

Identifying Script on Word-Level with Informational Confidence

Stefan Jaeger, Huanfeng Ma, and David Doermann
Institute of Advanced Computer Studies
University of Maryland, College Park, MD 20742, USA
{jaeger, hfma, doermann}@umiacs.umd.edu

Abstract

In this paper, we present a multiple classifier system for script identification. Applying a Gabor filter analysis of textures on word-level, our system identifies Latin and non-Latin words in bilingual printed documents. The classifier system comprises four different architectures based on nearest neighbors, weighted Euclidean distances, Gaussian mixture models, and support vector machines. We report results for Arabic, Chinese, Hindi, and Korean script. Moreover, we show that combining informational confidence values using sum-rule can consistently outperform the best single recognition rate.

1. Introduction

The purpose of this paper is twofold: First, our intention is to improve word-level script identification. In particular, we want to improve on our previous experiments described in [9]. Second, we use script identification to show the benefit of informational confidence values, a technique recently introduced in [5, 6]. Classifier combination is a particularly promising application for informational confidence values. We are thus going to apply a multiple classifier system to the problem of script identification in this paper, which is structured as follows: Section 2 presents some existing approaches for script identification and outlines our approach in detail. Section 3 describes the classifiers used in our experiments, and the way we compute confidence values for each of them. Section 4 elaborates on the main focus of this paper: informational confidence. Finally, Section 5 presents our classifier combination results for word-level script identification with informational confidence, and a short summary then concludes the paper.

2. Script Identification

Among the significant number of documents that can only be accessed in printed form, a large portion are mul-

tilingual documents, such as patents or bilingual dictionaries. In general, automatic processing of multilingual documents requires that the scripts must be identified before they can be fed into an appropriate OCR system.

Earlier work on script identification includes template-based approaches ([4]), approaches exploiting character-specific characteristics ([12, 13, 14]), as well as text line projections ([15]) and font recognition based on global texture analysis ([16]).

Our approach is different in that it operates on the word-level [9]. We use a modified Docstrum algorithm ([10]) to first segment the document into words and then apply a script classification on word-level. Our focus is on bilingual documents with one script being either English or a script based on the Latin alphabet. The other scripts we have experimented with are Arabic, Chinese, Hindi, and Korean. Script identification then becomes a 2-class classification problem for each word. Figure 1 shows a page from a bilingual Arabic-English word dictionary, with the Arabic words being successfully identified. We use Gabor filters to compute features for each script class. More information on these features and other pre-processing steps is given in [9].

3. Classifiers

In the following description, we use \mathbf{x} to represent the feature vector, which is a 32-dimensional vector value, and \mathbf{S} to represent the training set. Script identification is a 2-class classification problem in our case, so all the equations focus on the 2-class problem.

3.1. Nearest Neighbor (KNN)

First introduced by Cover and Hart [2] in 1967, the Nearest Neighbor (NN) classifier was proven to be a very effective and simple classifier. It found applications in many research fields. Using the NN classifier, a test sample \mathbf{x} is classified by assigning it the label that its nearest sample represents in the training samples. The distance between two

2٤, (pl. 7, 23), Needle.—**20**,
20٤, (pl. 44), Needle-case.—
 (b), Calumny.
أَبْرَشِيَّة G., Diocese.
إَبْرِيَز P., Pure gold.
إَبْرِيْسِم, **إَبْرِيْسِيْم** P., Silk.
أَبْرِيْق P., (pl. 69), Jug.
إَبْرِيْم P., (pl. 69), Kind of clasp.
أَبْص I, U (n. ac. 1, 27), Tied the
 (camel's) fore leg.—(b), Loosed.
3, Time.—(b), Eternity.—**18**,
 Inner side of the knee.

Figure 1. Arabic word segmentation.

feature vectors is usually measured by computing the Euclidean distance. In our experiments, we compute the classifier's confidence in a class label λ_i assigned to a testing sample \mathbf{x} as follows:

$$\text{conf}(\mathbf{x}|\lambda_i) = \min_{\mathbf{x}_t \in S_{\lambda_i}} (\text{dis}(\mathbf{x}, \mathbf{x}_t)),$$

where $\text{dis}(\mathbf{x}, \mathbf{x}_t)$ computes the Euclidean distance between two vectors \mathbf{x} and \mathbf{x}_t , and S_{λ_i} is the set containing all training samples with label λ_i .

