

# Language Identification of Character Images Using Machine Learning Techniques

Ying-Ho Liu<sup>+</sup>, Chin-Chin Lin<sup>++</sup>, and Fu Chang<sup>+</sup>

<sup>+</sup>Institute of Information Science, Academia Sinica, Taipei, Taiwan

<sup>++</sup>Department of Electrical Engineering, National Taipei University of Technology, Taipei, Taiwan  
e-mail: {daxliu, erikson, fchang}@iis.sinica.edu.tw

## Abstract

*In this paper, we propose a new approach for identifying the language type of character images. We do this by classifying individual character images to determine the language boundaries in multilingual documents. Two effective methods are considered for this purpose: the prototype classification method and support vector machines (SVM). Due to the large size of our training dataset, we further propose a technique to speed up the training process for both methods. Applying the two methods to classifying characters into Chinese, English, and Japanese (including Hiragana and Katakana) has produced very accurate and comparable test results.*

## 1. Introduction

Identifying the language type of textual entities is an important research problem for document image analysis, because of the need to pre-classify documents or individual characters before nominating a proper subsequent module (layout analyzer, character recognizer, etc.) to work on them. The method we propose in this paper classifies a character image as Chinese, English, or Japanese. Documents comprised of these three character types are easy to find in real life (see Figure 1). Our method, however, is not limited to this application and is potentially useful for classifying characters in different combinations of languages. Our method is in fact almost identical to

that used for character recognition. The only difference lies in the class types involved in the applications. At first, it may appear that employing a character-recognition method to identify language types runs the risk of generating too large an overhead. However, by using an effective machine-learning method, we are able to train a classifier to draw the demarcation along boundaries between language types, rather than character types.

## 2. Background

A variety of methods have been proposed for this purpose. They can be classified into the following three categories.

- (1) The first category identifies coarse textual entities, such as textlines, text blocks, or complete documents, into different languages, assuming that the characters in the coarse entities belong to the same language (in other words, the textual entities are homogeneous) and have been recognized by an OCR technique.
- (2) The second category identifies the language type of homogeneous coarse entities, without first recognizing the characters in them.
- (3) The third category identifies the language type of individual characters, without analyzing the coarse entities to which they belong.

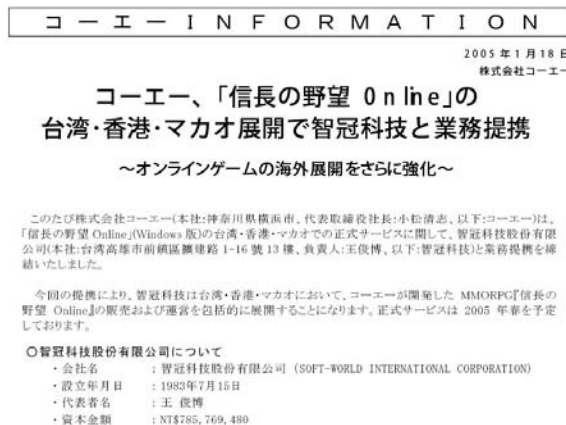


Figure 1. A document page comprised of English, Kanji (Chinese), and Japanese characters.

Methods in the first category are particularly appropriate for European languages, which have many characters in common. Nobile et al. [1] decompose European characters into 10 classes (A, x, I, g, j, U, D, Q, E, Z) and denote them as character shapes. Word shapes are then formed as sequences of these character shapes. For example, the string “Can’t get there from here” is coded as “AxxQA gxA AAxxx Axxx Axxx”. For each of the candidate European languages, one picks the top 50 most frequent word shapes, the top 5 most frequent word shapes that are unique to the language, the top 100 most frequent pairs of word shapes, and the top 100 most frequent triples of word shapes. The word shapes of a test textual entity are then compared with these four properties of each language to determine which language they belong to. Pham et al. [2] take a vector-quantization approach to language identification. Words of each candidate language are transferred to a vector of fixed size on the basis of the characters’ ASCII value. The codebook of each language, consisting of a fixed number of vectors, is then created with the help of a clustering algorithm. The same technique is applied to each test textual entity to create its own codebook. The language of a textual entity is then identified as the one whose codebook is most similar to the textual entity’s codebook. Nakayama and Spitz [3] define 12 shape classes of European characters. They then use the shapes of English, French and German as training data and apply LDA technique to perform classification.

