Towards More Effective Distance Functions for Word Image Matching

Raman Jain
Centre for Visual Information Technology
IIIT-Hyderabad, India
ramanjain@students.iiit.ac.in

C. V. Jawahar
Centre for Visual Information Technology
IIIT-Hyderabad, India
jawahar@iiit.ac.in

ABSTRACT

Matching word images has many applications in document recognition and retrieval systems. Dynamic Time Warping (DTW) is popularly used to estimate the similarity between word images. Word images are represented as sequences of feature vectors, and the cost associated with dynamic programming based alignment is considered as the dissimilarity between them. However, such approaches are computationally costly when compared to fixed length matching schemes. In this paper, we explore systematic methods for identifying appropriate distance metrics for a given database or language. This is achieved by learning query specific distance functions which can be computed online efficiently. We show that a weighted Euclidean distance can outperform DTW for matching word images. This class of distance functions are also ideal for scalability and large scale matching. Our results are validated with mean Average Precision (mAP) on a fully annotated data set of 160K word images. We then show that the learnt distance functions can even be extended to a new database to obtain accurate retrieval.

1. INTRODUCTION

Matching two word images by computing an appropriate similarity measure, has many applications in document analysis systems [3, 18, 22]. This includes applications in accessing historic handwritten manuscripts [16, 21], searching for relevant documents in a digital library of printed documents [3], holistic recognition [14] and enhancing OCR accuracies by post processing the classification results [11,19]. In this paper we aim at learning effective similarity measures, which are specific to word images. We limit our scope to matching printed word images. Though our approaches are demonstrated on English, our methods are language independent.

Though words can be matched by comparing holistic features [15], the popular approach for matching has been aligning sequences of feature vectors using Dynamic Time Warping (DTW) [17,20]. A sliding window (or a vertical strip) is moved from left to right and features are computed for each window. This results in a sequence of feature vectors. For computing the distance between two such sequences, they are first aligned using dynamic programming. Cost of alignment is treated as the distance between the two sequences. To match sequences of length M and N, one needs to do $O(MN)$ operations. For retrieving from a large data set, query has to be matched with every word in the database. For large scale matching and retrieval, thus, this becomes the bottleneck. Replacing DTW by Euclidean distance (possibly at the cost of accuracy) has been a step towards scalability [13]. This can result in constant time (i.e., $O(d)$) retrieval.

Computing similarity based on a simple Euclidean distance does not use the *knowledge* that the comparisons are made between two *word* images. While computing the distance between two feature vectors (whether they are made out of sequences or holistic features), we need to note that the individual features need not be uncorrelated. Most of the comparison techniques also do not use the fact that the words are generated out of a language model. This could further impose constraints on the possible feature vectors which could be generated from words in a given language. In this paper, we explore distance functions which can be learnt from examples. Our objective is to capitalize on the fact that not all possible combination of characters are valid in a language, and a distance function learnt from training examples can actually benefit matching and retrieval on a test (unseen or unannotated) data set. We validate our claims on annotated and unannotated data sets presented in Section 2. Since we have a reasonably large annotated corpus for comparison, we use mean average precision (area under the precision-recall curve) or mean precision for statistically validating the approach.

We start our experiments by comparing DTW and Euclidean distance in Section 3. We observe that it is not reasonable to conclude DTW is always superior to a fixed length representation. This gives us hope for building efficient, at the same time effective similarity measures. We then show how a specific weighted Euclidean distance can perform superior in a given setting (say when the set of possible queries are known *apriori*). We design a query-specific classifier (QSC), which is obtained by learning the weights (parameter associated with the distance function) on a "training" data set

(Section 4). However, this method is restrictive. We then extend QSC by systematically extrapolating the weights to get new (or unseen) query's weight. We demonstrate the performance of our method on a large corpus of more than five million words.

2. DATASETS AND EXPERIMENTAL SETTING

We first summarize the experimental framework we use throughout the paper. We consider three different types of data sets in English. They are aimed at quantitatively evaluating the performance of the matching methods, as well as demonstrating the generalization capabilities of learning schemes. Data sets can be summarized as:

- **Calibrated Data (CD):** To study the effect of font and size variations, we consider a calibrated data set of word images. They are generated by rendering the text and passing through a document degradation model [27]. Intensity of degradation is characterized using a scalar, and is used for computing the probability of degradation for the boundary pixels. We consider two subsets of this data set, CD1 and CD2. The set CD1 consists of 1000 words in multiple fonts and sizes. All images are equivalent to words typeset in 8pt to 15pt, and scanned at 300 dpi. CD2 is similar to CD1 but has higher amount of degradation.
- **Real Annotated Data (RD):** This data set consists of a set of words with their ground truth (text) associated to it. This data set is built out of 765 pages from scanned books, publicly available on the web [1]. All the words in these pages are manually annotated for conducting experiments and evaluating the performance. There are a total of 162,188 word images in this collection from four books which vary in fonts.
- **Unannotated Data (UD):** In addition to the completely annotated data sets mentioned above, we also consider a data set of 5,870,486 words which come out of scanned books. Since the data set is not ground truth-ed, it can be used primarily to evaluate the precision for selected queries and not the recall.

