Separability versus Prototypicality in Handwritten Word Retrieval

Jean-Paul van Oosten and Lambert Schomaker
Dept. of Artificial Intelligence, University of Groningen, The Netherlands
{*J.P.van.Oosten,L.Schomaker*}*@ai.rug.nl*

Abstract

User appreciation of a word-image retrieval system is based on the quality of a hit list for a query. Using support vector machines for ranking in large scale, handwritten document collections, we observed that many hit lists suffered from bad instances in the top ranks. An analysis of this problem revealed that two functions needed to be optimised concerning both separability and prototypicality. By ranking images in two stages, the number of distracting images is reduced, making the method very convenient for massive scale, continuously trainable retrieval engines. Instead of cumbersome SVM training, we present a nearest-centroid method and show that precision improvements of up to 35 percentage points can be achieved, yielding up to 100% precision in data sets with a large amount of instances, while maintaining high recall performances.

1. Introduction

In handwriting recognition, classification is often performed using statistical methods (see for example the overview in [4]). The class indexed i with the highest posterior probability given the sample to be classified is chosen as the result of the classifier:

$$i_{\text{recog}} = \underset{i}{\text{argmax}}\, P(C_i|X) \quad \text{where } i \in \{1, N_c\} \qquad (1)$$

However, when the goal is word search, rather than automatic text transcription, the user is more interested in retrieval of word instances. Instead of a single classification, the result is a sorted hit list H. Each instance indexed j is ranked with respect to the prototype or class-model corresponding to the search term:

$$H = \underset{j}{\text{sort}}(P(X_j|C)) \quad \text{where } j \in \{1, N_x\} \qquad (2)$$

Retrieval is usually performed on a large collection of instances and only the top of the sorted list, representing the best ranking instances, is considered as interesting. Under such a condition, a large number of classes and a massive data collection can pose a problem, since for each query there is a large number of distractors, i.e., concerning instances from all classes, other than the target class.

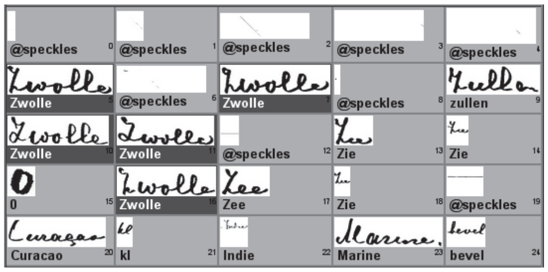

Figure 1. First 25 instances in a hit list of the word 'Zwolle'. Original test set performance: Accuracy: 99.2%, precision: 97.6% and recall: 97.6%. In a realistic test condition with 12k distractors, actual precision is as low as 2.8%.

This becomes apparent in retrieval engines for handwritten words in historical collections [15]. In the *Monk* system, twenty books of ≈1000 pages each contain millions of word zones or word candidates, and the lexicon is in the order of tens of thousand word class models. From the tradition of handwriting-recognition research, it seems reasonable to start with the classification problem (Eq. 1), using good shape features and a powerful classifier, such as, e.g., hidden-Markov models [9, 1] or the support-vector machine [17, 3]. For a word-mining task, such a classifier may be trained to discriminate a particular word class, and a ranked word list may be constructed, e.g., using the signed SVM discriminant value d_{SVM} for sorting. The basic assumption then is, that the distance from the margin, i.e., from the instances in the distractor classes, will be a good criterion for constructing a ranked hit list for a target class. However, upon applying this approach, we observed an interesting phenomenon in the resulting hit lists. As an example, Figure 1 shows the top-25 instances in a hit list for the word 'Zwolle'. The performance for the word classifier on the entire training set was 100% accuracy, with a 97% accuracy on an independent test set ($k = 7$ folds, $\sigma = \pm 1\%$). Following regular testing procedures for SVMs, the training and the test sets were of similar size, each containing a quarter of positive examples (typically 50) and three quarters of negative or distractor examples. However, the resulting hit list contains a number of counter-intuitive samples (e.g., speckle images) in the early ranks, followed by a strand of correct classifica-

