriptions of a collection can be achieved with reasonable accuracy and little human intervention. Wordspotting has the capability to automatically identify indexing terms, making it possible to use costly human labor more

sparingly than a full transcription would require. For example, using the Ward linkage clustering on data set β, it would be possible to obtain 2867 word image labels with a word error rate of 38.12%, by annotating just 291 clusters (cluster sizes between 3 and 50 members). That is, the wordspotting procedure would have reduced 2867 annotations to about 10% of that. Even greater savings (in terms of percent) can be expected from larger collections, since vocabularies grow sub-linearly in the size of the corresponding collections.

Acknowledgment

This work was supported in part by the Center for Intelligent Information Retrieval and in part by the National Science Foundation under grant number IIS-9909073. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

We would like to thank the Library of Congress for providing the digitized manuscript images that were used in this work.

References

[1] Baeza-Yates, R., and Ribeiro-Neto, B. *Modern Information Retrieval*. Addison-Wesley, Reading, MA, 1999.

[2] Belongie, S., Malik, J., and Puzicha, J. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence 24*, 4

(2002), 509–522.

[3] Govindaraju, V., and Xue, H. Fast handwriting recognition for indexing historical documents. In *Proc. of the Int'l Workshop on Document Image Analysis for Libraries* (Palo Alto, CA, January 23-24 2004), pp. 314–320.

[4] Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer-Verlag, New York, 2001.

[5] Heaps, H. S. *Information Retrieval: Computational and Theoretical Aspects*. Academic Press, Orlando, FL, 1978.

[6] Itakura, F. Minimum prediction residual principle applied to speech recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing 23* (1975), 67–72.

[7] Kane, S., Lehman, A., and Partridge, E. Indexing george washington's handwritten manuscripts. Tech. rep., Center for Intelligent Information Retrieval, Univ. of Massachusetts Amherst, 2001.

[8] Kim, G., Govindaraju, V., and Srihari, S. N. An architecture for handwritten text recognition systems. *Int'l Journal on Document Analysis and Recognition 2*, 1 (1999), 37–44.

[9] Kornai, A., Mohiuddin, K. M., and Connell, S. D. Recognition of cursive writing on personal checks. In *Proc. of the 5th Int'l Workshop on Frontiers in Handwriting Recognition* (Colchester, UK, September 2-5 1996).

[10] Lavrenko, V., Rath, T. M., and Manmatha, R. Holistic word recognition for handwritten historical documents. In *Proc. of the Int'l Workshop on Document Image Analysis for Libraries* (Palo Alto, CA, January 23-24 2004), pp. 278–287.

[11] Leedham, G., Varma, S., Patankar, A., and Govindaraju, V. Separating text and background in degraded documents images — a comparison of global thresholding techniques for multi-stage thresholding. In *Proc. of the 8th Int'l Workshop on Frontiers in Handwriting Recognition* (Niagara-on-the-Lake, Canada, August 6-8 2002), pp. 244–249.

[12] Luhn, H. P. The automatic creation of literature abstracts. *IBM Journal 2* (April 1958), 159–165.

[13] Manmatha, R., and Croft, W. B. Word spotting: Indexing handwritten manuscripts. In *Intelligent Multimedia Information Retrieval*, Mark T. Maybury, Ed. MIT Press, Cambridge, MA, 1997, pp. 43–64.

[14] Manmatha, R., Han, C., and Riseman, E. M. Word spotting: A new approach to indexing handwriting. In *Proc. of the Conf. on Computer Vision and Pattern Recognition* (1996), pp. 631–637.

[15] Manmatha, R., Han, C., Riseman, E. M., and Croft, W. B. Indexing handwriting using word matching. In *Digital Libraries '96: 1st ACM Int'l Conf. on Digital Libraries* (Bethesda, MD, March 20-23 1996), pp. 151–159.

[16] Manmatha, R., and Rothfeder, J. A scale space approach for automatically segmenting words from historical handwritten documents. *IEEE Trans. on Pattern Analysis and Machine Intelligence* (2005). (to appear).

[17] Manmatha, R., and Srimal, N. Scale space technique for word segmentation in handwritten manuscripts. In *Proc. of the Second Int'l Conf. on Scale-Space Theories in Computer Vision* (Corfu, Greece, September 26-27 1999), pp. 22–33.