Examples from these data sets are shown in Figure 2. The examples demonstrate how the same word appears in all the three different data sets. Note that the words in all the three data sets are not the same. It depends on the content of the books. However, they overlap. We use these datasets for designing more effective similarity scores for the comparison of word images. We use these similarity score for retrieving similar word images. We evaluate the retrieval performance with measures such as (i) Precision, (ii) Recall, (iii) F-Score, (iv) AP, (v) mean of AP. Most of our results are presented using mean of AP.

Precision is the ratio of number of relevant images to the total number of images retrieved, for a particular query. It measures how well a system discards irrelevant results while retrieving. *Recall* is the ratio of number of relevant images retrieved to the total number of relevant images present in

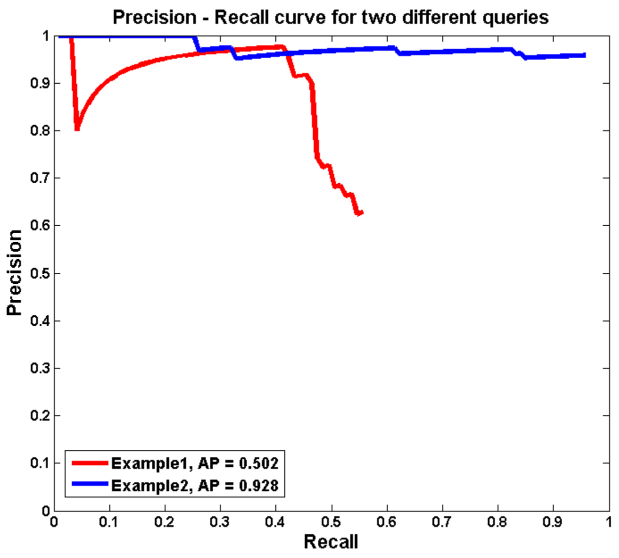
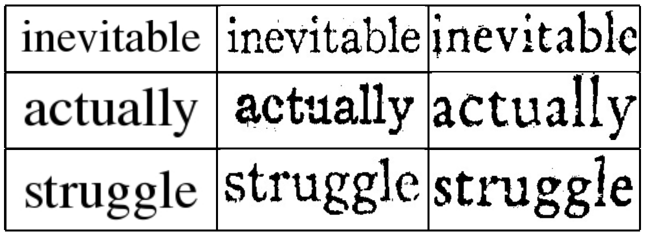


Figure 1: PR-curve on two different queries from the RD dataset.



inevitable	inevitable	inevitable
actually	actually	actually
struggle	struggle	struggle

Figure 2: A comparison of words from three databases. Words in first column are from CD1, Words in second column are from RD and Words in third column are from UD.

the database. It basically measures how well a system finds what the user wants. By changing a matching threshold, one can typically increase recall at the cost of precision. A precision-recall(PR) curve plots how the variation in one affects the other. *FScore* is the weighted harmonic mean of precision and recall and measured in isolation. It combines the precision and recall into one score. Average precision (AP) measures the area under the precision-recall curve. Average precision makes use of both recall and precision and encourages the relevant results to appear at higher ranked positions. Mean of the APs computed for multiple queries gives us *mean average precision (mAP)* [25]. We plot the precision recall graphs of two words (i.e., "even" and "think") in Figure 1. (These are computed in RD as discussed later). It may be seen that the AP can be significantly different for different words. The area under the curve for "even" is 0.502 and that for "think" is 0.928. Also note that, with a given feature set, not all words are equally easy (or difficult) to retrieve. A probable conclusion based on this is that the matching scheme (or distance measure) we use need not be the same for all words.

Measure	DTW (w)					Euclidean (d)				
	w=1	w=2	w=3	w=4	w=5	d=80	d=90	d=100	d=110	d=120
Precision	0.653	0.512	0.432	0.320	0.271	0.594	0.596	0.596	0.598	0.599
Recall	0.805	0.793	0.787	0.731	0.731	0.793	0.793	0.793	0.792	0.787
FScore	0.721	0.622	0.558	0.445	0.395	0.679	0.680	0.680	0.681	0.680
AP	0.853	0.623	0.434	0.374	0.152	0.762	0.760	0.759	0.764	0.765

Table 1: Baseline results on comparing DTW and Euclidean distance on 300 queries from CD dataset.

2.1 Feature Extraction and Matching

We match and retrieve similar word images by comparing their feature descriptors. For the comparison purpose, we extract a set of profile features, which were also used in many of the previous works [12,18,20]. Thus word images are represented as a sequence of feature vectors. *Note that the objective of this work is not to propose any new set of features*, and we refrain from attempting that here. The features we use are (i) upper word profile (ii) lower word profile (iii) projection profile (iv) transition (black-white) profile. Feature extraction is summarized pictorially in Figure 3. More details can be seen in the previous works [12,18,20]. For each of the "query" word images, we compute the nearest K word images and compute the performance measures mentioned above.