978-0-7695-4774-9/12 $26.00 © 2012 IEEE
DOI 10.1109/ICFHR.2012.269

Table 1. Counter-intuitive, low precision results for good classifiers

		Accuracy		Recall		Precision	
Set	N_{examples}	Mean	σ	Mean	σ	Mean	σ
Test	120+	0.98	0.02	0.97	0.05	0.96	0.07
	60-120	0.97	0.03	0.95	0.10	0.91	0.13
	35-60	0.97	0.04	0.93	0.15	0.85	0.19
	7-35	0.96	0.04	0.68	0.42	0.57	0.40
+12K Distractors	120+	0.99	0.01	0.97	0.05	0.26	0.26
	60-120	0.98	0.02	0.95	0.10	0.06	0.12
	35-60	0.97	0.02	0.93	0.15	0.03	0.06
	7-35	0.97	0.04	0.68	0.42	**0.01**	0.05

tions which is followed by a transitional stage of occasional errors.

The impression that there exists a problem is confirmed by a larger-scale analysis of the results (Table 1), also using a realistic large set containing $\approx 12 \times 10^3$ distracting word instances. From the results it can be seen that although the results for *recall* and *accuracy* on the realistic data set confirm the hopeful expectancies which were raised by the regular training and test sets, the *precision* of the output drops abysmally, to about 1% in the worst cases, notably the classes with a limited number of training examples (Table 1, lower right), when the number of distractors is large and realistic.

It is clear that something is needed to improve on the performance. User appreciation of hit lists is of paramount importance in live and continuously trainable systems that rely on user annotation over the internet, such as *Monk*[15, 16]. Upon giving the first handful of (bootstrap) examples, a usable machine-learning system should be able to produce an acceptable ranking such that newly found instances of the same class can be easily labelled. The above, concrete observation thus gives rise to a more fundamental question: How is it possible that accuracy is not a good predictor of precision in a retrieval context?

In this study, we will 1) analyse the reason for unexpected, low precision in presumably well-performing classifiers; 2) explore a number of methods to counteract the precision drop and 3) present a convenient approach using nearest-centroid matching, with results in a similar ballpark as the abovementioned SVM approach, at the same time however, avoiding expensive training on the tens of thousands of word classes.

2. Separability versus Prototypicality

The SVM is a discriminative classifier, optimised for *classification* (Eq. 1). The class of an unknown sample X (Figure 2) is decided by determining on which side of the decision boundary β the sample falls. For *retrieval* purposes, it appears reasonable to use the distance to the boundary, $d(X, \beta)$, as a ranking measure: the farther the instance is located from the boundary, the more certain an SVM classifier is of the classification.

Unfortunately, this gives unexpected results, such as shown in Figure 1 for the query word 'Zwolle'. Instances that are ranked at the top ("@speckles") appear to be counter intuitive to a human user. It seems that there are

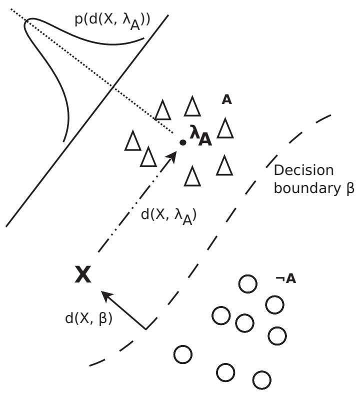

Figure 2. Separability vs. Prototypicality: For an unknown instance X, a large distance from a margin β does not imply a short distance from the prototype λ_A

two problems: 1) the distance to the boundary is not an intuitive measure, and 2) a fairly large number of distractors causes noise in a hit list, and consequently, a lower precision. The implication is that enlarging the dataset increases the probability that incorrect instances occur even before the first correct hit. This has a large impact on the user appreciation and is hard to explain. More informally: Many hits do not appear similar to the expected, canonical prototype for the query. In order to give a plausible explanation of this phenomenon, we present a schematic, two-dimensional overview. The position of an instance X in Figure 2 has a large distance $d(X, \beta)$ from the boundary β (which is desirable). However, the instance X is not very prototypical, being located far from the known instances of the target class A. In other words, the distance of the instance X to the prototype, or centroid of class A, $d(X, \lambda_A)$, is large.

