Handwritten Word Spotting with Corrected Attributes

Jon Almazán*[1], Albert Gordo[2,1], Alicia Fornés[1], and Ernest Valveny[1]

[1]Computer Vision Center, Universitat Autònoma de Barcelona, Barcelona, Spain
[2]LEAR, INRIA, Grenoble, France

Abstract

We propose an approach to multi-writer word spotting, where the goal is to find a query word in a dataset comprised of document images. We propose an attributes-based approach that leads to a low-dimensional, fixed-length representation of the word images that is fast to compute and, especially, fast to compare. This approach naturally leads to an unified representation of word images and strings, which seamlessly allows one to indistinctly perform query-by-example, where the query is an image, and query-by-string, where the query is a string. We also propose a calibration scheme to correct the attributes scores based on Canonical Correlation Analysis that greatly improves the results on a challenging dataset. We test our approach on two public datasets showing state-of-the-art results.

1. Introduction

This paper addresses the problem of multi-writer word spotting. The objective of word spotting is to find all instances of a given word in a potentially large dataset of document images. This is typically done in a query-by-example (QBE) scenario, where the query is an image of a handwritten word and it is assumed that the transcription of the dataset and the query word is not available. In a multi-writer setting, the writers of the dataset documents may have completely different writing styles than the writer of the query. Multi-writer word spotting can therefore be seen as a particular case of semantic content based image retrieval (CBIR), where the classes are very fine-grained – we are interested in *exactly* one particular word, and a difference of only one character is considered a negative result – but also contain a very large intra-class variability – different writers may have completely different writing styles, making the same word look completely different (*cf*. Fig. 4). This huge variability in styles makes this a much more difficult problem than typeset or single-writer handwritten word spotting.

Because of this complexity, most popular techniques are based on describing word images as sequences of features of variable length and using techniques such as Dynamic Time Warping (DTW) or Hidden Markov Models (HMM) to classify them. Variable-length features are more flexible than feature vectors and have been known to lead to superior results in difficult word-spotting tasks since they can adapt better to the different variations of style and word length [7, 9, 22, 24, 25]. Unfortunately, this leads to two unsatisfying outcomes. First, due to the difficulties of learning with sequences, many supervised methods cannot perform out of vocabulary (OOV) spotting, *i.e*., only a limited number of keywords, which need to be known at training time, can be used as queries. Second, because the methods deal with sequences of features, computing distances between words is usually very slow at test time. As a consequence, approaches such as exhaustive sliding window search are not feasible, and the words in the dataset documents need to be segmented [7, 22, 24, 25]. In many real scenarios this is error prone and time consuming, and can be just unfeasible if the dataset is large enough.

Indeed, with the steady increase of datasets size there has been a renewed interest in compact, fast-to-compare word representations. Recent examples of this are the work of Rusiñol *et al*. [28], where word images are represented with SIFT descriptors aggregated using the bag of visual words framework [4], or the work of Almazán *et al*. [1], which uses HOG descriptors [5]. In both cases, the fixed-length descriptors are very fast to compare and can be used together with a sliding window approach on non-segmented documents. Although the results on simple datasets are encouraging, the authors argue that these fixed-length descriptors do not offer enough flexibility to perform well on more complex datasets and especially in a multi-writer scenario.

*J. Almazán, A. Fornés, and E. Valveny are partially supported by the Spanish projects TIN2011-24631, TIN2009-14633-C03-03, TIN2012-37475-C02-02, 2010-CONE3-00029, by the EU project ERC-2010-AdG-20100407-269796 and by a research grant of the UAB (471-01-8/09). The authors would also like to thank M. Rusiñol and V. Frinken for their contribution to the paper.

Through this paper we follow these recent works [1, 28] and focus on fixed-length representations, which can potentially be used in a segmentation-free context. In particular, we adopt the Fisher vector (FV) [29] representation computed over SIFT descriptors extracted densely from the word image. The Fisher vector can be understood as a bag of words that also encodes higher order statistics, and has been shown to be a state-of-the-art encoding method for several computer vision tasks such as image classification and retrieval [3]. Yet, as argued by other authors [1, 28] , descriptors such as the FV do not *directly* capture all the flexibility needed in a multi-writer setting: although the results on a single-writer dataset are competitive, the accuracy dramatically drops when using more challenging datasets with large variations in style. We postulate that leveraging supervised information to learn the similarities and differences between different writing styles is of paramount importance to compensate for the lack of flexibility of the fixed-length representations, and that not exploiting this information is one of the main causes of their subpar performance.

