Rejecting both segmentation and classification errors in handwritten form processing

Claudio De Stefano*, Francesco Fontanella*, Angelo Marcelli†, Antonio Parziale† and Alessandra Scotto di Freca*
*Dipartimento di Ingegneria Elettrica e dell'Informazione (DIEI)
Università di Cassino e del Lazio meridionale, Cassino (FR), Italy
Email: {destefano, fontanella, a.scotto}@unicas.it
†Dipartimento di .Ing. dell'Informazione, Ing. Elettrica e Matematica Applicata (DIEM)
Università di Salerno, Fisciano (SA), Italy
Email: {amarcelli, anparziale}@unisa.it

***Abstract*—The form processing systems commercially available include a verification step during which a human operator verifies the output provided by the system to ensure 100% accuracy. In order to reduce the time and the cost of such a stage, the OCR engine incorporated into the system provides a reliability measure of the classification to be used for implementing a reject option: in this way only rejected samples are passed to the verification stage. Most of the strategies for designing such a reject option consider that the source of classification errors are within the OCR engine. Such an assumption becomes less reasonable as the forms become less structured, as in case when boxes are provided for the entire data field and not only for isolated characters. Under these circumstances, we investigate to which extent the reliability measure provided by an OCR engine designed to deal with boxed isolated characters can be used to detect both segmentation and classification errors. The experimental results, obtained on a large data set of forms currently in use by a large organization, show that the proposed method successfully achieves its aim. It represents a powerful tool for the system manager to plan system enhancement as the volume of forms containing less constrained data fields increases.**

***Keywords*-OCR; Random Forest; Classification Reliability; Reject Option.**

I. Introduction

Handwritten form processing is a routine task performed daily in many organizations for capturing the data to be processed. The core technology behind most of the applications is an OCR engine that associates an internal, computer-readable representation to the images of the characters extracted from the document image. Such representation is eventually converted into the format required by the final application. One of the key issues on determining the success of these systems is, therefore, the recognition rate of the OCR, which depends on both the features and the classification strategy of the OCR engine, as well as on the quality of the input images.

Any of these aspects has been deeply investigated ([1]), but automatically capturing the data with 100% of accuracy is beyond the reach of the solutions that have been proposed so far. For these reasons, the systems that are commercially available for daily use within organizations envisage a verification stage in which a human operator is needed to confirm or modify the output of the OCR engine. The verification stage is obviously the bottleneck of those systems: performing the verification on each output of the classifier dramatically reduces the throughput of the system and raises its operational cost, while unchecked recognition errors may lead to inconsistency in the data provided to the final application. In order to reduce the cost of the verification while ensuring a high level of accuracy, OCR engines are provided with a reject option that aims at detecting the cases when the output of the OCR engine is deemed not reliable, so as to limit the call for verification to just these cases.

Implementing a reject option is not an easy task and many solutions have been proposed in the literature, depending on some estimate of the reliability, or confidence, of the classifier decision ([2], [3], [4], [5], [6]). To this aim, classifiers providing an estimate of the probability that the input sample belongs to the class it has been assigned to are usually considered. This probability is assumed as a measure of classification reliability. From this point of view, the definition of a reject option can be seen as an optimization task, whose aim is that of finding constraints on the reliability value so as to reject the samples whose classification reliability does not satisfy the constraints. The ideal solution is the one that rejects only the specimen leading to recognition errors but none of the ones that can be correctly recognized. In real cases, however, there are regions of the feature space where two or more classes overlap: thus, any attempt to reject misclassified samples also implies the rejection of a number of correctly classified ones.

In [7] the problem of achieving the optimal tradeoff between error and reject rates is addressed from a probabilistic point of view. The author shows that the error rate decreases monotonically as the rejection rate increases. Based on this relationship, the author proves that, if the conditional probability densities are exactly known, then the optimal error versus rejection tradeoff can be found. In real applications, however, these densities are generally unknown. This implies that the implementation of a reject strategy is usually derived by studying the behavior of the reliability measure provided by the classifier on each sample of a training set, in order to effectively set the constraints on its values.