The support-vector machine training mechanism has an emphasis on *separability*: the ability to categorise and separate class instances from non-class instances. This ability is usually achieved by evaluating the computed signed distance of an unknown sample to the decision boundary $d(X, \beta)$ which indicates on which side the instance X falls. However, by focusing on separation, an important aspect of pattern recognition is neglected: The phenomenon of *prototypicality* which concerns the similarity of an instance to the canonical class prototype, for instance, measured as the distance to the centroid or prototype of the class $d(X, \lambda_A)$. Quantitatively, prototypicality can be defined as $p(d(X, \lambda_A))$ and is the underlying rationale for Bayesian (as opposed to SVM) classifiers.

For a search and annotation tool of handwritten historical documents, separability and prototypicality need to be optimised simultaneously. However, most classifier methods optimise for one property, not both. The solution proposed in this study, is to combine classifiers in a two-stage process. The classifier that optimises separability is used in the first stage to divide the instances and produce the most likely class C for an unlabelled instance. The goal is to reduce the number of distractors for the second stage. More specifically, the set of distractors of an instance classified as C will be a considerable reduction of the set of all instances.

All instances labelled as C are then gathered for the second stage, where all instances are re-ranked or resorted with a secondary feature or method, one that optimises the ability to rank instances according to prototypicality. This ensures that if an instance is classified as class C in the first stage, but is an atypical result (such as the first few results in Figure 1, i.e., the speckles), the instance will end up at a later position in the hit list than other, more prototypical examples. Similar problems will occur if reject criteria need to be defined while using the SVM [10], or when there are very few negative examples to train from (for example, in a machine diagnostics problem)[14].

The results from the SVM experiment in the introduction suggest that a larger number of distractors has a negative effect on retrieval precision. It should be noted that the experiments in this study are conducted in a laboratory setting, using only human labelled instances. In a real-world setting, the problem of distractors will even be worse: the problem space is then heavily populated with non-word images and other noise. For example, in Monk, over all collections there are 22×10^3 classes, with over 124×10^6 word images, including rejectable candidates and noise.

3. Methods

Figure 3 shows the probability of finding the first correct hit in the ranks 0 to r of the hit lists generated in the preliminary study from the introduction. It is apparent that the probability of finding the first correct hit in the first five ranks is roughly 0.4, when using the SVM discriminant value for initial (tier 1) ranking. By reordering the images using a different feature ('img' or 'qp'), the performance can be improved, such that the first correct hit is found in the first five ranks 80% of the time. This is hopeful, but this is not enough and the hit list still contains counter intuitive results in the top ranks. Another possible method for improving the tier-1 performance, is using a multiclass SVM (for example, using decision trees [13]). However, this approach is fairly complex and

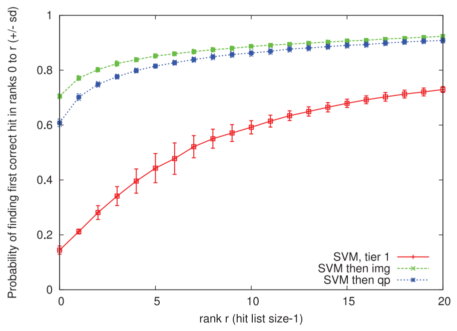

Figure 3. Probability of finding the first hit in ranks 0 to r for raw and reranked SVM output ($N_{\text{folds}} = 7$)

requires a large number of training instances for each of the more than 10^4 classes. The *Monk* system is a continuous, '24/7' training system: Labels are continuously added or changed, and it would be too time consuming and require human monitoring to train and retrain SVM classifiers when the system is updated. Nearest-centroid classifiers, on the contrary, can be easily updated with new knowledge by just adding a new feature vector to the set of training samples and averaging the samples to get the centroid. Rather than constituting a simplistic old-fashioned method, nearest-neighbour approaches are at the core of important advances in computational linguistics [5] and image retrieval [6, 7]. Therefore, in this study, we will use a nearest-centroid classifier for the classification stage, instead of SVMs.