3.2. Weighted Euclidean Distance (WED)

From the training samples, we compute the mean $\mu^{(i)}$ and standard deviation $\alpha^{(i)}$ of the training samples for each class λ_i . Then for each test sample \mathbf{x} , we compute the distance between \mathbf{x} and λ_i using the following formula:

$$\text{dis}(\mathbf{x}, \lambda_i) = \sum_{k=1}^d \left| \frac{x_k - \mu_k^{(i)}}{\alpha_k^{(i)}} \right| \quad i = 1 \dots M$$

where d is the feature dimension and M is the number of classes. Testing samples are assigned the class label with the minimum distance. We compute the classifier's confidence in a class label λ_i as follows:

$$\text{conf}(\mathbf{x}|\lambda_i) = \frac{\sum_{j=1, j \neq i}^M \text{dis}(\mathbf{x}, \lambda_j)}{\sum_{j=1}^M \text{dis}(\mathbf{x}, \lambda_j)}$$

3.3. Support Vector Machine (SVM)

SVMs were first introduced in the late seventies, but are now receiving increased attention. The SVM classifier con-

structs a 'best' separating hyperplane (the maximal margin plane) in a high-dimensional feature space which is defined by nonlinear transformations from the original feature variables. Burges [1] gave a detailed description on how to find the separating hyperplanes. We chose the SVM implementation *SVM-light* [7] and the polynomial kernel function in our work. In a SVM classifier, the distance between the testing sample and the hyperplane reflects the confidence in the final classification. The larger the distance, the more confidence the classifier has in its classification result. Hence, we can directly use this distance as confidence in our experiments.

3.4. Gaussian Mixture Model (GMM)

The Gaussian Mixture Model (GMM) classifier is used to model the probability density function of a feature vector, \mathbf{x} , by the weighted combination of M multi-variate Gaussian densities (Λ):

$$p(\mathbf{x}|\Lambda) = \sum_{i=1}^M p_i g_i(\mathbf{x}),$$

where the weight (mixing parameter) p_i corresponds to the prior probability that feature \mathbf{x} was generated by component i , and satisfies $\sum_{i=1}^M p_i = 1$. Each component λ_i is represented by a Gaussian mixture model $\lambda_i = N(p_i, \mu_i, \Sigma_i)$ whose probability density can be described as:

$$g_i(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i)\right),$$

where μ_i and Σ_i are the mean vector and covariance matrix of Gaussian mixture component i respectively. Details about how to estimate the classifier can be found in [9]. We take the likelihood value $\ln(p(\lambda_i|\mathbf{x}))$ as the classifier's confidence in labeling testing pattern \mathbf{x} as λ_i .

4. Informational Confidence

Most researchers would agree that the ability of a classifier to provide soft decisions is an integral part of classification. Classifiers providing merely hard decisions with no grading whatsoever give away much of the information inherent in their trained parameters, such as distances to templates or hyperplanes. This information is extremely useful for improving recognition rates with post-processing techniques like language models or other statistical methods, e.g. hidden Markov Models. Fortunately, most state-of-the-art classifiers now do return soft decisions. In fact, the importance of soft decisions has given rise to new research in fuzzy logic and soft-computing, motivated by the human mind's remarkable ability to process pervasively imprecise and uncertain information.

A soft-decision provided by a classifier for a pattern with unknown class is basically a list of candidates classes, each being assigned a confidence by the classifier. The confidence, as the name already suggests, indicates how confident the classifier is that a particular class label is indeed the correct class label for the input pattern. In the literature, there exist different names for this confidence provided by the classifier, e.g. measurement or likelihood. Throughout this paper, we will use the term confidence. For implementational reasons, we assume that confidence takes on discrete values.