A further problem in handwritten form processing applications, is the fact that samples processed by the OCR engine do not necessarily correspond to isolated characters. In these cases, character images are extracted from the whole image by a segmentation algorithm that exploits layout information on where the characters are within the form layout. In order to reduce segmentation errors, the forms may be designed in such a way that the position of each character is known and fixed, and that different position are well apart from each other, as in cases of boxed forms, i.e. forms where each character is written in a specified area of the form. These are, however, the kind of forms people like to fill the less, because they significantly slow down the writing speed, and are acceptable only when the amount of characters is in the order of a few tens, such as postal order of payment, custom declaration, license plate application and so far.

This is the reason why forms designed to collect larger amount of characters do not provide a box for each character, but rather a box for the entire data field, and request to fill the form by block characters, relying on both the collaboration of the subject to write each characters separately and the segmentation algorithm to separate connected characters. Because there may be non-collaborative subjects, i.e. subjects that fill the form by connecting adjacent characters or even mixing block characters and cursive handwriting, and because character segmentation is an ill-posed problem for which an error free solution may not be designed, it is possible that some of the images extracted from the image form and passed to the OCR engine do not contain a single character.

In the scenario described above, we have investigated to which extent the reliability provided by the OCR engine allow us to implement a reject option that deals simultaneously with both segmentation and classification errors, in order to reject images containing more than one characters or cursive handwriting, as well as images containing single characters that are not correctly recognized. Designing a successful reject option will allow to use an OCR engine for boxed isolated characters to process also boxed data fields: only when the time and the cost for verifying the rejected samples exceeds a break-even point, the option of considering a more powerful recognition engine, explicitly designed to deal with both connected characters and/or cursive handwriting, needs to be considered.

To this aim we have developed a system that combines an OCR engine, based on the Random Forest (RF) algorithm [8], with a validation tool that implements the reject option. For each input sample, RF provides as output the class to which the sample has been assigned to and the probability of that class given the input sample. During the training phase of our system, for each sample of a training set, the output of the RF classifier, in terms of both the class of the input sample and its probability, is provided to the validation tool together with the ”true class” of that sample: using this information, a suitably devised optimization algorithms estimates the class thresholds (one for each class to be recognized) for implementing the reject option. In the operative phase, a classification provided by RF is accepted by the validation tool only if its probability is higher than the value of the threshold associated to the suggested class. In the opposite, the sample is rejected.

We have used the Random Forest algorithm as OCR engine for two main reasons: i) it is a classification method that has shown to be particularly effective for dealing with complex classification problems, outperforming both bagging and boosting [8] [9]; ii) it provides an effective measure of the classification reliability.

The experimental results, obtained on a large data set of forms currently in use by an organization, show that the proposed method successfully achieves its aim, since it rejects more than 95% of the samples corresponding to segmentation errors and more than 70% of misclassified samples corresponding to isolated characters, while reducing the correct recognition rate for samples corresponding to isolated characters less than 15%.

In the following section we illustrate the proposed method, while the experimental work, aimed at evaluating the classification rate, is described next. Eventually, the conclusion discusses the experimental findings and outlines future works.

II. The System Architecture

As mentioned in the Introduction, the proposed system consists of two main modules: (i) an OCR engine; (ii) a validation tool that implements a reject option. The system architecture is shown in Fig. 1.

As concerns the OCR module, it consists of an ensemble of decision trees learned according to Breiman algorithm [8], and commonly known as “Random Forest” (RF). This classifier provides as output the class to which a sample has been assigned to and the probability of that class given the input sample.

The training of the validation tool is performed by using a validation set, statistically independent of the one used to train the OCR module. For each sample of the validation set, the output of the RF, in terms of both the class of the input sample and its probability, is provided to the validation tool, together with the ”true class” of that sample: using this information, a suitably devised optimization algorithms estimates the class thresholds (one for each class to be recognized) for implementing the reject option.