Methods in the second category perform classification without first recognizing character images. Wood et al. [4] propose an approach for classifying coarse textual entities into European languages, Russian, Arabic, Chinese, and Korean text. They first use horizontal and vertical filters to eliminate strokes shorter than 5 pixels or longer than 20 pixels, after which they form document profiles by projecting filtered images on to the horizontal and vertical axes. These profiles reflect the characteristics of the underlying language and can therefore be used for classification. Spitz [5] performs document language identification in two stages. In the first stage, a coarse textual entity is classified into a Han-based script (Chinese, Japanese, and Korean) or a Latin-based script, according to the upward concavity property of each character. In the second stage, character density is calculated for each *textline* to differentiate among the three languages in Han-based documents. For Latin-based documents, word shapes similar to those in [1] are used for further classification. In a different approach, Hochberg et al. [6] extract connected components from document images. They then compute five features for each component, namely, the relative vertical centroid, the relative horizontal centroid, the number of white holes, the number of black pixels in a unit area, and the component’s aspect ratio. Finally, they compute the mean, standard deviation, and skew of these

five features for all components, resulting in a 15-dimensional vector for one textual entity. The Fisher linear discriminant is then computed for these vectors and used to identify the language.

Methods in the third category treat individual characters as independent entities. Thus far, Sanguansat et al. [7] is the only work we have found in this category. They propose a method to distinguish Thai from English characters. Characters are first classified into those with a looped part and those without. In the former type, the looped part occupies a smaller portion in Thai characters than in English characters. In the latter type, cavities and vertical-to-horizontal-gradients are used for classification. However, there are certain Thai and English characters that are rather difficult to discern by either method. In such cases, context information can be used for disambiguation.

Of the above methods, the first category serves as a post-process for OCR, while the other two serve as a pre-process. Also, the first two categories assume the homogeneity of textual entities, while the third category makes no such an assumption. For this reason, the third category is the most useful for dealing with documents in which there are no clues in the layout that can be used to determine language boundaries. This is the case with most multilingual documents. Thus, the problem faced in this category is probably the most difficult to deal with, since – unlike the other two categories – character images are treated as independent entities, rather than joint entities.

We experiment with two machine-learning methods for this purpose: support vector machines (SVM) (Vapnik [9]) and a prototype classification method (Chang et al. [10-11]). Both techniques have proved useful for character recognition applications, especially for the recognition of handwritten digits and Chinese/Japanese characters. When applied to the current problem, both methods achieve high and comparable accuracy rates. This fact qualifies them as language classifiers. The SVM, however, requires longer training and testing times than the other method for the current application, which is characterized by a large training dataset and high imbalance among data of different class types. We thus recommend the prototype classification method for this particular application.

The remainder of this paper is organized as follows. In Section 2, we describe the improved prototype learning method. Section 3 details the necessary computations for obtaining the best SVM solutions. In Section 4, we present a special technique for speeding up the two methods. Section 5 contains the experiment results. Finally, in Section 6, we present our conclusions and the direction of future work.

## 2. The Prototype Classification Method

The prototype classification method we employ in this paper is a modification of the method described in [11]. In the training phase, a learning algorithm is used to construct prototypes from training samples. The general framework of the learning process consists of two loops. The outer loop decides whether the current set of prototypes is sufficient for classifying all training samples properly. If this is the case, the process terminates. Otherwise, the outer loop decides which class types need to have more prototypes constructed, and transfers the problem to the inner loop. The inner loop behaves like a traditional clustering technique. It computes the location of cluster centers for the number of prototypes specified by the outer loop and these centers are taken as our prototypes.

Of the clustering techniques available, we adopt the fuzzy c-means (FCM) method (Bezdek [12]). As indicated in [11], this algorithm yields slightly higher accuracy rates than the K-means (KM) clustering algorithm. There is, however, a problem associated with incorporating FCM into the prototype construction process. It is possible that the process may not terminate within a finite number of iterations. As a remedy for this potential problem, [11] suggests that a sample  $S$  should be abandoned if recruiting  $S$  as a seed to generate new prototypes does not reduce the number of unabsorbed samples. A sample is *absorbed* if it matches in class type with the nearest prototype.