All these profiles/features are computed on non-overlapping sliding windows (or a vertical strip) of w pixels. Prior to the feature computation, all blank vertical strips are removed. Since words can be of different length, the feature vector sequences can be of different length. This makes DTW a natural choice for comparison (and computing similarity). However, Euclidean and similar distances (L^P norms defined over a vector space) have many advantages in scalability and learning distance functions. We compute a fixed length representation (say of dimensions d) by varying w for each image.

Given a query image, we compute the distance of the query image from all the database images and find the nearest K matches. When query is text, an image is rendered first and the feature vectors are computed corresponding to the query word. Similar query processing was used in [3,12]

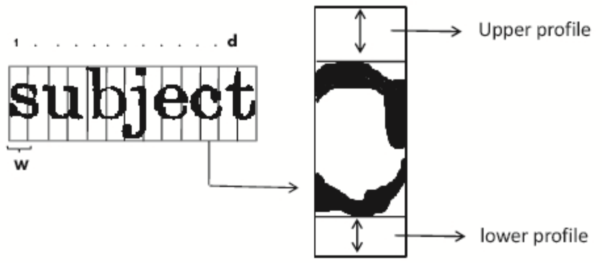


Figure 3: A pictorial representation of how profiles are computed and represented as a feature sequence.

3. DTW VS FIXED LENGTH MATCHING

We first compare DTW and Euclidean distance based matching schemes. For the DTW, we vary w and create feature sequences of varying length. For the Euclidean distance based matching, we vary the dimensionality — d. We conduct our experimental study on CD1 and CD2 together. Results of this experiment are presented in Table 1. We compare the performance of DTW and Euclidean distance while computing the nearest K neighbors. For these experiments, K was set as 50. We observed that DTW gives superior results to Euclidean distance in all these cases. Possibly similar observations were also shown in some of the previous works. (However, this is not always true as we show later in this section.) The success of this had been attributed to the fact that a dynamic warping can align well under the font/style variation of characters (either in print or in handwriting) and thus provide a better estimate of the dissimilarity.

However, it is reasonable to see that, when the number of fonts increases, both DTW and Euclidean distance based matching result in poorer performance. For example, if the mAP in a data set of 10 fonts in 0.829, the mAP in a data set of 50 fonts is only 0.453. We use our calibrated data sets for this study. This can be primarily attributed to the features/descriptors we use. However, this could be a general concern in many document image analysis tasks. (Interested readers may also see how character recognition can be poor when the number of fonts increase [6]). Figure 4 summarizes the effect of number of fonts on the retrieval.

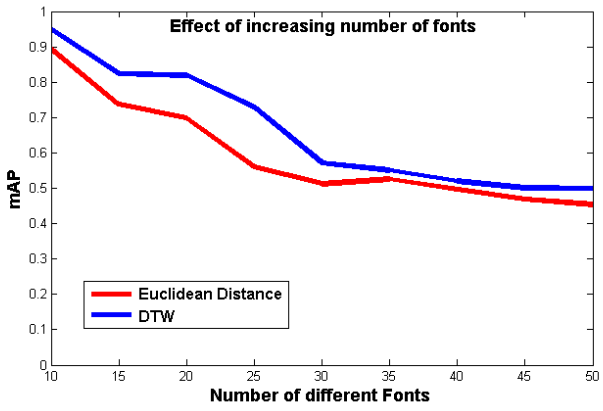


Figure 4: This graph shows how mAP changes when number of fonts increases.

We further conduct experiments to closely compare the performance of DTW and Euclidean distance based matching in a wider problem setting. When the data is degraded, DTW is getting affected more severely than Euclidean distance. The results can be seen in Table 2. This can be partly attributed to the features and partly to the matching scheme. Figure 5 shows that the observation is consistent.

i.e., with increase in degradation, DTW loses its advantages over Euclidean in the performance. In the rest of the paper, we now investigate, how a fixed length representation can be matched better by learning an appropriate distance function. This is primarily motivated by the need to build a scalable, efficient and robust document image retrieval system.

Dataset	DTW	Euclidean
CD2	0.421	0.471
RD	0.778	0.817

Table 2: Results(mAP) on Degraded dataset CD2 and RD with 300 queries in each.

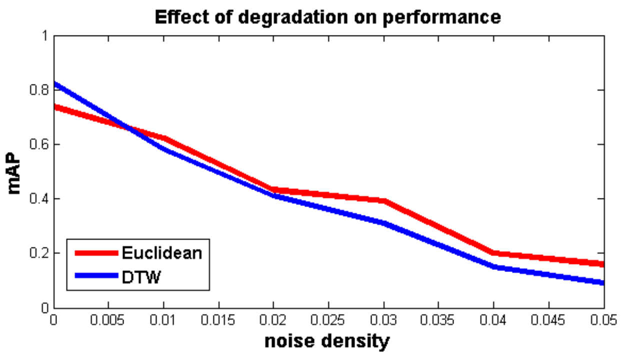


Figure 5: Performance comparison of DTW and Euclidean distance based matching with degradations in data