Finally, in the operative phase, for each input sample $\mathbf{x_i}$, the class C_i to which the sample has been assigned to and the corresponding probability A_i, are processed by the validation module, which accepts the classification provided

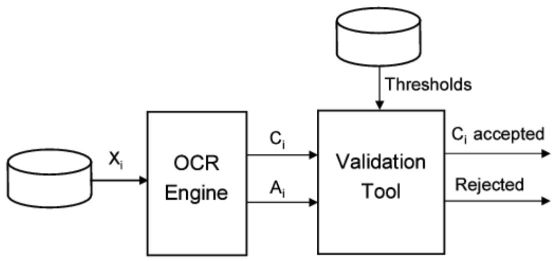

Figure 1. The architecture of the proposed system.

by RF if $A_i \geq T_{C_i}$, where T_{C_i} is the value of the threshold associated to the suggested class. Otherwise, the sample $\mathbf{x_i}$ is rejected.

The descriptions of both modules are reported in the following subsections.

A. Random forest

The term "Random Forest" refers to a general methodology for building an ensemble of L tree-based classifiers. The Random Forest approach does not refer to a single algorithm, but rather to a family of methods. The original algorithm proposed by Breiman in [8] is usually referred in the literature as *Forest-RI* and is used as reference method in most of the papers dealing with RF. Given a training set that contains N feature vectors, each consisting of M features, the forest-RI algorithm consists of the following steps:

1) Draw, from the dataset, N samples at random with replacement. The resulting set will be the training set of the tree.
2) Set a number $K \ll M$.
3) At each node, randomly draw K features from the set of available features.
4) Among the values of each of the K features drawn, choose the best binary split according to the Gini index. Select the feature with the best index value.
5) Grow the tree to its maximum size according to the stopping criterion chosen.
6) Let the tree unpruned.

Note that node splitting (step 5) usually is stopped when one of the following conditions occur: (i) The number of samples in the node to be split is below a given threshold; (ii) all the samples in the node belong to the same class.

Once the forest has been built, an unknown sample is labeled according to the Majority Vote rule: i.e., it is labeled with the most popular class among those provided by the ensemble trees. It is worth noting that in [10] it is proved that RF does not overfit the data as more trees are added, but rather its generalization error tends to a limiting value.

1) Class Probability Estimation in Random Forest: In the RF approach, given a sample $\mathbf{x}$ to be classified, the conditional probabilities for each class are computed by averaging the conditional probabilities given by the trees making up the ensemble. These conditional probabilities are computed as follows. Given an input sample $\mathbf{x_i}$, and a generic decision tree T_j, let us denote by $v(x)$ the leaf node of T_j that assigns a class to x. The probability $P(c|\mathbf{x_i}, T_j)$ that the sample $\mathbf{x_i}$ belongs to the class c $\ (c \in \{1, 2, \ldots, C\})$, is estimated by the following equation:

$$P(c|\mathbf{x_i}, T_j) = \frac{n_c}{n} \tag{1}$$

where n_c is the number of training samples belonging to class c that have been classified by $v(x)$ in T_j, while n is the total number of training samples classified by $v(x)$ in T_j. Both numbers are obtained at the end of the learning procedure of RF.

Given a RF consisting of L trees and an unknown sample $\mathbf{x_i}$ to be classified, the probability $P(c|\mathbf{x_i})$ that $\mathbf{x_i}$ belongs to the class c can be estimated as follows:

$$P(c|\mathbf{x_i}) = \frac{1}{L}\sum_{j=1}^{L} P(c|\mathbf{x_i}, T_j) \tag{2}$$

As a consequence, the RF algorithm gives as output the vector:

$$\mathbf{p} = \{P(1|\mathbf{x}), P(2|\mathbf{x}), \ldots, P(C|\mathbf{x})\} \tag{3}$$

Eventually, the final output of RF is the pair (C_i, A_i), where C_i is the label of the class corresponding to the highest value in the output vector $\mathbf{p}$ of eq. (3), and $A_i = P(C_i|\mathbf{x_i})$.