One major drawback of confidence in practice is that confidence values from different information sources, e.g. multiple classifiers, are almost never compatible with each other, featuring different means, standard deviations etc. For instance, an information source providing a confidence of 0.8 is not necessarily twice as confident as another information source with confidence 0.4. Our solution to this problem is a standard representation for confidence values in which each confidence value matches its actual informational content. This approach was first presented in [5, 6] with good results for character recognition.

4.1. Information-Theoretical Approach

We take the notion of information literally and require confidence to be equal to information as defined by Shannon, i.e. equal to the negative logarithm [11]. Let K be a set of confidence values for a classifier C :

$$K = \{K_0, \dots, K_i, \dots, K_N\} \quad (1)$$

Furthermore, let K_0 denote the lowest possible confidence, and K_N be the confidence indicating that the classifier is absolutely certain about its output. Then, we compute new, informational confidence values K_i^{new} that satisfy the following linear equation:

$$\begin{aligned} K_i &= E * I(1 - p(K_i)) \\ &= E * -\ln(1 - p(K_i)) \end{aligned} \quad (2)$$

In (2), functional term I is just the negative logarithm, while E is a multiplying scalar and $p(K_i)$ stands for the performance of K_i . In other words, the new confidence values depend linearly on the information contained in their performance. The better the performance $p(K_i)$ of a confidence value, the more information it provides. A perfect performance of 1 provides infinite information, while a performance equal to 0 provides no information at all. Note that (2) says nothing about what $p(K_i)$ or E actually are. Nevertheless, we can find out the meaning of $p(K_i)$ and E by simply resolving (2) for $p(K_i)$:

$$\begin{aligned} K_i &= E * -\ln(1 - p(K_i)) \\ \Leftrightarrow \frac{K_i}{E} &= -\ln(1 - p(K_i)) \end{aligned}$$

$$\begin{aligned} \Leftrightarrow e^{-\frac{K_i}{E}} &= 1 - p(K_i) \\ \Leftrightarrow p(K_i) &= 1 - e^{-\frac{K_i}{E}} \end{aligned} \quad (3)$$

This straightforward derivation shows that $p(K_i)$ is actually the exponential distribution of a random variable with expectation value E : In statistics, the exponential density function $e_\lambda(x)$ with parameter λ is defined as follows:

$$e_\lambda(x) = \begin{cases} \lambda * e^{-\lambda x} & : x \geq 0 \\ 0 & : x < 0 \end{cases} \quad \lambda > 0 \quad (4)$$

The corresponding distribution $E_\lambda(k)$ describes the probability that the exponentially distributed random variable assumes values lower than or equal to k . This probability equals the area under the density curve delimited by k . We can compute $E_\lambda(k)$ by solving the following integral:

$$\begin{aligned} E_\lambda(k) &= \int_{-\infty}^k e_\lambda(x) dx \\ &= \int_0^k \lambda * e^{-\lambda x} dx \\ &= [-e^{-\lambda x}]_0^k \\ &= 1 - e^{-\lambda k} \end{aligned} \quad (5)$$

Note that (5) becomes identical to (3) once we set $\lambda = \frac{1}{E}$. This confirms the aforementioned result stating that the performance in (2) is actually an exponential distribution. Moreover, we now also know that parameter E is an expectation value in the statistical sense. This follows from the fact that an exponentially distributed random variable with distribution $E_\lambda(k)$ has an expectation value of $\frac{1}{\lambda}$.

The next subsection proposes a technique for computing informational confidence values based on this relationship between performance and distribution.

4.2. Learning Informational Confidence

Once we know the performance of a confidence value, we can compute the corresponding new informational confidence value by simply applying the definition in (2). Hence, the idea is to first estimate the performance of a confidence value K_i on an evaluation set, and then compute its new value according to (2). Motivated by the fact that the distribution provides the percentage of area covered under the density function for a particular confidence value K , we use the following estimate $\hat{p}(K_i)$ of $p(K_i)$:

$$\hat{p}(K_i) = R_i \quad (6)$$

R_i is the percentage of patterns in the evaluation set that classifier C correctly classifies with confidence equal to or less than K_i . For meaningful confidence values, R_i will in general increase monotonously over the set of confidence

values K , with $R_i \leq R$, and R being the overall recognition of C .