DTW is computationally intensive compared to Euclidean distance as shown in Figure 6. For example, DTW takes approximately 16 secs while Euclidean distance takes only 0.3 secs to find the K nearest match in a database of 162,188 words. To match sequences of length M and N, DTW needs to do $O(MN)$ operations while using Euclidean distance, one needs to do $O(d)$ operations. For the comparison, we set the dimensionality for the Euclidean distance matching as the average of the sequence lengths in the database (in our case 110) and the algorithm have been implemented in C++ and run on a 2Ghz Dual core machine running Linux. It is well known that the fixed length distance based matching scheme is scalable to large collection. DTW based matching is thus not suitable for such an indexing. There are many useful data structures and indexing schemes proposed in literature to address this problem. Interested readers may find specific details at [5, 8]

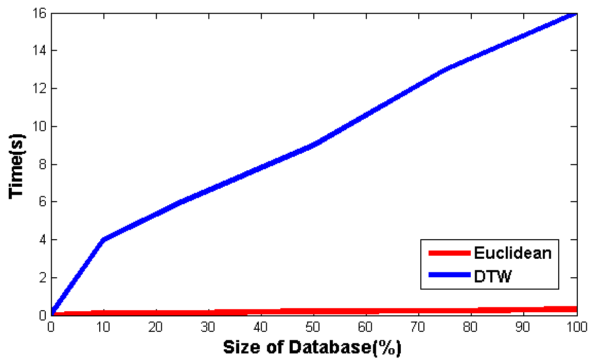


Figure 6: This graph compares average time taken by two different methods, DTW and Euclidean Distance, per query on increasing database size.

4. RETRIEVAL BY LEARNING QUERY SPECIFIC CLASSIFIER

We are interested in the following problem: given a query word image, we would like to retrieve all similar word images from a database of word images, in a ranked manner. This retrieval problem can be understood as the design of an appropriate classifier for a given query. If one considers a Nearest neighbor classifier, then the retrieved set is the ranked list of word images sorted in the increasing order of distance $d(\mathbf{f_i}, \mathbf{q})$, where $\mathbf{q}$ is the query word image and f_is are the database images. In the simplest form, this could be an Euclidean distance $\sqrt{\sum_j (f_i^j - q^j)^2}$. Where q^j is the jth feature of the feature vector representation. However, this may not be the ideal manner in which we can design the classifier. This is because: (i) this does not consider the information that f_i is in fact a fixed-length-representation built out of a sequence representation (ii) there exists a latent language model which generates these sequences and assign different probabilities for generation of words (iii) the classifier (or distance function) used to rank the word images could be different for each query. This necessitates exploring for a query specific classification scheme. Mahalanobis distance could have taken care of some of the aspects related to correlation of feature vectors.

We use a weighted Euclidean distance for matching word images and retrieving relevant documents

$$d'(\mathbf{f_i}, \mathbf{q}, \mathbf{w}) = \sqrt{\sum_j w^j (f_i^j - q^j)^2}$$

where $\mathbf{w} = [w^j]^T$ is a weight vector. We learn $\mathbf{w}$ from examples. Such learning per query has been used in relevant feedback literature for retrieving documents [23] as well as images [2, 7, 9]. In general, in these two cases, the weight vector is learnt using the user feedback (User labels each of the retrieved results as positive or negative. Thus the user refines the retrieved results by continuous dialogue). During retrieval, in each of the iterations t, weight is typically refined (with any of the below equations) as:

$$w^j(t+1) \leftarrow w^j(t) + \frac{1}{\sigma_+^j} \tag{1}$$

$$w^j(t+1) \leftarrow w^j(t) + \frac{\sigma_-^j}{\sigma_+^j} \tag{2}$$

where σ_+^j is the standard deviation of the jth feature of positive examples. Similarly σ_-^j is for negative examples. In the first case, weights are learnt only from the positive examples, while in the second, weights get refined based on positive and negative examples. Both these (and variants of

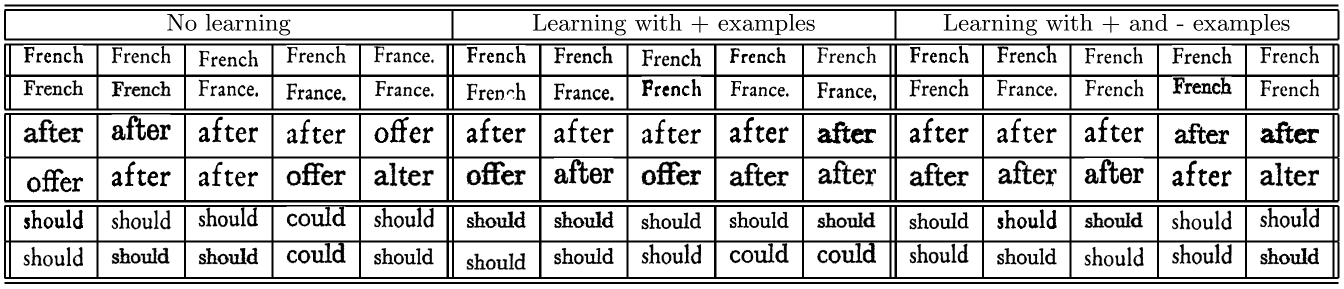

No learning
Learning with + examples
Learning with + and - examples
French French French French France. French French French French French French French French French French
French French France. France. France. French France. French France. France, French France. French French French
after after after after offer after after after after after after after after after after
offer after after offer alter offer after offer after after after after after after alter
should should should could should should should should should should should should should should should
should should should could should should should should could could should should should should should

Figure 7: Example of images retrieved with and without learning, shown according to the rank in row major

these) are popular in relevant feedback literature [28]. However, our setting is some what different from the relevance feedback methods. We are not looking for users to *interactively* retrieve word images. We are interested in learning a query specific classification/ranking scheme from a set of training examples, and directly use them on the test or new data.