B. The Validation Tool

As previously mentioned, the purpose of the validation tool is to reject the samples which have been erroneously classified by the OCR module. As discussed in [11], the use of multiple class-related reject thresholds (one for each class to discriminate) can provide an error-reject tradeoff better than that obtainable by applying the Chow's reject rule, which uses a single reject threshold. This is the reason why we considered 52 class-related reject thresholds in our reject rule.

The problem of finding the optimal values for the reject thresholds is computationally very complex: in our case, encoding each threshold value by means of a bit string of 10 bits, so as to obtain a reasonable precision for representing a real value in the range $[0.0, 1.0]$, a candidate solution composed of a bit string of 520 bits would be obtained. This means that the optimization algorithm should find the optimal solution in a search space whose cardinality is 2^{520}.

In order to reduce the complexity, we have reformulated the problem of finding the whole set of N_c thresholds (N_c is the total number of classes to be discriminated) in N_c sub-problems in which a single threshold must be found. In particular, given a dataset $\mathcal{D}$, the optimal value of each class-related threshold T_{C_i} is searched by considering only the responses of the OCR module providing as output the class C_i: for each response, both the associated probability and the knowledge about the "true class" of the corresponding sample, are used to optimize a suitably defined objective function.

The rationale of this choice is twofold. On the one hand it guarantees to find the global optimum: since each threshold T_{C_i} has effect only on the responses of the OCR module providing as output the class C_i, its optimal value can be searched without considering neither the other responses, nor the other threshold values. On the other hand, it allows us to exhaustively search the optimal solution for each threshold: in this case, in fact, a candidate solution represents the threshold value by a bit string of 10 bits and the cardinality of the search space is only 2^{10}.

As regards the objective function, it has been defined as follows. Given a dataset $\mathcal{D}$, a class C_i and a threshold T_{C_i}, the objective function $f_o(C_i, T_{C_i}, \mathcal{D})$ assumes the form:

$$f_o(C_i, T_{C_i}, \mathcal{D}) = \frac{N_{cr} + N_{ea} + N_{sa}}{N_{C_i}} \tag{4}$$

where N_{C_i} is the number of samples in $\mathcal{D}$ for which the OCR module provides as output the class C_i and the three terms in the numerator have the following meaning:

- N_{cr}: the number of character samples in $\mathcal{D}$ belonging to the class C_i that have been correctly classified by the OCR module with a reliability value lower than T_{C_i}; these samples would be rejected by the validation tool;
- N_{ea}: the number of character samples of $\mathcal{D}$ belonging to other classes that have been erroneously classified by the OCR module as class C_i, with a reliability value higher than T_{C_i}; these samples would be accepted by the validation tool;
- N_{sa}: the number of samples of $\mathcal{D}$ representing segmentation errors that have been assigned to the class C_i by the OCR module with a reliability value higher than T_{C_i}; these samples would be accepted by the validation tool.

Thus, the optimization algorithm implemented in the validation tool can be formulated as follows: for each class C_i, find the value $\hat{T_{C_i}}$, which solve the minimization problem

$$\min_{0 \leq T_{C_i} \leq 1} f_o(C_i, T_{C_i}, \mathcal{D}) \tag{5}$$

III. Experimental Results

In the experiments, we used a large data set of forms currently in use by an organization. The forms contain boxes for the entire data field and writers were requested to fill the form by block characters, using both uppercase and lowercase letters. After the digitalization, a segmentation algorithm was applied to each image to extract sub-images containing single characters. In particular, the connected components of the ink, corresponding to pieces of ink produced without lifting the pen, were extracted from the data fields of each image and processed in order to identify those corresponding to parts of characters: such components must be merged to obtain the actual characters produced the writers. This is the case, for instance, of pieces of ink representing horizontal or vertical bars in characters such as 'T' or 'F'. The components obtained at the end of this step constitute the samples to be processed by the OCR engine.