In this paper we propose to use labeled training data to learn how to embed our fixed-length descriptor in a more discriminative, low-dimensional space, where similarities between words are preserved independently of the writing style. However, as opposed to other methods, our aim is not to learn models for particular keywords, but to learn *what* makes words and letters unique independently of their writers' style. Indeed, we believe that learning robust models at the word level is an extremely difficult task due to the intrinsic variation of writing styles, and its adaptation to new, unseen words at test time usually yields poor results. Instead, we believe that a successful approach should be able to transfer and share information between different instances at training time. The use of attributes is, arguably, the most popular approach to achieve these goals.

As our first contribution, we propose an embedding approach that encodes word strings as a pyramidal histogram of characters – which we dubbed PHOC –, inspired by the bag of characters string kernels used for example in the machine learning and biocomputing communities [16, 17]. In a nutshell, this binary histogram encodes whether a particular character appears in the represented word or not. By using a spatial pyramid we add some coarse localization, *e.g.*, this character appears on the first half of the word, or this character appears in the last quarter of the word (see Fig. 1). The histogram can also encode bigrams or other combinations of characters. These histograms are then used as a source of attributes, see Fig. 2. During the learning of these attributes we use a wide variety of writers and characters, and so the adaptation to new, unseen words is almost seamless. A naive implementation of this attributes representation greatly outperforms the direct use of FVs. A very similar string embedding has been simultaneously proposed in [27]. However, in their case, the representation is used in a label embedding context, and not as a source of attributes.

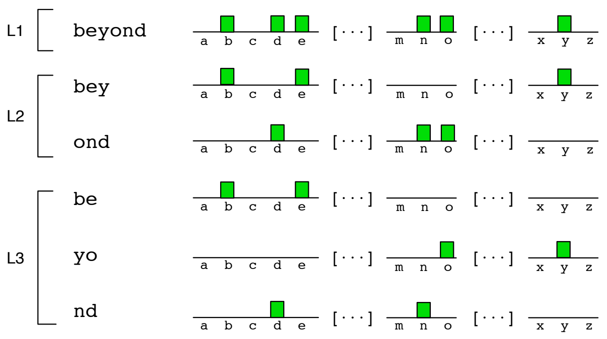

Figure 1. PHOC histogram at levels 1, 2, and 3. The final PHOC histogram is the concatenation of these partial histograms.

We found that accurately calibrating the attribute scores can have a large impact in the accuracy of the method. Although Platts scaling makes a significant difference over non-calibrated scores, one drawback is that each score is calibrated independently of the others. We believe that calibrating the scores jointly can lead to large improvements, since the information of different attributes is shared. This is particularly true in the case of pyramidal histograms, where the same character may be simultaneously represented by various attributes depending on its position inside the word. This motivates our second contribution, a scheme to calibrate all the attribute scores jointly by means of Canonical Correlation Analysis (CCA) and its kernelized version (KCCA), where the main idea is to correlate the predicted attribute scores with their ground truth values. This calibration method can noticeably outperform the standard Platts scaling while, at the same time, perform a dimensionality reduction of the attribute space. We believe that the uses of this calibration scheme are not limited to word image representation and can also be used in other attribute-based tasks.

Finally, as our third contribution, we note that this attribute-based framework naturally bridges the gap between "query-by-(visual) example" (QBE) and "query-by-(typed) string" (QBS), a very related problem where the input query is a string, without any need to explicitly synthesize queries [26] or to learn string models [9]. Like in [23], we learn a joint representation for word images and text. However, contrary to [23], where they use statistical models and they need to estimate the probability distribution online for a given query, the PHOCs extracted from a string and PHOCs predicted from images lie in the same subspace and can be compared directly – particularly after the CCA calibration – simply using the dot-product as a similarity measure.

The rest of the paper is organized as follows. In Section 2 we review the literature on fixed-length representations for word images and describe our baseline FV representation as well as the proposed attributes-based representation. In Section 3 we describe the proposed calibration system. Section 4 deals with the experimental validation of our approach. Finally, Section 5 concludes the paper.

2. Word Representation

In this section we describe how we obtain the representation of a word image. First we review fixed-length word image representations and introduce the FV as our reference representation. Then, in Section 2.1, we show how to use labeled training data to embed these FV representations in a more discriminative and low-dimensional space by means of attributes. Although we use the FV as the reference representation, the approach can be directly applied to other representations such as HOGs or bag of words.

Early examples of holistic representations are the works of Manmatha *et al*. [18] and Keaton *et al*. [14]. In [18], a distance between binary word images is defined based on the result of XORing the images. In [14], a set of features based on projections and profiles is extracted and used to compare the images. In both cases, the methods are limited to tiny datasets. A more recent work [20] exploits the Fisher kernel framework [13] to construct the Fisher vector of a HMM. This representation has a fixed length and can be used for efficient spotting tasks, although the paper focuses on only 10 different keywords. Finally, recent approaches that are not limited to keywords can be found in [10, 28, 1]. Gatos *et al*. [10] perform a template matching of block-based image descriptors, Rusiñol *et al*. [28] use an aggregation of SIFT descriptors into a bag of visual words to describe images, while Almazán *et al*. [1] use HOG descriptors [5] combined with an exemplar-SVM framework. These fast-to-compare representations allow them to perform word spotting using a sliding window over the whole document without segmenting it into individual words.