Learning using Equation 2 is mildly different from a normal setting in our case. We partition database images as positive, negative and neutral examples. Positive examples are the true exemplars. Neutral ones are the examples, which share characters at certain positions. For example, for a query word of "search", the word "series" is considered as neutral, since both of them share "se". Since we are using fixed length representations built out of profile features, this minimizes contradictions. Without introduction of a neutral class, learning becomes difficult. All examples which are neither positive nor neutral are negative examples.

4.1 Retrieval with learnt distance functions

Our retrieval approach learns the query specific distance function for optimizing the rank. The iterative method described in the previous section can be done in closed form for a completely annotated data set. This also removes the necessity of having a user to refine the results. Learning distance functions from labeled examples have received considerable attention in the recent past [4, 26].

Table 3 presents a quantitative comparison of performance enhancement with learning. We can see that learning with Equations 1 or 2 are almost similar. Figure 7 shows qualitative examples of the retrieval with and without learning. It can be seen that visually similar looking images are retrieved with no learning, and with learning a better set is obtained.

Datasets	No Learning	QSC with Eq 1	QSC with Eq 2
CD	0.737	0.946	0.944
RD	0.817	0.930	0.939

Table 3: Results(mAP) on two Dataset with 300 queries in each.

Some of the state of art approaches learn a Mahalanobis distance metric for the k- nearest neighbor classification by semidefinite programming [26]. The metric is learnt such that top k- neighbors always belong to the the same class while examples from the different classes are separated by a larger margin. Some of others learn a distance function for each training image as a combination of elementary distances between patch-based visual features. Then apply these to classification of novel images [10]. There is another approach where the weights are the ratios of standard deviations of the complete dataset to the the images selected by user, but in our case we do not use neutral examples and also their method is restricted towards new unseen queries. Learning has been used in document retrieval, one of such method is [12]. In their work they also aim to search for keywords given by the user in a large collection of digitized printed historical document and retrieve a better set with the help of user feedback but again their method doesn't use any distance function learning.

5. EXTRAPOLATION OF QUERY SPECIFIC CLASSIFICATION

In the previous section we presented methods which allow us to learn better distance functions for a given query. However, this method is limited to a finite set of query images. In a generic setting of document image retrieval, user would like to give an arbitrary query and retrieve the most relevant document images based on word similarity computation. We support textual queries for this purpose. We enable users to provide user queries (rather than already existing word images) by rendering queries and generating word images. Even then the retrieved results need not be accurate because the distance functions are not designed for these queries. We would like to extrapolate the learnt distance functions into a novel setting.

It is known that when the database is diverse, retrieval performance deteriorates. This is due to the fact that database contains multiple (possibly unknown) fonts and styles. To address this issue we extrapolate the learnt distance function to novel content and style space without any user involvement. The weight vector learnt for similar queries (where substrings are same), will be similar in dimensions corresponding to the common substrings. We use this fact and synthesize new weight vectors which share substrings with the query word in the training database. To enhance the performance in a heterogeneous style collection we retrieve words based on multiple distance functions and generate a final list by merging the multiple set.

In real life database we have variations in style and content. However, the number of possible variations and styles (say fonts, sizes) is far less compared to the content variations. It has been shown in the past that given a collection, style and content can be separated [24] with the help of expectation maximization (EM) algorithm. However, a purely unsupervised distance function learning in such a framework will be computationally intensive. That is why we resort to the method of learning appropriate distance function from the training dataset and extend to the test dataset.

Here, distance functions are learnt for each word in the RD dataset using the method described in Section 4. This is done in closed form using the groundtruth word images in the training set. Since the feature descriptors also depend on the style variations of the text we do style specific learning. Hence for each word image w_i and style s_k we have a word weight vector $W_w(s_k, w_i)$. During retrieval, from the unseen textual word query, word images with the same text as query are synthesized in different styles. Distance functions for these synthetic word images are computed from already learnt weight vectors by extrapolation. We achieve this by disintegrating the learnt weight vectors for words into learnt weights vectors for subwords during training. And then by using these subword weight vectors to compute weight vectors for unseen words.