In order to assess the performance of our system, we have manually labeled the samples adding a further class for representing segmentation errors, i.e. images that correspond to pieces of ink containing more than one characters or cursive handwriting. Thus, we have 52 classes to be discriminated (uppercase and lowercase letters) and a further class (denoted as "cursive") which corresponds to segmentation errors. The data are strongly unbalanced, with classes having hundreds of samples and classes having just few tens.

The samples were represented by using the features proposed in [12]. Each sample image was divided into six parts and was described by a feature vector containing measures associated to the different parts. For each part, 22 features were computed, for a total of 132 features: 13 of these features represent concavity measures, while the remaining ones are related to other contour information.

The samples were arranged into three statistically independent datasets. For each of them, the number of samples per class has been determined so as to resemble the class distribution of the samples of the whole database. The dataset TR1 contains 5369 samples, includes only isolated characters and was used to train the RF classifier. The dataset TR2 contains 10355 samples, includes both isolated characters and "cursive" (5210 characters and 5145 segmentation errors) and was used for the training phase of our system, i.e. for setting the threshold values. Finally, the dataset TS contains 10358 samples, includes both isolated characters and "cursive" (5212 characters and 5146 segmentation errors) and was used for testing our system. As regards the learning of the OCR module, we trained 20 RF classifiers with different random seeds: the results reported in the following have been obtained by averaging those relative to the 20 RF classifiers.

In a first set of experiments, we considered the optimization problem of eq. (5). The results obtained on TS are reported in Fig. 2. The OCR engine gets a correct recognition rate on character samples equal to 72.5% but, obviously, an error rate equal to 100% on "cursive" samples. In this case, the introduction of the reject option implemented in the validation tool, allowed us to reject the large majority of segmentation errors (more than 95%), and the majority of misclassified characters (more than 70%). It also implies the rejection of about 20% of characters correctly classified by the OCR engine.

Summarizing, the introduction of the reject option has a negative effect on the character recognition rate, reduced of about 14%, but it also has a very positive effect on the global performance of the system, which shows an error

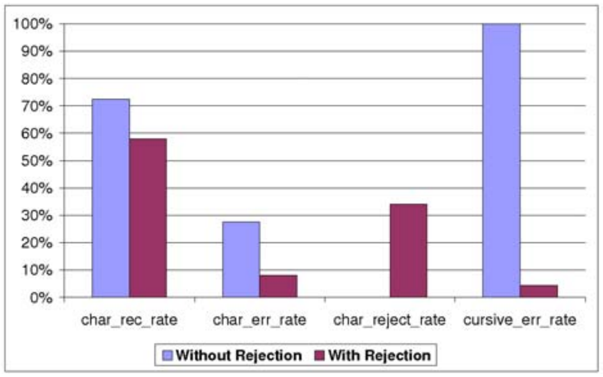

Figure 2. The results on TS with and without the reject option.

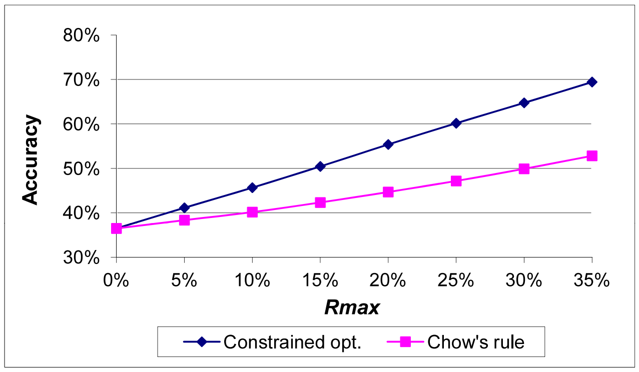

Figure 3. The accuracy on TS as a function of R_{max}, obtained by using both our reject rule and the Chow's reject rule.

rate on character samples lower than 8% and an error rate on "cursive" samples lower than 5%. The reject rate for samples corresponding to isolated character is about 34%.