Here we adopt a similar approach and represent word images using the FV framework. On preliminary experiments, we observed the FV representation to greatly outperform bag of words and HOG representations. SIFT features are densely extracted from the image and aggregated into a FV representation using a vocabulary of 16 Gaussians. To (weakly) capture the structure of the word image, we use a spatial pyramid of 2×6 leading to a final descriptor of approximately $25,000$ dimensions.

2.1. Supervised Word Representation with PHOC Attributes

One of the most popular approaches to perform supervised learning for word spotting is to learn models for particular *keywords*. A pool of positive and negative samples is available for each keyword, and a model (usually a HMM) is learned for each of them. At test time, it is possible to compute the probability of a given word being generated by that keyword model, and that can be used as a score. Note that this approach restricts one to keywords that need to be learned offline, usually with large amounts of data. In [25], this problem is solved by learning a semicontinuous HMM (SC-HMM). The parameters of the SC-HMM are learned on a pool of unsupervised samples. Then, given a query, this SC-HMM model can be adapted, online, to represent the query. This method is not restricted to keywords and can perform OOV spotting. However, the labels of the words where not used during training.

One disadvantage of these approaches that learn at the word level is that information is not shared between similar words. For example, if learning an HMM for a "car" keyword, "cat" would be considered a negative sample, and the shared information between them would not be explicitly used. We believe that sharing information between words is extremely important to learn good discriminative representations, and that the use of attributes is one way to achieve this goal. Attributes are semantic properties that can be used to describe images and categories [6], and have recently gained a lot of popularity for image retrieval and classification tasks [6, 15, 31, 32]. Attributes have also shown ability to transfer information in zero-shot learning settings [15] and have been used for feature compression since they usually provide compact descriptors. The selection of these attributes is commonly a task-dependent process, so for their application to word spotting we should define them as word-discriminative and writer-independent properties.

One straightforward approach is to define character attributes. We define attributes such as "word contains an *a*" or "word contains a *k*", leading to a histogram of 26 dimensions when using the English alphabet[1]. Then, at training time, we learn models for each of the attributes using the image representation of the words (FVs in our case) as data, and set their labels as positive or negative according to whether those images contain that particular character or not (see Figure 2). Remember that we assume labeled training data, and so that information is available. Then, at testing time, given the FV of a word, we can compute its attribute representation simply by concatenating the scores that those models yield on that particular sample. After calibrating the scores (using, *e.g*., Platts scaling), these attribute representations can be compared using measures such as the Euclidean distance or the cosine similarity.

The previous representation is writer-independent, since a good generalization is achieved by using a large number of writers to learn the models. However, it is not word-discriminative: words such as "listen" and "silent" share the same representation. Therefore, we propose to use a

[1]We do not make any distinction between lower-case and upper-case letters, which leads to a case-insensitive representation. It is trivial to modify it to be case-sensitive, at the cost of doubling the number of attributes.

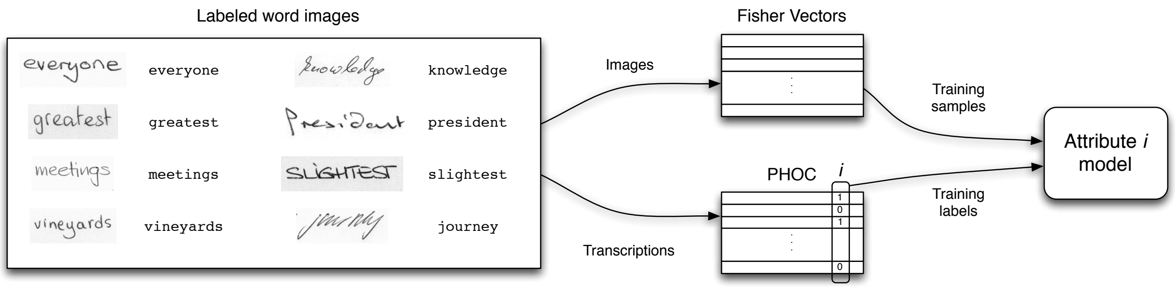

Figure 2. Training process for i-th attribute model. A classifier is trained using the FV representation of the images and the i-th value of the PHOC representation as label.

pyramid version of this histogram of characters, which we dubbed PHOC (see Fig. 1). Instead of finding characters on the whole word, we focus on different regions of the word. At level 2, we define attributes such as "word contains character *x* on the first half of the word" and "word contains character *x* on the second half of the word". Level 3 splits the word in 3 parts, level 4 in 4, etc. In practice, we use levels 2, 3, and 4, leading to a histogram of $(2+3+4)\times 26 = 234$ dimensions. Finally, we also add the 75 most common English bigrams at level 2, leading to 150 extra attributes for a total of 384 attributes. Note how, in this case, "listen" and "silent" have significantly different representations. In the context of an attributes-based representations, the spatially-aware attributes allow one to *ask* more precise questions about the location of the characters, while the spatial pyramid on the image representation allows one to *answer* those questions.