The choice of word-based image retrieval instead of character-based approaches is based, firstly, on the observation that in some historical document collections contractions and loops are used to suggest characters in order to speed up writing (Figure 4). This makes creating a mapping between letter identity and character shape non-trivial. Secondly, due to the large variety of scripts and languages, most character-based approaches would need to be fine-tuned for each script and language, leading to long projects to process new collections ("each book its PhD project"). Our goal is to collect huge number of labelled word images first in order to develop character-based classifiers at a later stage, when necessary.

As discussed in the introduction, classification is performed by finding the class with the highest probability given the data. Since nearest neighbour classifiers are distance-based, the class with the highest probability is the class with the smallest distance to the instance:

$$\underset{i}{\operatorname{argmax}}\, P(C_i|X) = \underset{i}{\operatorname{argmin}}\, d(C_i|X) \qquad (3)$$

Similarly, retrieval is performed by ranking all instances based on their distance to a class-model.

Two features were experimentally chosen from a set of features to be used in the experiments. The first, dubbed "qp", is a feature based on the biologically inspired features introduced in [15], and a simple feature

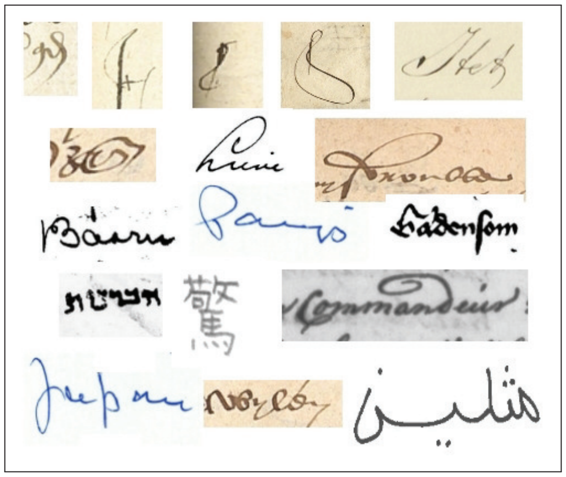

Figure 4. This variety of styles and shapes in a realistic collection illustrates that 'optical character recognition' of handwriting, by some form of sliding window over a word, is only applicable to a small subset. Many patterns are abbreviations, linguistic contractions or suffer from deformed, 'suggested' characters. In the absence of character models, the total-word image on the contrary provides a rich and redundant pattern in all cases, and can be labelled easily by volunteers.

consisting of the normalised and scaled image. The dimensionality of the former feature is 4358, whereas the scaled image has a size of 100×50, yielding a comparable dimensionality of 5000. In both feature types, the vector consists of probability values, adding up to one.

Two methods of retrieval will be compared: 1) direct retrieval: ranking, in a single step, all instances from the test set with the distance of the image to the centroid of the target class, and 2) the two stage re-ranking method as described in the previous section: do recognition on all instances first, then for each class C rank its candidates. The re-ranking method can be done in four ways using the two features: recognition with either feature and ranking with either feature. All four combinations are used to study the effect of using a different, secondary feature in the re-rank phase.

There are a number of measures to be used for comparing recognition and retrieval: (a) For recognition, we define top-1 recognition accuracy as: The probability that the nearest-centroid is of the correct class. For retrieval, the standard measures (b) precision and (c) recall will be considered as well as (d) the *average edit distance* in the top-7 of each hit list.

Accuracy (a) is defined as the percentage correctly classified instances:

$$\text{Accuracy } = \frac{N_{correct}}{N_{total}} \tag{4}$$

with $N_{correct}$ is the total number of correctly classified instances (in the top-1), and N_{total} is the total number of instances.

Precision (b) is defined as the proportion of correctly retrieved instances of class C in a fixed hit list H, with target size n, and can be computed with

$$\text{Precision in top-}n = \frac{N_{correct}}{\min(n, |H|)} \tag{5}$$

where $N_{correct}$ is the number of instances with the correct label in the top-n and $|H|$ is the number of items in the hit list[1]. The minimum of n and $|H|$ is used because the hit list can be smaller than the target size of n items.