In order to better evaluate the effectiveness of our approach, we have also applied the Chow's reject rule. As anticipated in the Introduction, this rule provides the optimal error-reject tradeoff, under the assumption that the a posteriori probabilities are exactly known. We conjecture that the reliability measures provided by RF are a good estimate of such probabilities: thus, we used them in a second set of experiments for implementing the Chow's reject rule and for comparing the obtained results with that of our reject rule. To this aim, we used the same dataset (TR2) to estimate both the values of Chow's reject threshold, and those of our thresholds, and considered a range of reject rates from 5% to 35%. This choice is motivated by the fact that we obtained, for the unconstrained optimization problem, an overall reject rate on isolated characters of about 34% and thus, constraints on the rejection rate higher than this value would have no effect.

The comparison with the Chow's reject rule requires to compute the accuracy, defined as the percentage of correctly classified patterns with respect to the total number of accepted ones, for each reject rate in the considered range. Such requirement implies the introduction of a constraint in our optimization algorithm, so as to assure that the overall reject rate be lower than the allowed maximum value. To this aim, we have reformulated the optimization problem of eq. (5) as follows: for each class C_i, find the value $\hat{T_{C_i}}$, which solve the constrained minimization problem:

$$\begin{cases} \min f_o(C_i, T_{C_i}, \mathcal{D}) \\ R(C_i, T_{C_i}, \mathcal{D}) < R_{max} \end{cases} \tag{6}$$

where $R(.)$ is the reject rate obtained by using the threshold T_{C_i} and R_{max} is the overall reject rate allowed on isolated characters.

Fig. 3 shows the accuracy on TS, as a function of R_{max}, obtained by using both our reject rule and the Chow's reject rule. For any value of R_{max}, the accuracy of our system is higher than that of the Chow's rule, and the performance increment becomes more relevant as R_{max} increases.

Finally, it is worth noticing that the accuracy obtained by using the threshold values of the unconstrained optimization algorithm is equal to 82.53%: this value is considerably higher than that obtained by the constrained optimization algorithm in similar conditions, i.e. with an almost identical value of the rejection rate. This result can be explained considering that, in case of unconstrained optimization, the reject rate on isolated character represents the average value over all the classes. On the contrary, in case of constrained optimization, the constraint on the maximum reject rate is applied to the computation of each threshold and, thus, it must be satisfied by each class to be discriminated. In other words, in the first case the constraint is more relaxed and this allows the optimization algorithm to provide better results.

Fig. 4 reports the recognition rate, the error rate and the reject rate as a function of R_{max}, for samples corresponding to isolated characters (the percentages refer to the total number of character samples). Similarly, Fig. 5 reports the trend of the error rate for "cursive" samples (the percentage refer to the total number of "cursive" samples). In both figures, the first bar corresponds to the case in which the reject rule has not been applied, while the last bar reports the results of the reject option with the unconstrained optimization.

The data reported in the figures show that the recognition rate slightly decreases as R_{max} increases, while the error rate, on both isolated characters and "cursive" samples, exhibits a more sharp decreasing trend. These results confirm the effectiveness of our system: as R_{max} increases, the overall increment of the rejection rate is largely due to samples misclassified by the OCR engine. This behavior is more evident for the error rate on "cursive" samples, where even small values of R_{max} allow the system to reject many of the segmentation errors: for $R_{max} = 10\%$ about 35% of "cursive" samples are rejected, while for $R_{max} = 15\%$ the percentage is about 50%.

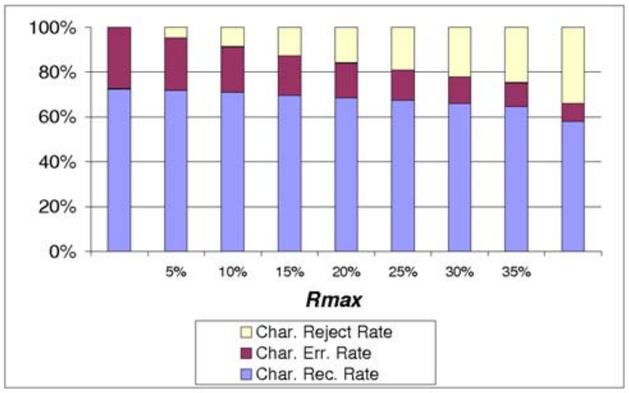

Figure 4. Results for characters samples, as a function of R_{max}. The first bar reports the results without reject rule, while the last bar reports the results of the reject rule with unconstrained optimization.