In our application, we have observed that this remedy may cause another problem due to the high imbalance of data. The data-imbalance issue refers to the situation where training samples in some class types outnumber those in other class types. In our application, the number of Chinese samples is 9.3 times the number of English samples, while the number of Japanese samples is 5.3 times the number of English samples. These samples clearly constitute an unbalanced dataset. Thus, in the early stage of prototype construction, prototypes of populous class types may locate further away from class boundaries than those of under-populated class types, and the number of unabsorbed samples may increase transiently with the number of prototypes. At this point, one should allow the creation of more prototypes, instead of abandoning them. Based on this intuitive idea, we modify the prototype construction process as follows. For convenience, we refer to samples of class type  $C$  as  $C$ -samples, and prototypes of class type  $C$  as  $C$ -prototypes. Also, the *range* of a  $C$ -prototype  $\mathbf{P}$  is the set of all  $C$ -samples that find  $\mathbf{P}$  as the nearest  $C$ -prototype.

- Step 1. Initiate a prototype for each class type.
- Step 2. Perform the absorption test. If no more samples are unabsorbed, terminate the process.
- Step 3. Select a  $C$ -sample from the unabsorbed  $C$ -samples for each class type  $C$ .
- Step 4. Use FCM to determine  $n+1$   $C$ -prototypes, using

the newly selected  $C$ -sample and existing  $n$   $C$ -prototypes as seeds. Check if all the  $n+1$   $C$ -prototypes have a non-empty range. If some of them are empty, set the newly selected  $C$ -sample as *futile* and return to Step 3; otherwise, return to Step 2. (Note that a futile sample is no longer taken as an unabsorbed sample.)

The non-emptiness of all prototype ranges guarantees the termination of the process within a finite number of iterations, since – in the worst case – all prototype ranges will shrink to a single-member set, ensuring that all samples (except those that are declared futile) are absorbed. Following [11], we call the above process a fuzzy construction process (FCP).

### 3. Support Vector Machines

SVM is a powerful tool for binary classification. SVM is an attractive technique because, in addition to the optimality of its solutions, it allows users to choose from a variety of options. The many choices, however, require great caution and perseverance on the user's part, since the experimental work can be tedious and may require special techniques to reduce the otherwise excessive amount of computing time. In this section, we list all the relevant options for our application. Then, in the next section, we describe a technique of our own to speedup the training process.

The first option is the type of decomposition we have to make in order to solve our problem, which is to classify an object into three class types. In order to apply it to a multi-classification problem, we have two choices: the one-against-one approach and one-against-others approach (Hsu and Lin [13]).

The second option is the kernel function, which specifies a measure for the similarity between two vectors. The third option is the value of the parameters associated with the kernel functions.

The fourth option is the parameter  $C$ , which specifies the level of tolerance for miss-classification. The purpose of this parameter is to control possible over-fitting in the training process.

To determine the best values for all these choices, we have to go through a cross-validation process, in which we randomly divide the training samples into  $K$  equal-sized parts, called  $K$  folds. We then train an SVM model using  $K-1$  folds as training samples and the remaining fold as test samples. We run this process  $K$  times, each time using a different fold as the test data. We then take the average of the test accuracy rates of all  $K$  runs. This test enables us to determine the average SVM accuracy rate for one combination of options.

### 4. Speed-Up with the Generation of a Re-

## duced Dataset

The size of the training samples for our application is too large. Thus, a speedup technique must be used to curtail the lengthy computation time.

The idea is to initiate a reduced dataset R using some data reduction technique. We then apply FCP to this dataset to determine both the number and the location of prototypes. When FCP process is finished, we pass the prototypes through a validation test to check if all training samples have been absorbed, except, of course, those that have been declared futile. The FCP automatically meets the validation test, if it works on the complete set of training samples. However, since it works only on the reduced dataset, some training samples outside the dataset may be found to be unabsorbed. If this is the case, we add the unabsorbed samples to R and proceed with a new run of FCP on the augmented R. This time, the FCP starts with the prototypes constructed in the previous run as the initial prototypes. The resultant prototypes are then passed through a validation test to see if there are still any unabsorbed samples. With the alternate augmentation of R and prototypes, we will eventually find a set of prototypes that meets the validation test.