Since the feature vectors are extracted using a sliding window, the descriptor depends on the width of the image. Therefore, each subword's contribution to the word descriptor (and eventually word weight vector) is directly proportional to the width of that subword in the word. A subword can be a pair or a sequence of characters or just a character, here we refer to character as subword. For any style we know the approximate relative width for each subword in the word. Now, we divide the word weight vector $W_w(s_k, w_i)$ according to relative widths of the subwords to get weight vectors for each of them. These subword weight vectors are then scaled to have a constant dimension l_d. For style s_k, final weight vector for a subword l_j, $W_l(s_k, l_j)$, is computed as:

$$W_l(s_k, l_j) = \sum_{i \in W_{l_j}} \frac{W_{wl}(s_k, l_j, w_i)}{|W_{l_j}|} \tag{3}$$

where W_{l_j} is set of all words that contain l_j and $W_{wl}(s_k, l_j, w_i)$ is the weight vector for subword l_j when it occurs in word w_i and style s_k. We call this *content specific disintegration* of word weight vector or *construction* of subword weight vector. This is demonstrated in Figure 9(a). We take an example of a word "above" from RD dataset. It is shown how with the help of query specific learning method we learn a weight vector for this word image and then disintegrate it to get subword weight vectors. Please note that the subword weight vectors are subdivisions taken from word weight vector based on the relative subword width, which is approximate and not exact. Inspite of this, our method achieves good precision that shows its robustness. The projected subword weight vectors, $W_{wl}(s_k, l_j, w_i)$, are shown at the end of the pipeline. Such weight vectors are computed for each subword-style combination across all the word images in the dataset. And the final subword weight vector computed using Equation 3.

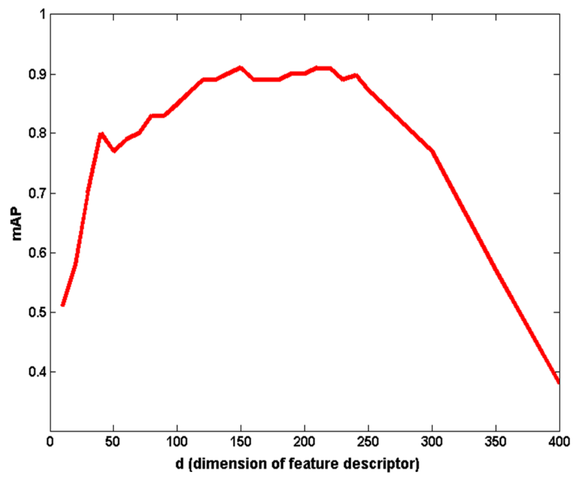


Figure 8: Graph shows how mAP changes when dimension of feature vectors increases.

Now as the output of the above training phase we have style and content dependent weight vectors for each style-subword combination. Essentially, we learn the matrix,

$$M(k, j) = W_l(s_k, l_j), \quad k = 1, 2...|S| \quad and \quad j = 1, 2...|C|$$

where S is set of all styles present in the database and C is set of all characters or subwords.

This matrix, M, of weight vectors is used to extrapolate the learning from training data to get weight vector for any new/unseen word. Specifically, features are extracted from the synthetic test word images for each style. Then learnt weight vectors for each of the subwords in the test word image are mapped from l_d to a new dimension. These new lengths are directly proportional to the relative lengths of the corresponding subwords and chosen such that their sum is equal to d. These are appended to get the d dimensional weight vector of the query word image for that particular style. This process of extrapolation for retrieval is shown in Figure 9(b). As an example we take a textual query "close" from the UD dataset.

We perform two experiments to evaluate our query specific learning with extrapolation method. In the first one, we test our method by retrieving from a dataset which is same as the training dataset. For this purpose, we train using CD, RD datasets and retrieve from the respective dataset. During evaluation, we selected 300 queries from each of the datasets and retrieved 30 images per query with the smallest distance. The mean of average precisions are reported in first and second row of the Table 4.

In the second experiment, we test our method by retrieving from a dataset which is different from the training dataset. For this, we train using RD dataset and retrieve from UD dataset. Since UD dataset is unannotated, we do not know the total number of true positives and therefore cannot compute average precision. Therefore, we compute mean preci-