The recall measure (c) is defined as the proportion of instances of class C that can be found in the hit list; formally, it can be defined as

$$\text{Recall for class } C = \frac{N_{obtained}}{N_{targets}} \tag{6}$$

where $N_{obtained}$ is the number of instances retrieved with class C, and $N_{targets}$ is the total number of instances with class C in the given test set. The reported precision and recall are accumulated over all classes as proportions.

The concept of prototypicality cannot be seen in isolation from the application context. More specifically, users of a retrieval engine for historical handwritten words will have an evaluation of the quality of a hit list. In other words, $P(X_j|C)$ must reflect an underlying measure of similarity. In information retrieval, relevance feedback is used to estimate user appreciation[11]. Relevance feedback is outside the scope of this study, but to estimate the user appreciation, we use *average edit distance* as the fourth performance measure. The assumption is that if the ASCII-distance between the query and the actual label of an instance is small, the hit list will be *intuitive*, meaning that it reflects the users measure of similarity well. The specific edit distance implemented in this study is the Levenshtein distance[8].

The data set is drawn from the historical document collection from the Dutch Queen's Office (see also [15]), or "Kabinet der Koningin" (KdK). The complete data set has over 13×10^3 classes. However, in order to do a 7-fold cross-validation experiment, only the 1404 classes with seven or more human labelled word instances will be considered. These classes will be divided into four categories, based on the number of instances: 7 up to 35 instances, 35 up to 60 instances, 60 up to 120 instances and 120 or more instances, similar to what has been done in [15]. In total, there are more than 84×10^3 instances used. The experiments are performed on a cluster of eight Linux machines with 54 cores in total, connected to a 1.6 petabyte storage, of which the *Monk* system currently uses roughly 0.5 petabyte.

For each line strip, a number of word candidates are selected, based on the number and size of connected

[1]According to the Wikipedia article on precision and recall (`http://en.wikipedia.org/wiki/Precision_and_recall`, last accessed 30 January 2012), this is also called "precision at n" or "P@n"

Table 2. Top-1 accuracy ($N_{\text{folds}} = 7$)

Feature	N_{examples}			
	7-35	*35-60*	*60-120*	*120+*
qp	0.62	0.93	0.92	0.94
img	0.62	0.86	0.87	0.93

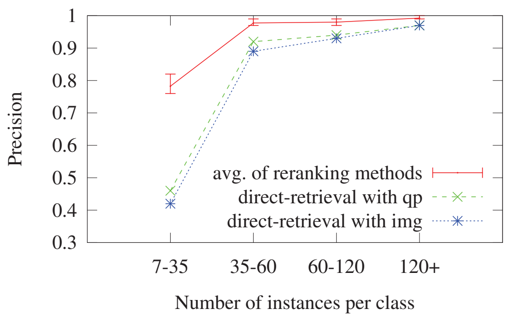

Figure 5. Precision performance in top-1, re-rank vs direct ($N_{\text{folds}} = 7$)

components. This means that the line is usually over-segmented, which leads to overlap between images. To avoid that multiple images that actually belong to the same word instance end up in both the training and test set, the assignment of an image to a fold is based on the page number from which the image originated from: fold $\equiv$ page number $\pmod 7$, where 7 is the number of folds. This has the additional benefit that words that are written consistently on one page, but inconsistently over the entire collection are also assigned to the same fold.

4. Results

We look at two types of comparisons: between re-rank methods (choice of features) and between average re-rank performance and direct retrieval (i.e., without re-ranking). Table 2 shows the top-1 recognition accuracy, averaged over all seven folds for both features. The 'qp' feature outperforms the scaled image feature, especially in the 35-60 and 60-120 categories. Furthermore, the table shows that to accurately classify an instance, the nearest-centroid classifier needs around 35 training instances.