Given a transcription of a word we need to determine the regions of the pyramid where we assign each character. For that, we first define the normalized occupancy of the k-th character of a word of length n as the interval $Occ(k,n) = [\frac{k}{n}, \frac{k+1}{n}]$, where the position k is zero-based. Note that this information is extracted from the word transcription, not from the word image. We remark that we do not have access to the exact position of the characters on the image at training time, only its transcription is available. We use the same formula to obtain the occupancy of region r at level l. Then, we assign a character to a region if the overlap area between their occupancies is larger or equal than 50% the occupancy area of the character, *i.e.*, if $\frac{|Occ(k,n)\cap Occ(r,l)|}{|Occ(k,n)|} \geq 0.5$, where $|[a,b]| = b-a$. This is trivially extended to bigrams or trigrams.

The idea of separating words into characters has been used before (see, *e.g.*, the character HMM models of [7, 8]). However, these approaches have been tied to particular HMM models with sequence features, and so their performance is limited by them. In our case, we propose a broader framework since we do not constrain the choice of features or the method to learn the attributes. Furthermore, one extremely interesting property of this attributes representation is that both image representations (calibrated scores obtained after applying the attribute models to the FV representations) and text representations (PHOC histograms obtained directly from the word transcription) lie in the same space. Indeed, assuming perfect attribute models and calibration, both representations would be identical. This leads to a very clean model to perform query-by-string (QBS, sometimes referred to as query-by-text or QBT), where, instead of having a word image as a query, we have its transcription. Since attribute scores and PHOCs lie in the same space, we can simply compute the PHOC representation of the text and *directly* compare it against the dataset word images represented with attribute scores. Although some other works have approached the QBS problem, solutions usually involve synthesizing image queries [26] or creating model representations specifically designed for QBS tasks [9]. To the best of our knowledge, we are the first to provide an unified framework where we can perform OOV QBE and QBS, as well as to be able to query text datasets using word images without an OCR transcription of the query word.

3. Calibration of scores

Through the previous section we presented an attributes-based representation of the word images. Although this representation is writer-independent, special care has to be put when comparing different words, since the scores of one attribute may dominate over the scores of other attributes. Therefore, some calibration of the attribute scores is necessary. This is particularly true when performing QBS, since otherwise attribute scores are not comparable to the binary PHOC representations.

One popular approach is Platts scaling. It consists of fitting a sigmoid over the output scores to obtain calibrated probabilities, $P(y=1|s) = (1+exp(\alpha s+\beta))^{-1}$, where α and β can be estimated using MLE. In the recent [30], Extreme Value Theory is used to fit better probabilities to the scores and to find a multi-attribute space similarity. Although the similarity measure involves all the attributes, the calibration of each attribute is done individually.

Here we propose to perform the calibration of the scores

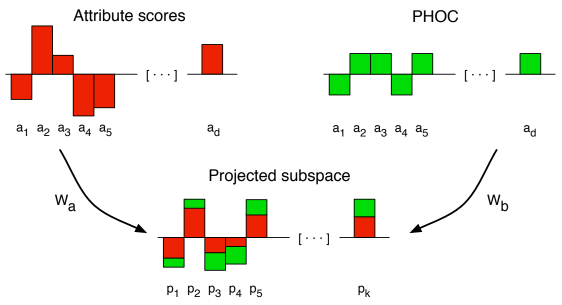

Figure 3. Projection of predicted attribute scores and attributes ground truth into a more correlated subspace with CCA.

jointly, since this can better exploit the correlation between different attributes. To achieve this goal, we make use of Canonical Correlation Analysis to embed the attribute scores and the binary attributes in a common subspace where they are maximally correlated (Fig. 3). CCA is a tool to exploit information available from different data sources, used for example in retrieval [12] and clustering [2]. In [11], CCA was used to correlate image descriptors and their labels, which brought significant benefits for retrieval tasks. We believe this is the most similar use of CCA to our approach. However, while [11] combined images and labels with the hope of bringing some semantic consistency to the image representations, our goal here is to bring the imperfect predicted scores closer to their perfect value.