[18] Marti, U.-V., and Bunke, H. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *Int'l Journal of Pattern Recognition and Artifical Intelligence 15*, 1 (2001), 65–90.

[19] Nagy, G. Twenty years of document image analysis in pami. *IEEE Trans. on Pattern Analysis and Machine Intelligence 22*, 1 (2000), 38–62.

[20] Nelder, J. A., and Mead, R. A simplex method for function minimization. *Computer Journal 7* (1965), 308–313.

[21] Pacquet, T., and Lecourtier, Y. Recognition of handwritten sentences using a restricted lexicon. *Pattern Recognition 26*, 3 (1993), 391–407.

[22] Plamondon, R., and Srihari, S. N. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence 22* (2000), 63–84.

[23] Ratanamahatana, C. A., and Keogh, E. Making time-series classification more accurate using learned constraints. In *Proc. of the 4th SIAM Int'l Conf. on Data Mining* (Lake Buena Vista, FL, April 22-24 2004), pp. 11–22.

[24] Rath, T. M., Kane, S., Lehman, A., Partridge, E., and Manmatha, R. Indexing for a digital library of George Washington's manuscripts: A study of word matching techniques. Tech. rep., Center for Intelligent Information Retrieval, Univ. of Massachusetts Amherst, 2000.

[25] Rath, T. M., and Manmatha, R. Features for word spotting in historical manuscripts. In *Proc. of the 7th Int'l Conf. on Document Analysis and Recognition* (Edinburgh, Scotland, August 3-6 2003), vol. 1, pp. 218–222.

[26] Rath, T. M., and Manmatha, R. Word image matching using dynamic time warping. In *Proc. of the Conf. on Computer Vision and Pattern Recognition* (Madison, WI, June 18-20 2003), vol. 2, pp. 521–527.

[27] Rothfeder, J. L., Feng, S., and Rath, T. M. Using corner feature correspondences to rank word images by similarity. In *Proc. of the Workshop on Document Image Analysis and Retrieval (electronically published)* (Madison, WI, June 20 2003).

[28] Sakoe, H., and Chiba, S. Dynamic programming optimization for spoken word recognition. *IEEE Trans. on Acoustics, Speech and Signal Processing 26* (1980), 623–625.

[29] Sankoff, D., and Kruskal, J. B. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley, Reading, MA, 1983.

[30] Scott, G. L., and Longuet-Higgins, H. C. An algorithm for associating the features of two patterns. *Proc. of the Royal Society of London B224* (1991), 21–26.

[31] Srihari, S., and Kim, G. Penman: A system for reading unconstrained handwritten page images. In *Symposium on document image understanding technology (SDIUT 97)* (April 1997), pp. 142–153.

[32] T. M. Rath, V. Lavrenko, and Manmatha, R. A search engine for historical manuscript images. In *Proc. of the 27th Annual Int'l ACM SIGIR Conf.* (Sheffield, UK, July 25-29 2004), pp. 369–376.

[33] Tomai, C. I., Zhang, B., and Govindaraju, V. Transcript mapping for historic handwritten document images. In *Proc. of the 8th Int'l Workshop on Frontiers in Handwriting Recognition* (Niagara-on-the-Lake, ON, August 6-8 2002), pp. 413–418.

[34] Triebel, R. Automatische Erkennung von handgeschriebenen Worten mithilfe des Level-building Algorithmus, December 1999. Student Thesis, Institut für Informatik, Albert-Ludwigs-Universität Freiburg (in German).

[35] Trier, Ø. D., Jain, A. K., and Taxt, T. Feature extraction methods for character recognition - a survey. *Pattern Recognition 29*, 4 (1996), 641–662.

[36] Vinciarelli, A., Bengio, S., and Bunke, H. Offline recognition of large vocabulary cursive handwritten text. In *Proc. of the 7th Int'l Conf. on Document Analysis and Recognition* (Edinburgh, Scotland, August 3-6 2003), vol. 1, pp. 1101–1105.

[37] Zipf, G. *Human Behaviour and the Principle of Least Effort.* Addison-Wesley, Cambridge, MA, 1949.