Analogous to the distribution $E_\lambda(k)$, we can consider $\hat{p}(K_i)$ as an estimate of the percentage of total information conveyed when classifier C has confidence K_i . The maximum information is received when C returns its highest possible confidence. Inserting the performance estimate $\hat{p}(K_i)$ directly into (2) then provides the new informational confidence values \hat{K}_i , which replace the old values:

$$K_i^{new} = E * -\ln(1 - \hat{p}(K_i)) \quad (7)$$

Note that we use an estimate of $\hat{p}(K_i)$ that is slightly different from the previous work in [5]. Also, in the experiments presented in the next section, we will use a slight refinement of (7) as introduced in [6], namely:

$$K_i^{new} = {}^{I(C)}\sqrt{R} * -\ln\left(1 - {}^{I(C)}\sqrt{\hat{p}(K_i)}\right) \quad (8)$$

In this equation, E is set to the overall recognition rate R of C , and all values are normalized according to the overall information $I(C)$ conveyed by C , with $I(C) = -\ln(1 - R)$. A more detailed discussion of this formula is not in the scope of this paper. Readers are referred to [6] for more details.

In summary, learning informational confidence values is a 3-step process: In the first step classifier C is trained with its specific training method and training set. In the second step, the performance for each confidence value and the overall recognition rate of C are estimated on a second training (evaluation) set according to (6). Finally, new informational confidence values are computed and stored in a look-up table according to (8). In all future classifications, confidence values provided by classifier C will then always be replaced with their informational counterparts stored in the look-up table.

5. Classifier Combination

We will concentrate on four elementary combination schemes in this paper: sum-rule, max-rule, product-rule and majority vote. As their names already suggest, sum-rule adds all confidence values for a class, product-rule multiplies confidence values, and max-rule simply takes the maximum value without any further operations. Majority vote is a simple voting, with a random break of ties in our case. Though there exist more complex schemes, it is by no means certain that those schemes are superior to more simpler ones. Also, the sum-rule is the natural way of integrating information from different sources [11]. It is very robust against noise as well, at least theoretically [8].

In the following experiments we compute the informational confidence values for each classifier, as described

Classifier	Arabic	Chinese	Korean	Hindi
KNN	90.90	92.19	94.04	97.51
WED	80.07	84.89	84.68	91.97
GMM	88.14	90.59	90.72	93.11
SVM	90.93	93.43	92.54	97.27

Table 1. Recognition rates for each script.

Combination	Arabic	Chinese	Korean	Hindi
Sum-rule	90.27	92.92	93.06	97.05
Inf. sum-rule	92.66	94.43	94.31	98.08
Max-rule	88.14	90.83	90.97	93.13
Product-rule	90.97	93.34	92.50	97.09
Majority vote	92.46	92.81	92.93	96.38

Table 2. Performance of combination schemes for each script.

above, and store them in a look-up table. For each test pattern, these new values replace the confidence values provided by the respective classifier, and then serve as input to the combination process. Note that we assume that high values indicate high confidence. This requires a mirroring of confidence values for the KNN and SVM before normalization.

5.1. Combination Experiments

Table 1 lists the individual recognition rates of each classifier for each writing system. The KNN and SVM classifiers provide the best overall performance. Classification rates are lowest for Arabic script since the texture formed by Arabic words is more similar to textures of Latin words than textures of other scripts.

Table 2 shows the combined recognition rates for each combination scheme and each script. The elementary combination schemes provide almost always lower recognition rates than the single best classifiers. Only the sum-rule in connection with informational confidence values provides better rates (see second row in Table 2). In fact, informational confidence values always outperform the single best recognition rate. The biggest improvement of more than 1.7% is on the most difficult script: Arabic.