Let us assume that we have access to N labeled samples for training purposes, where $A \in \mathbb{R}^{D\times N}$ is the D-dimensional attribute score representation of those samples, and where $B \in \{0,1\}^{D\times N}$ is their binary attribute representation. Let us denote with μ_a and μ_b the sample means of A and B. Let us also define the matrices $C_{aa} = \frac{1}{N}(A-\mu_a)(A-\mu_a)'+\rho I$, $C_{bb} = \frac{1}{N}(B-\mu_b)(B-\mu_b)'+\rho I$, $C_{ab} = \frac{1}{N}(A-\mu_a)(B-\mu_b)'$, and $C_{ba} = C'_{ab}$, where ρ is a regularization factor used to avoid numerically ill-conditioned situations and I is the identity matrix.

The goal of CCA is to find a projection of each view that maximizes the correlation between the projected representations. This can be expressed as:

$$\underset{w_a, w_b}{\operatorname{argmax}} \frac{w'_a C_{ab} w_b}{\sqrt{w'_a C_{aa} w_a}\sqrt{w'_b C_{bb} w_b}}. \tag{1}$$

In general we are interested in obtaining a series of projections $W_a = \{w_{a1}, w_{a2}, \dots, w_{ak}\}$, with $w_{ai} \in \mathbb{R}^D$, subject to those projections being orthogonal. This is solved through a generalized eigenvalue problem, $Zw_{ak} = \lambda_k^2 w_{ak}$, with $Z = C_{aa}^{-1} C_{ab} C_{bb}^{-1} C_{ba}$. The k leading eigenvectors of Z form the $W_a = \{w_{a1}, w_{a2}, \dots, w_{ak}\}$ projection vectors that project the scores A into the k-dimensional common subspace. Similarly, we can solve for w_b and arrive to an analogous equation to obtain the $W_b = \{w_{b1}, w_{b2}, \dots, w_{bk}\}$ projection vectors that project the attributes B into the k-dimensional common subspace. Note that this equation has to be solved only once, offline. Since D is the number of attributes, which is usually small (384 in our case), solving the eigenvalue problem is extremely fast. At testing time, given a sample $x \in \mathbb{R}^D$, we can embed it into this space by computing $W'_a(x-\mu_a)$ or $W'_b(x-\mu_b)$, depending on whether x represents attribute scores or pure binary attributes.

This CCA embedding can be seen as a way to exploit the correlation between different attributes to correct the scores predicted by the model. Furthermore, after CCA the attribute scores and binary attributes lie in a more correlated space, which makes the comparison between the scores and the PHOCs for our QBS problem more principled. CCA can also be seen as a label embedding method, similar in spirit to the recent approach of [27]. CCA is also used as a dimensionality reduction tool: we reduce the dimensionality from 384 down to 192-256 dimensions.

One may also note that the relation between the attribute scores and the binary attributes may not be linear, and that a kernelized CCA could yield larger improvements. In this case, we follow the approach of [11]: we explicitly embed the data using a random Fourier feature (RFF) mapping [21], so that the dot-product in the embedded space approximately corresponds to a Gaussian kernel $K(x,y) = exp(-\gamma||x-y||^2)$ in the original space, and then perform linear CCA on the embedded space. In this case, at testing time, a sample x is first embedded using the RFF mapping and then projected using the projections learned with CCA.

4. Experiments

Datasets: We evaluate our method in two public datasets of handwritten text documents: the IAM off-line database[2] [19] and the George Washington (GW) database[3] [22]. The IAM is a large database comprised of $1,539$ pages of modern handwritten English text written by 657 different writers. The document images are annotated at word level and contain the transcriptions of more than $115,000$ words. The GW dataset contains 20 pages of letters written by George Washington and his associates in $1,755$. The writing styles present only small variations and it can be considered as a single-writer dataset. Images are also annotated at word level and contain approximately $5,000$ words. We do not preprocess the images (skew correction, slant correction, smoothing, etc) in the reported experiments. We observed a small gain of 1-2 points by correcting the slant of the words, but we prefer to report results on unprocessed images since preprocessing may not be feasible in a real setup.

Descriptors: We use FV as our base representation. SIFT features are densely extracted from the images, re-

[2]http://www.iam.unibe.ch/fki/databases/iam-handwriting-database
[3]http://www.iam.unibe.ch/fki/databases/iam-historical-document-database/washington-database

duced to 64 dimensions with PCA, and aggregated into a FV which considers the gradients with respects of the means and the variances of the GMM generative model. We use 1M SIFT features extracted from unlabeled words from IAM to learn, offline, the PCA projections as well as the GMM[4]. We use 16 Gaussians for the GMM, which leads to a descriptor of $2 \times 64 \times 16 = 2,048$ dimensions. Since we consider a 2×6 spatial pyramid, this leads to a final histogram of $12 \times 2,048 = 24,576$ dimensions. The descriptor is then power- and L2- normalized. Please *cf*. [29] for more details regarding the construction of FV representations. When computing the attribute representation, we use levels 2, 3, and 4, as well as 75 common bigrams at level 2, leading to 384 dimensions. When learning and projecting with CCA and KCCA, the representations (both score attributes and PHOCs) are first L2-normalized and mean centered. We use CCA to project to a subspace of 192 dimensions. The dimensionality was limited by the number of linearly independent features and the regularization factor. For KCCA, we project into 256 dimensions. Small improvements are achieved by projecting into spaces with more dimensions.