Figures 5, 6 and 7 compare the average of the re-rank methods to the direct retrieval methods. The bars on the averages show the minimum and maximum value of the re-rank methods. These results show the gain in performance when using the re-ranking methods instead of direct retrieval. As was expected, reducing the number of distractors has a positive impact on performance.

Table 3 and Table 4 show the precision (in the top-1) and recall figures. In general, these results show that re-ranking with a different feature can boost performance. Using the qp feature as a classification feature and the image feature for ranking works best for this data collection, even getting a top-1 precision of 1.0 (i.e., 100%) with a standard deviation of 0 in the 120+ category.

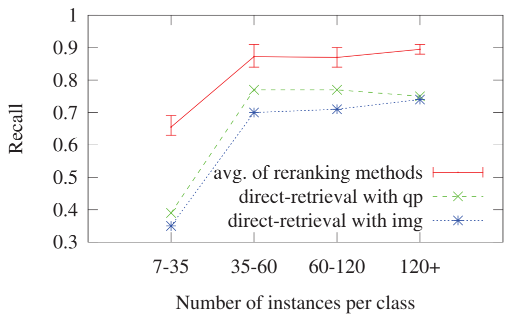

Figure 6. Recall performance, re-rank vs direct ($N_{\text{folds}} = 7$)

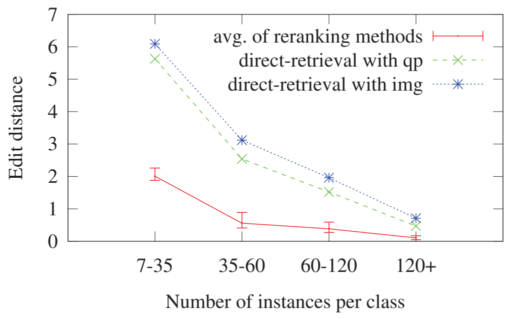

Figure 7. Average edit distance in top-7, re-rank vs direct ($N_{\text{folds}} = 7$)

Overall, the results show that all methods perform roughly the same when there are enough labelled samples (i.e., in the 120+ category).

5. Conclusions

In the design of a large scale retrieval engine for historical handwritten manuscripts, it was observed that classifier accuracy is not a good predictor of retrieval precision. Very low precision performances occurred on good classifiers when using a realistic number of distractors. In retrospect, the choice of using the signed distance d_{SVM} from the margin for ranking was evidently suboptimal, but it elucidated two separate functions to be performed: 1) data reduction by optimal *separation* and 2) ranking instances in terms of their *prototypicality* with respect to their class.

The re-ranking method has two main advantages: the focus on both separability and prototypicality increases the probability that the top of a hit list is more similar to the user's expectation than otherwise. Secondly, the reduction of distractors lowers the number of noisy instances in a hit list and is advantageous in terms of processing demands. As the results presented in the previous section show, reducing the number of distractors in a retrieval experiment improves precision and average edit

Table 3. Precision results ($N_{\text{folds}} = 7$, $\sigma \leq 0.03$)

Method	N_{examples}			
	7-35	*35-60*	*60-120*	*120+*
Direct, feat=img	0.42	0.89	0.93	0.97
Direct, feat=qp	0.46	0.92	0.94	0.97
Re-rank, img, img	0.76	0.97	0.98	0.99
Re-rank, img, qp	0.76	0.97	0.98	0.99
Re-rank, qp, qp	0.79	0.98	0.97	0.99
Re-rank, qp, img	**0.82**	**0.99**	**0.99**	**1.00**

Table 4. Recall results ($N_{\text{folds}} = 7$, $\sigma \leq 0.03$)

Method	N_{examples}			
	7-35	*35-60*	*60-120*	*120+*
Direct, feat=img	0.35	0.70	0.71	0.74
Direct, feat=qp	0.39	0.77	0.77	0.75
Re-rank, img, img	0.63	0.84	0.84	0.88
Re-rank, img, qp	0.63	0.84	0.85	0.89
Re-rank, qp, qp	0.67	0.90	0.89	0.90
Re-rank, qp, img	**0.69**	**0.91**	**0.90**	**0.91**

distance in the hit list, which we assume will increase the user appreciation of hit lists.