5.2. AdaBoost

In order to have a better comparison with other state-of-the-art combination schemes, we also experimented with Boosting, in particular AdaBoost. AdaBoost (Adap-

Classifier	Arabic	Chinese	Korean	Hindi
KNN	90.95	92.22	94.12	97.39
WED	82.24	85.17	85.18	92.13
GMM	88.03	91.02	91.14	92.85
SVM	90.48	93.49	93.03	97.38

Table 3. Recognition rates with Boosting.

tive Boosting) was introduced by Freund and Schapire in 1995 to expand the boosting approach introduced by Schapire. In our work, we concentrate on the AdaBoost approach called AdaBoost.M1 (Freund and Schapire 1996). The AdaBoost algorithm generates a set of classifiers and votes them. It changes the weights of the training samples based on classifiers previously built (trials). The goal is to force the final classifiers to minimize expected error over different input distributions. The final classifier is formed using a weighted voting scheme. Details of AdaBoost.M1 can be found in [3]. In our experiments, we set the number of trials to 20.

Table 3 is a representative example of our Boosting experiments with different training sets. Except perhaps for WED, the improvements are only small when compared to the recognition rates in Table 1, and very often even worse. Also, the nearest neighbor classifier does not lend itself to boosting since it never misclassifies on the training set.

6. Summary

We have presented a classifier system of four classifiers for script identification. The performance of all four classifiers varies among different scripts. While the performance for Hindi script is pretty good, Arabic script identification has the lowest performance. When we combine all four classifiers, we achieve the greatest improvement for Arabic, and get also small improvements for the other scripts. The elementary combination schemes taken alone almost never resulted in an improved recognition rate. However, informational confidence values in combination with sum-rule consistently outperform the best single recognition rate.

While sum-rule is the most natural combination scheme for informational confidence values, informational confidence is by no means a contradiction to other combination schemes. On the contrary, informational confidence can be considered a general standard representation for confidence on which other combination schemes can rest.

Acknowledgment:

The partial support of this research under DOD contract MDA90402C0406 and NSF grant EIA0130422 is gratefully acknowledged.

References

- [1] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [2] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Trans. Information Theory*, IT-13(1):21–27, 1967.
- [3] Y. Freund and R. Schapire. Experiments with a New Boosting Algorithm. In *Proc. of 13th Int. Conf. on Machine Learning*, pages 148–156, Bari, Italy, 1996.
- [4] J. Hochberg, P. Kelly, T. Thomas, and L. Kerns. Automatic script identification from document images using cluster-based templates. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(2):176–181, 1997.
- [5] S. Jaeger. Informational Classifier Fusion. In *Proc. of the 17th Int. Conf. on Pattern Recognition*, pages 216–219, Cambridge, UK, 2004.
- [6] S. Jaeger. Using Informational Confidence Values for Classifier Combination: An Experiment with Combined On-Line/Off-Line Japanese Character Recognition. In *Proc. of the 9th Int. Workshop on Frontiers in Handwriting Recognition*, pages 87–92, Tokyo, Japan, 2004.
- [7] T. Joachims. *Advances in Kernel Methods-Support Vector Learning*, chapter Making Large-Scale SVM Learning Practical, pages 41–56. B. Schölkopf, C. Burges, and A. Smola, MIT-Press, 1999.
- [8] J. Kittler, M. Hatef, R. Duin, and J. Matas. On Combining Classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.
- [9] H. Ma and D. Doermann. Word level script identification for scanned document images. *Proc. of Int. Conf. on Document Recognition and Retrieval (SPIE)*, pages 178–191, 2004.
- [10] L. O’Gorman. The document spectrum for page layout analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(11):1162–1173, 1993.
- [11] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Tech. J.*, 27(623–656):379–423, 1948.
- [12] P. Sibun and A. L. Spitz. Language determination: Natural language processing from scanned document images. In *Proc. 4th Conference on Applied Natural Language Processing*, pages 115–121, Stuttgart, 1994.
- [13] A. L. Spitz. Determination of the script and language content of document images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(3):235–245, 1997.
- [14] C. Tan, T. Leong, and S. He. Language Identification in Multilingual Documents. In *Int. Symposium on Intelligent Multimedia and Distance Education (ISIMADE’99)*, pages 59–64, Baden-Baden, Germany, 1999.
- [15] B. Waked, S. Bergler, and C. Y. Suen. Skew detection, page segmentation, and script classification of printed document images. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC’98)*, pages 4470–4475, San Diego, CA, 1998.
- [16] Y. Zhu, T. Tan, and Y. Wang. Font recognition based on global texture analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(10):1192–1200, 2001.