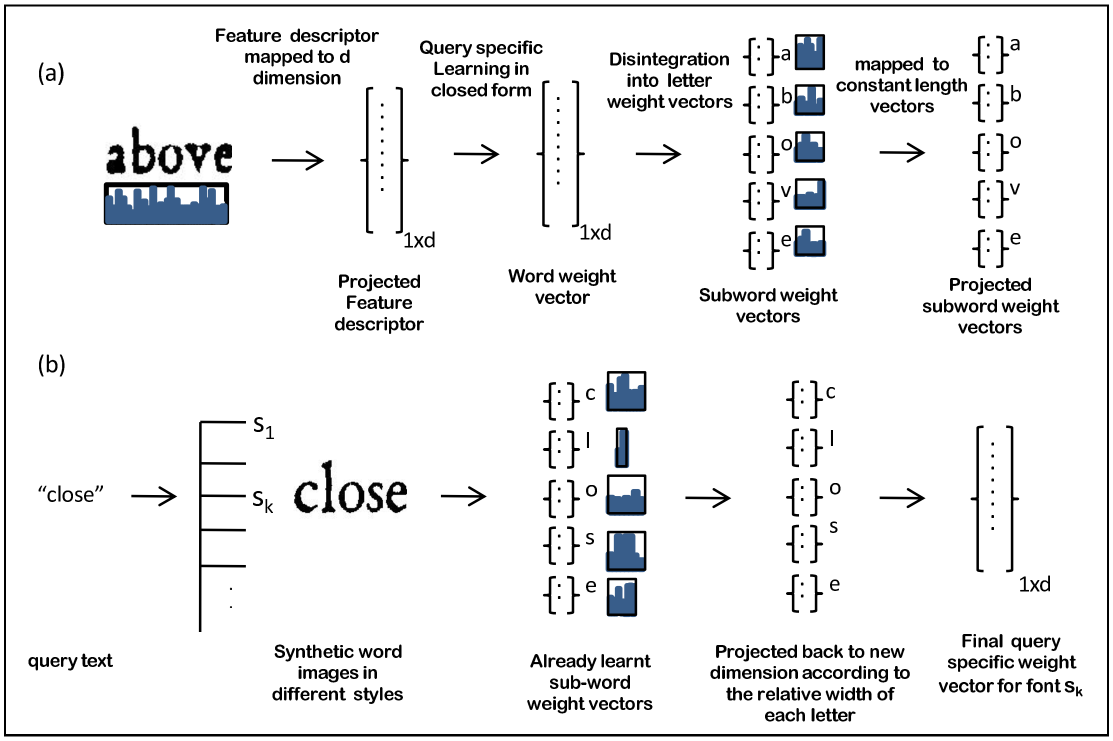

(a)
Feature descriptor mapped to d dimension
Query specific Learning in closed form
Disintegration into letter weight vectors
mapped to constant length vectors
above
1xd
1xd
Projected Feature descriptor
Word weight vector
Subword weight vectors
Projected subword weight vectors
(b)
"close"
s1
sk
close
1xd
query text
Synthetic word images in different styles
Already learnt sub-word weight vectors
Projected back to new dimension according to the relative width of each letter
Final query specific weight vector for font sk

Figure 9: (a) The pipeline shows how a weight vector is learnt for each subword during training in RD dataset and (b) The pipeline shows how a weight vector is generated by extrapolation for an unseen query which is later used for retrieval.

sion as a performance measure which is shown in last row of the Table 4. We use 25 selected queries for this experiment and retrieved 30 images per query. We also show its qualitative results on three queries in Figure 10. Variation in the top 10 retrieved results shows that our method is robust towards variation in fonts and styles. In both the experiments our method outperforms the other two and also shows good generalization.

Dataset	Measure	DTW	Euclidean	QSC with extrapolation
CD	mAP	0.853	0.764	0.902
RD	mAP	0.778	0.817	0.923
UD	mP	0.890	0.915	0.955

Table 4: Comparative results of extrapolation on CD, RD and UD datasets.

The above method of extrapolation involves projecting a feature descriptor for word images to lower or higher dimension feature space (d dimension). Projecting to a very different feature dimension may lead to loss of important information. So it is important to set the value of d appropriately. We conduct an experiment to observe the effect of d on the retrieval performance. Figure 8 shows how mAP varies with d. From the plot, it can be observed that the performance is stable for a good range: $d = 110$ to $d = 270$. The range of d depends on the database which means that it can be different for different languages.

The extrapolation method presented in this section enables generalization of the learning done from training data. It makes it possible to generate weight vector for any unseen word.

6. ACKNOWLEDGMENTS

This work is supported by Ministry of Communication and Information Technology, Government of India.

7. CONCLUSIONS

In this paper, we argue that a matching scheme based on fixed length representation can be learnt to enhance the word image matching module. We demonstrate that a query specific classification algorithm can be designed for retrieval of word images. This can be done in a computationally efficient manner. Our future work includes the development of a scalable system which can retrieve word images by implicitly using language models.

8. REFERENCES

[1] Digital library of india. http://dli.iiit.ac.in/.
[2] S. Aksoy, R. M. Haralick, F. A. Cheikh, and M. Gabbouj. A weighted distance approach to relevance feedback. In *ICPR*, pages 4812–4815, 2000.
[3] A. Balasubramanian, M. Meshesha, and C. V. Jawahar. Retrieval from document image collections. In *Document Analysis Systems*, pages 1–12, 2006.