How to initiate R requires some explanation. There are several data reduction methods that reduce the set of training samples and still achieve a comparable performance to the  $k$ -NN classification method. Wilson and Martinez [14] provide very detailed descriptions of the algorithms, as well as benchmark performances. Our reason for using a reduction method is not to generate the smallest and best-performing dataset, but to obtain a sufficiently small and reduced dataset at a low computational cost. For this reason, we choose the CNN method, as proposed in Hart [15] and also described in [14].

The final reduced dataset R, from which the FCP constructs prototypes and passes them through the validation test, can also be used to curtail the computational cost of SVM. The idea is to build and solve various SVM problems based on R in order to find the optimal setting that includes the choice of decomposition method, the kernel function, and the parameter values. When the optimal setting has been determined, we proceed to build and solve SVM problems based on the full dataset to obtain the final solutions.

## 5. Experiment Results

For training and testing purposes, we collected a large set of Chinese, Japanese, and English character images from newspapers and magazines. Half of the samples were treated as training samples and the other half as test samples. The number of samples in each language is listed in Table 1. There are 889,846 samples of them in total.

**Table 1. The number of samples in each category.**

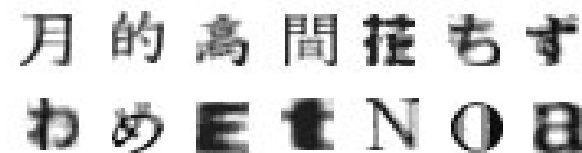
	Chinese	Japanese	English
#Training Samples	249,425	168,623	26,875
#Test Samples	249,425	168,623	26,875

All samples were normalized to a  $64 \times 64$  bitmap, which was further divided into  $16 \times 16$  equal-sized blocks. From these blocks, we obtained a 256-dimensional feature vector. Each feature derived its value from the number of black pixels in one of the 256 blocks. Figure 2 shows some training samples.



**Figure 2. Training samples**

A reduced dataset R, described in Section 4, was used in both FCP and SVM to obtain their final solutions in an acceptable amount of training time. The initial size of R, obtained with the help of CNN, was 3,402. The FCP then worked on this set, augmenting it with as many samples as necessary. At the end of this process, the size of R was 16,128. The size of the full training dataset was 444,923; thus, the reduction rate was 3.6%. In the process of generating R, FCP built 1,137 prototypes, or 7% of R. Figure 3 presents some constructed prototypes expressed in gray-scaled images, where the gray scales represent the feature values.



**Figure 3. Constructed prototypes**

For the SVM method, the best solution was obtained by using the one-against-others approach, with RBF as the kernel function. The optimal  $\gamma$  and  $C$  were found at 0.0001 and 100 respectively. Note that the search range for  $\gamma$  and  $C$  was set to  $\{10^k: k = -5, -4, \dots, 4, 5\}$ . To obtain SVM solutions, we used the software package in [16].

In Table 2, we list the training and test results of FCP and SVM. The computing platform was Intel P4 with 3.4G CUP and 3G RAM. The FCP training time included the time to run CNN, which took 67 minutes. The SVM training time included the time to search for the optimal setting, which took 6 hours and 50 minutes. As mentioned in Section 4, the search for the optimal setting of SVM was carried out with a reduced dataset, obtained with the help of FCP. The number of support vectors obtained by

**Table 2. Training and test results of FFCP and SVM.**

	Training Time	#Support Vectors	#Prototypes	Test Accuracy	Test Time	Computing Speed (Chars/s)
FCP	8hrs 43mins		1137	99.83%	6mins	1235.90
SVM	13hrs 17mins	18,663		99.92%	1hrs 37mins	76.29

SVM was 18,663, or 16 times the prototype size. This large number of support vectors was responsible for the long SVM test time, which was 16 times longer than the FCP test time. Nevertheless, the two methods achieved very close test results. The difference was 0.09%, with the SVM method achieving a slightly better result.