Experimental setup: We split the IAM dataset in 3 partitions at the writer level, containing 40% / 40% / 20% of the writers. Each partition is therefore completely writer independent from the others. We used the first partition to learn the attributes representation, the second partition to learn the calibration as well as for validation purposes, and the third partition for testing purposes. We use the "calibration" partition to validate the parameters of the attribute classifiers, and a small subset of it to validate the calibration (the regularization ρ for CCA, plus the bandwidth γ and the number of random projections for KCCA). The testing set is never observed until testing time. To train the attributes we use a one-versus-rest linear SVM with a SGD solver inspired in the implementation of L. Bottou[5]. At testing time, we use each word of the test dataset as a query and use it to rank the rest of the dataset using the cosine similarity between representations. We do not use stopwords or words that only appear once in the dataset as queries. However, those words are still present on the dataset and act as distractors. As it is standard on retrieval problems, we compute the mean average precision (map) of each query and report the mean of all the queries. We repeat the experiment 3 times with different train and test partitions and average the results. On average, the test folds contain $16,103$ words, of which $5,784$ are used as queries.

We split the GW dataset in a similar way. Since there is no clear writer separation, we split it at word level. Partitions contain 20% / 20% / 60% of the words. The size of the partitions is chosen to ease the comparison with previous works. Unlike in IAM, queries traditionally include stopwords on the GW dataset. Again, experiments are repeated 3 times with different train and test partitions and the results are averaged. On average, the test folds contain $2,847$ words, of which $2,357$ are used as queries.

Experimental results: We first show our results in Table 1 (first two main columns). On both datasets, the attribute representation (even with no calibration) significantly outperforms the FV baseline. This is particularly true on the more challenging IAM. When calibrating the scores, the improvements over the basic FV representation are even more remarkable. Regarding the calibration, we observe significant gains for CCA and KCCA with respect to Platts on the IAM dataset. These gains, however, do not translate into the GW dataset (particularly on the QBS case). We believe there are two reasons for this. First, due to the simplicity of GW, the attribute scores are already very good (notice the 70% QBE map with no calibration at all compared to the 34% on IAM), and so they may not require a complex calibration. Second, the scarcity of the training data (approximately 950 words used for the CCA learning on GW, compared to more than $33,000$ on IAM) is probably leading to suboptimal projections, which would perform better if more training data was available.

It is also interesting to check how the learning performed on the IAM dataset (where all the writers had a "modern" writing style) adapts to a dataset with a very different (250 years old) calligraphic style. We learn the attributes and the Platts weights and CCA and KCCA projections on the IAM dataset as before, and apply it directly to the FV extracted from the GW dataset. Third main column of Table 1 shows these results. We observe how there is an obvious degradation. This is not surprising due to the large differences in style, but also because the attributes learned on the GW are specialized to that particular writing style and so perform better when only that style is present at test time. Still, the results after learning on IAM are reasonable: after projecting with CCA or KCCA, we obtain results comparable to the FV baseline, but using 192 or 256 dimensions instead of $25,000$. The results on QBS show that indeed we are learning attributes correctly and not simply projecting on a different space completely uncorrelated with the transcription of the words. We also note how CCA and KCCA adapt to this "domain shift" much more gracefully than Platts, which actually degrades the results on the QBE task.

Table 2 compares the proposed approach with recent methods on the QBE task. We first compare with our reimplementation of the **exemplar SVM**-based approach of [1], where, at query time, a classifier is trained using the query as a positive sample and the training set as negative samples, and the model is used to rank the dataset. We also compare

[4]We learn the PCA and the GMM on IAM even when performing experiments on GW since this makes the FVs comparable and simplifies some experiments. A gain of 2-3 points is obtained on the GW experiments when learning the PCA and the GMM on it instead of on IAM.

[5]http://leon.bottou.org/projects/sgd

Table 1. First two main columns: retrieval results on the IAM and GW datasets. Last main column: results on the GW dataset when learning is performed solely on the IAM dataset.

	IAM		GW		GW (adapted)	
	QBE	QBS	QBE	QBS	QBE	QBS
FV	14.81	–	63.21	–	63.21	–
Att.	34.32	32.97	69.34	72.32	57.33	34.78
Att. + Platts	45.46	65.01	85.69	90.33	43.69	42.9
Att. + CCA	49.46	70.42	85.85	87.5	63.78	52.84
Att. + KCCA	54.78	71.81	85.63	87.14	61.33	54.29

Table 2. QBE task: comparison with the state-of-the-art.