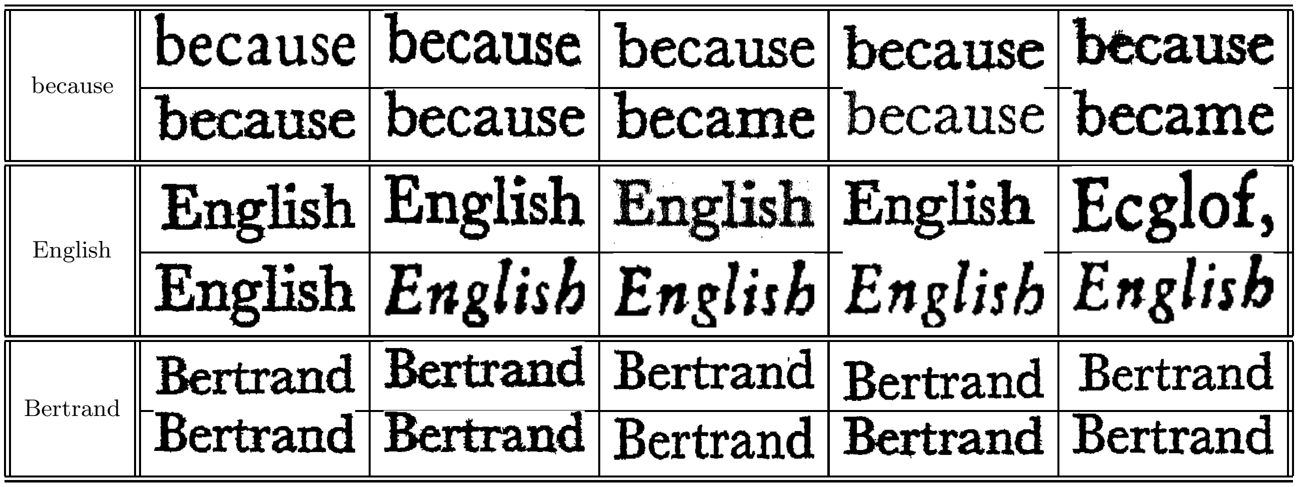

because
because because because because because
because because became because became
English
English English English English Ecglof,
English English English English English
Bertrand
Bertrand Bertrand Bertrand Bertrand Bertrand
Bertrand Bertrand Bertrand Bertrand Bertrand

Figure 10: Examples of word images retrieved from unannotated database of more than 5M words. First column shows the query word. Some of the retrieved results are shown. For the top-20 retrieval, mean of precision is 0.936

[4] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6:937–965, 2005.

[5] C. Böhm, S. Berchtold, and D. A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys.*, 33(3):322–373, 2001.

[6] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *VISSAPP (2)*, pages 273–280, 2009.

[7] T. Deselaers, R. Paredes, E. Vidal, and H. Ney. Learning weighted distances for relevance feedback in image retrieval. In *ICPR*, pages 1–4, 2008.

[8] D. S. Doermann. The indexing and retrieval of document images: A survey. *Computer Vision and Image Understanding*, 70(3):287–298, 1998.

[9] A. Franco, A. Lumini, and D. Maio. A new approach for relevance feedback through positive and negative samples. In *ICPR (4)*, pages 905–908, 2004.

[10] A. Frome, Y. Singer, and J. Malik. Image retrieval and classification using local distance functions. In *NIPS*, pages 417–424, 2006.

[11] T. Hong, T., and J. J. Hull. Algorithms for postprocessing ocr results with visual inter-word constraints. In *ICIP*, 1995.

[12] T. Konidaris, B. Gatos, K. Ntzios, I. Pratikakis, S. Theodoridis, and S. J. Perantonis. Keyword-guided word spotting in historical printed documents using synthetic data and user feedback. *IJDAR*, 2007.

[13] A. Kumar, C. V. Jawahar, and R. Manmatha. Efficient search in document image collections. In *ACCV (1)*, pages 586–595, 2007.

[14] V. Lavrenko, T. M. Rath, and R. Manmatha. Holistic word recognition for handwritten historical documents. In *DIAL*, pages 278–287, 2004.

[15] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999.

[16] Y. Lu, C. L. Tan, W. Huang, and L. Fan. An approach to word image matching based on weighted hausdorff distance. *ICDAR*, 2001.

[17] R. Manmatha, C. Han, and E. M. Riseman. Word spotting: A new approach to indexing handwriting. In *CVPR*, pages 631–637, 1996.

[18] M. Meshesha and C. V. Jawahar. Matching word images for content-based retrieval from printed document images. *IJDAR*, 11(1):29–38, 2008.

[19] V. Rasagna, A. Kumar, C. V. Jawahar, and R. Manmatha. Robust recognition of documents by fusing results of word clusters. In *ICDAR*, 2009.

[20] T. M. Rath and R. Manmatha. Word image matching using dynamic time warping. In *CVPR (2)*, pages 521–527, 2003.

[21] T. M. Rath and R. Manmatha. Word spotting for historical documents. *IJDAR*, 9(2-4):139–152, 2007.

[22] J. Rodriguez-Serrano, F. Perronnin, J. Llados, and G. Sanchez. A similarity measure between vector sequences with application to handwritten word image retrieval. *CVPR*, 2009.

[23] Y. Seiji and O. Takashi. Document retrieval using relevance feedback. *CICSJ Bull*, 21(1):32–36, 2004.

[24] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Computing*, 12(6):1247–1283, 2000.

[25] C. J. van Rijsbergen. *Information Retrieval.* Butterworth, 1979.

[26] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 2005.

[27] Q. Zheng and T. Kanungo. Morphological degradation models and their use in document image restoration. In *ICIP (1)*, pages 193–196, 2001.

[28] X. S. Zhou and T. S. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia System*, 8(6):536–544, 2003.