We also came up with a confusion matrix for the three languages, as shown in Table 3. The first row of this table, for example, shows that almost all Chinese characters were correctly classified. Only 225 and 200 of them were miss-classified as Japanese and as English respectively. Miss-classifications are attributed to poor quality of character images and also to the similarity between certain character types in the three languages.

**Table 3. Confusion matrix**

	as Chinese	as Japanese	as English
Chinese	249,000	225	200
Japanese	170	168,393	60
English	50	42	26,783

## References

- [1] N. Nobile, S. Bergler, C. Y. Suen, and S. Khoury, "Language Identification of On-Line documents Using Word Shapes," *Proceedings. Fourth International Conference on Document Analysis and Recognition*, vol. 1, pp. 258 - 262, Aug. 1997.
- [2] T. Pham and D. Tran, "VQ-Based Written Language Identification," *Proceedings. Seventh International Symposium on Signal Processing and Its Applications*, vol. 1, pp. 513 - 516, July 2003.
- [3] T. Nakayama, A.L. Spitz, "European language determination from image," *Proceedings of the Second International Conference on Document Analysis and Recognition*, pp. 159 - 162, Oct. 1993.
- [4] S. L. Wood, X. Yao, K. Krishnamurthi, and L. Dang, "Language Identification for Printed Text Independent of Segmentation," *Proceedings. International Conference on Image Processing*, vol. 3, pp. 428 - 431, Oct. 1995.
- [5] A.L. Spitz, "Determination of the Script and Language Content of Document Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 235-245, March 1997.
- [6] J. Hochberg, K. Bowers, and M. Cannon, "Script and Language Identification for Handwritten Document Images," *International Journal on Document Analysis and Recognition*, vol. 2, pp. 45-52, 1999.
- [7] P. Sanguansat, P. Yanwit, P. Tangwiwatwong, W. Asdornwised, and S. Jitapunkul, "Language-based Hand-printed Character Recognition: A Novel Method using Spatial and

## 6. Conclusion and Future Works

Using two effective methods for multi-class classification, we solve the language identification problem for individual character images. To do this, we use the same methods, and even the same features, used for character recognition, except that in the current application the class types are languages, rather than character categories. As it is not possible to apply these methods to a large training dataset, we also propose a technique for producing a reduced dataset. This is done in the prototype construction process. The comparable test results obtained by these two methods favor our choice of the prototype classification method, since it requires much less time for the test process. Some work remains to be done in this area. For example, the acceleration of prototype matching and a post-process to correct the slight classification errors are two possible directions of future research.

- Temporal Informative Features," *IEEE 13th Workshop on Neural Networks for Signal Processing*, pp. 527-536, Sept. 2003.
- [8] D. Cooper, "How to Read Less and Know More: Approximate OCR for Thai", *Proceedings of the 20th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 216 - 225, Philadelphia, 1997.
- [9] V. Vapnik, *The Nature of Statistical Learning Theory*, New York: Springer Verlag, 1995.
- [10] F. Chang, C-C. Lin, and C-J. Chen, "Applying A Hybrid Method To Handwritten Character Recognition," *Intern. Conf. Pattern Recognition 2004*, vol. 2, pp. 529-532, Cambridge, 2004.
- [11] F. Chang, C-H. Chou, C-C. Lin, and C-J. Chen, A Prototype Classification Method and Its Application to Handwritten Character Recognition, *IEEE Conf. System, Man, and Cybernetics*, pp. 4738-4743, Hague, 2004.
- [12] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.
- [13] C.-W. Hsu and C.-J. Lin, A comparison of methods for multiclass support vector machines, *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415-425, 2002.
- [14] D. R. Wilson and T. R. Martinez, Reduction Techniques for Instance-Based Learning Algorithms, *Machine Learning*, vol. 38, pp. 257-286, 2000.
- [15] P. Hart, The condensed nearest neighbor rule, *IEEE Trans. Information Theory*, pp. 515-516, May 1968.
- [16] [http://www.csie.ntu.edu.tw/Chinese\\_index.html](http://www.csie.ntu.edu.tw/Chinese_index.html).