IAM		GW	
Baseline FV	14.81	Baseline FV	63.21
Exemplar SVM [1]	15.07	Exemplar SVM [1]	65.84
DTW	12.65	DTW [25]	50.0
Character HMM [7, 9]	15.1 / 36.0	SC-HMM [25]	53.0
Proposed (Platts)	45.46	**Proposed (Platts)**	**85.69**
Proposed (KCCA)	**54.78**	**Proposed (KCCA)**	**85.63**

with a classical **DTW** approach using variable length features. On IAM, we use the *Vinciarelli* [33] features. On GW, we report the results of [25] on **DTW** as well as their results with **SC-HMM**. Although the results of [25] are not *exactly* comparable, we use partitions of the same size and very similar protocols. We also report results using the **character HMM** of [7], as well as the results reported on [9] using that method with a simpler subset. Although these lasts results are not directly comparable, they can give an accurate idea of the complexity of the dataset.

We observe how the FV baseline is already able to outperform some popular methods on both datasets. This is in line with the findings of [20], where the FV of a HMM outperforms the standard HMM on keyword classification tasks. The exemplar SVM has a limited influence. We believe it is more suited for segmentation-free approaches (as applied in [1]), where the random negatives mining is useful to discard windows that only contain partial words. The character HMM seems to perform well on the IAM dataset, precisely because it exploits the relations between characters of different words during training. Finally, we observe how the proposed method (either with Platts or with KCCA calibration) clearly outperforms all the other methods with a very large margin. These improvements are not only in terms of accuracy and memory use. Our optimized DTW implementation in C took more than 2 hours to compare the $5,000$ queries of IAM against the $16,000$ dataset words on a 12-core Intel Xeon E7540 at 2.00GHz with 128Gb of RAM, using one single core. By contrast, comparing the same queries using our attributes embedded with CCA took less than 3 seconds on the same machine.

For the QBS experiments, we compare ourselves with the recent method of Frinken *et al.* [9], which is the only one, to the best of our knowledge, that reports results on GW and IAM datasets for QBS. Note however that comparison should be exercised with caution: although we use similar set partitions for both GW and IAM datasets, [9] does not perform word spotting but *line* spotting, *i.e.*, finds the lines where the word appears. On IAM, Frinken [9] reports a 79% map *using as queries the most common non-stop words appearing in the training set*, while we obtain a 71% using all the non-stop words appearing in the test set, *whether they appear on the training set or not*. On GW they use the same protocol to select queries as we do, and they report a 84% of map, which compares to our 87%. Finally, [9] also shows results on GW when training their Neural Network exclusively on the IAM dataset, reporting a 43%, where our method obtains a 54%.

Finally, on Figure 4 we show some qualitative results on the IAM dataset, where we observe how the retrieved words can have very different styles from the query and still be retrieved successfully.

5. Conclusions

This paper proposes a method for multi-writer word spotting in handwritten documents. We show how an attributes-based approach based on a pyramidal histogram of characters can be used to learn how to embed the word images in a more discriminative space, where the similarity between words is independent of the writing style. This attributes representation leads to an unified representation of word images and strings, resulting in a method that allows one to perform either query-by-example or query-by-string searches. We show how to jointly calibrate all the attributes scores by means of CCA and KCCA, outperforming standard calibration methods. We compare our method in two public datasets, outperforming state-of-the-art approaches and showing that the proposed attribute-based representation is well-suited for word searches, whether they are images or strings, in handwritten documents.

Although in this paper we have focused on already segmented words, one of the main purposes of having compact features is to be able to perform retrieval without segmenting the image words. In the future we plan to explore the use of these approaches in a segmentation-free context, where the word images are not segmented.

References

[1] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Efficient exemplar word spotting. In *BMVC*, 2012. 1, 2, 3, 6, 7

[2] M. B. Blaschko and C. H. Lampert. Correlational spectral clustering. In *CVPR*, 2008. 5

[3] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011. 2

[4] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints.