It appeared to be beneficial for retrieval performance to use different features in the stages. The optimal features and processing order will depend on the material. In the KdK data set, precision benefited the most by using a strong, robust feature for recognition first, and a secondary feature with a strong image-based component that works well on collections where words are written fairly consistently. On data sets where the writing varies a lot within a class, other features or classifier methods may prove to be more advantageous, including (k-means) clustering to capture the different writing styles.

When a class has enough instances (i.e., the 120+ category), choice of feature does not seem to have much effect on retrieval performance. On the other hand, reducing the number of distractors by a two-step approach is still beneficial. In the bootstrapping phase of a retrieval system (i.e., the 7-35 category), the choice of feature does have a big impact. Even small performance increases have large consequences in this stage, helping the user to label new instances with little effort (*Monk* presents hit lists in its web-based labelling interface).

The methods presented in this paper can use all kinds of classifiers. Currently, nearest-centroid classifiers are used due to the nature of '24/7' learning, where new labels are being added frequently. It would be cumbersome to retrain classifiers such as SVMs every time a new label was added. The SVM has one benefit in the bootstrap phase: its recognition accuracy is better than the performance of a nearest neighbour classifier. However, the 7-35 category in this experiment has the most classes by far, which would be very inconvenient for the training of tens of thousands multi-class SVMs.

The *Monk* project has a large number of collections with different script types: 15th and late 19th century texts (cursive with a lot of abbreviations and variation), Qumran scrolls (isolated characters), captain's logs (cursive) and even Thai[12] and Bangla[2] texts. The different shapes and writing styles have different requirements of the features; For each script, features will be selected to optimise both separability and prototypicality.

References

[1] T. Artieres, S. Marukatat, and P. Gallinari. Online handwritten shape recognition using segmental hidden Markov models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):205–217, 2007.

[2] T. Bhowmik, J. van Oosten, and L. Schomaker. Segmental K-means learning with mixture distribution for HMM based handwriting recognition. *Pattern Recognition and Machine Intelligence*, pages 432–439, 2011.

[3] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[4] Bunke, H. Recognition of cursive Roman handwriting: past, present and future. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 448–459. IEEE, 2003.

[5] W. Daelemans and A. van den Bosch. *Memory-based language processing*. Cambridge Univ Pr, 2005.

[6] G. Giacinto. A nearest-neighbor approach to relevance feedback in content based image retrieval. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 456–463. ACM, 2007.

[7] H. Jégou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, 87(3):316–336, 2010.

[8] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[9] U. Marti and H. Bunke. Handwritten sentence recognition. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 3, pages 463–466. IEEE, 2000.

[10] H. Mouchère. *Étude des mécanismes d'adaptation et de rejet pour l'optimisation de classifieurs: Application à la reconnaissance de l'écriture manuscrite en-ligne*. PhD thesis, l'Institut National des Sciences Appliquées de Rennes, 2007.

[11] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Readings in information retrieval*, 24:5, 1997.

[12] O. Surinta, L. Schomaker, and M. Wiering. Handwritten character classification using the hotspot feature extraction technique. In *Proceedings of the First International Conference on Pattern Recognition Applications and Methods, 2012*, pages 261–264, 2012.

[13] F. Takahashi and S. Abe. Decision-tree-based multiclass support vector machines. In *Proceedings of the 9th International Conference on Neural Information Processing*, volume 3, pages 1418–1422. IEEE, 2002.

[14] D. Tax. *One-class classification*. PhD thesis, Technische Universiteit Delft, 2001.

[15] T. van der Zant, L. Schomaker, and K. Haak. Handwritten-word spotting using biologically inspired features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(11):1945–1957, 2008.

[16] T. van der Zant, L. Schomaker, S. Zinger, and H. van Schie. Where are the search engines for handwritten documents? *Interdisciplinary Science Reviews, 34*, 2(3):224–235, 2009.

[17] V. Vapnik. *Estimation of Dependencies Based on Empirical Data*. Springer-Verlag, New York, 1982.