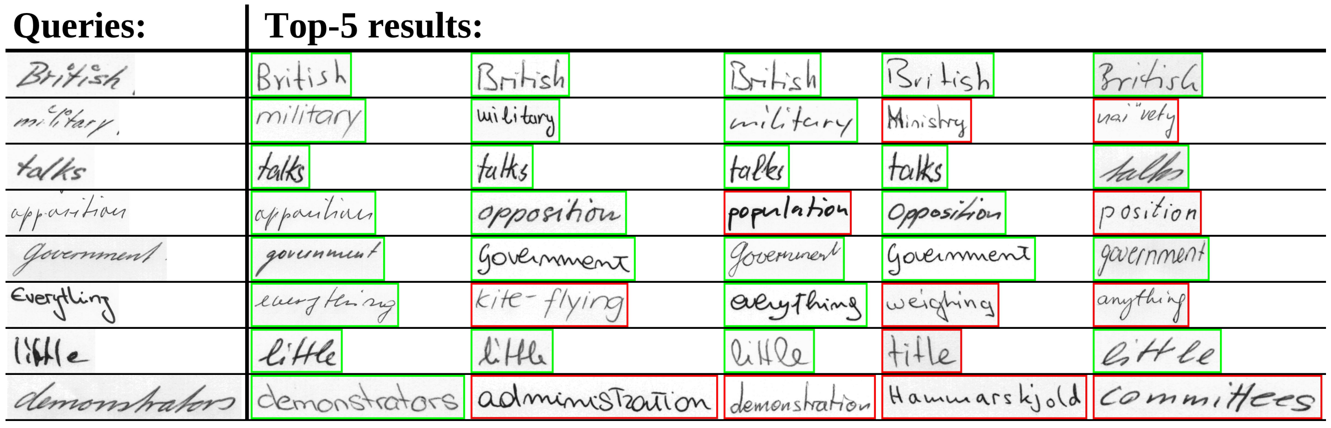

Figure 4. Sample queries from the IAM dataset and top-5 results using our attributes+KCCA approach. Relevant words to the query are outlined in green.

In *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004. 1

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1, 3

[6] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009. 3

[7] A. Fischer, A. Keller, V. Frinken, and H. Bunke. HMM-based word spotting in handwritten documents using sub-word models. In *ICPR*, 2010. 1, 4, 7

[8] A. Fischer, A. Keller, V. Frinken, and H. Bunke. Lexicon-free handwritten word spotting using character HMMs. *PRL*, 2012. 4

[9] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke. A novel word spotting method based on recurrent neural networks. *IEEE TPAMI*, 2012. 1, 2, 4, 7

[10] B. Gatos and I. Pratikakis. Segmentation-free word spotting in historical printed documents. In *ICDAR*, 2009. 3

[11] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE TPAMI*, 2012. 5

[12] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis; an overview with application to learning methods. Technical report, Department of Computer Science, Royal Holloway, University of London, 2003. 5

[13] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, 1999. 3

[14] P. Keaton, H. Greenspan, and R. Goodman. Keyword spotting for cursive document retrieval. In *DIA*, 1997. 3

[15] C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009. 3

[16] C. Leslie, E. Eskin, and W. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Pacific Symposium on Biocomputing*, 2002. 2

[17] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *JMLR*, 2002. 2

[18] R. Manmatha, C. Han, and E. M. Riseman. Word spotting: A new approach to indexing handwriting. In *CVPR*, 1996. 3

[19] U.-V. Marti and H. Bunke. The IAM-database: An english sentence database for off-line handwriting recognition. *IJDAR*, 2002. 5

[20] F. Perronnin and J. Rodríguez-Serrano. Fisher kernels for handwritten word-spotting. In *ICDAR*, 2009. 3, 7

[21] A. Rahimi and B. Recht. Randomm features for large-scale kernel machines. In *NIPS*, 2007. 5

[22] T. Rath and R. Manmatha. Word spotting for historical documents. *IJDAR*, 2007. 1, 5

[23] T. Rath, R. Manmatha, and V. Lavrenko. A search engine for historical manuscript images. In *SIGIR*, 2004. 2

[24] J. Rodríguez-Serrano and F. Perronnin. Local gradient histogram features for word spotting in unconstrained handwritten documents. In *ICFHR*, 2008. 1

[25] J. Rodríguez-Serrano and F. Perronnin. A model-based sequence similarity with application to handwritten word-spotting. *IEEE TPAMI*, 2012. 1, 3, 7

[26] J. Rodríguez-Serrano and F. Perronnin. Synthesizing queries for handwritten word image retrieval. *PR*, 2012. 2, 4

[27] J. Rodríguez-Serrano and F. Perronnin. Label embedding for text recognition. In *BMVC*, 2013. 2, 5

[28] M. Rusiñol, D. Aldavert, R. Toledo, and J. Lladós. Browsing heterogeneous document collections by a segmentation-free word spotting method. In *ICDAR*, 2011. 1, 2, 3

[29] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the Fisher vector: Theory and practice. *IJCV*, 2013. 2, 6

[30] W. J. Scheirer, N. Kumar, P. N. Belhumeur, and T. E. Boult. Multi-attribute spaces: Calibration for attribute fusion and similarity search. In *CVPR*, 2012. 4

[31] B. Siddiquie, R. S. Feris, and L. S. Davis. Image Ranking and Retrieval Based on Multi-Attribute Queries. In *CVPR*, 2011. 3

[32] L. Torresani, M. Szummer, and A. Fitzgibbon. Efficient object category recognition using classemes. In *ECCV*, 2010. 3

[33] A. Vinciarelli and S. Bengio. Offline cursive word recognition using continuous density hidden markov models trained with PCA or ICA features. In *ICPR*, 2002. 7