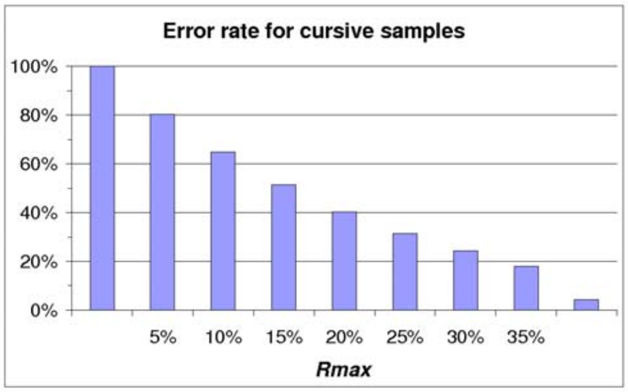

Figure 5. The error rate for "cursive" samples, as a function of R_{max}. The first bar reports the results without reject rule, while the last bar reports the results of the reject rule with unconstrained optimization.

IV. Conclusions

In the framework of handwritten form processing, we have investigated to which extent the reliability provided by an OCR engine, designed to deal with boxed isolated characters, can be used to implement a reject option able to detect both segmentation and classification errors. The OCR engine has been developed by adopting the RF paradigm, while the reject option is based on the use of as many thresholds as the classes to be discriminated. The values of such thresholds have been determined by a suitably designed optimization algorithm, which exploits the result of the OCR engine, in terms of both the class of the input sample and its probability, as well as the knowledge about the actual class of that sample. The experimental results, obtained on a large data set of forms, confirmed the effectiveness of our system, allowing the rejection of the large majority of samples corresponding to segmentation errors (more than 95%), the majority of samples corresponding to characters misclassified by the OCR engine (more than 70%), with a limited reduction (less than 15%) of the character recognition rate.

These results are particularly meaningful considering that in our experiments we have included a number of samples representing segmentation errors almost equal to that of samples corresponding to isolated characters. This very unfavorable condition represents a severe test bed for our system since, in real applications, it is expected that the percentage of segmentation errors is significantly lower than that of correct segmentations.

In a future work we will try to characterize the behavior of our system as the number of segmentation error varies.

References

[1] R. Plamondon and S. Srihari, "On-line and off-line handwriting recognition: A comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 2, pp. 63–84, 2000.

[2] L. Cordella, C. De Stefano, F. Tortorella, and M. Vento, "A method for improving classification reliability of multilayer perceptrons," *Neural Networks, IEEE Transactions on*, vol. 6, no. 5, pp. 1140 –1147, 1995.

[3] C. De Stefano, C. Sansone, and M. Vento, "To reject or not to reject: that is the question-an answer in case of neural classifiers," *Systems, Man, and Cyber. Part C, IEEE Transactions on*, vol. 30, no. 1, pp. 84–94, 2000.

[4] T. Landgrebe, D. M. J. Tax, P. Paclík, and R. P. W. Duin, "The interaction between classification and reject performance for distance-based reject-option classifiers," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 908–917, 2006.

[5] B. Hanczar and E. R. Dougherty, "Classification with reject option in gene expression data," *Bioinformatics*, vol. 24, no. 17, pp. 1889–1895, 2008.

[6] A. Koerich, "Rejection strategies for handwritten word recognitions," in *IWFHR-9*. IEEE Computer Society Press, 2004, pp. 479–484.

[7] C. Chow, "On optimum recognition error and reject trade off," *IEEE Trans. Inf. Theor.*, vol. 16, no. 1, pp. 41–46, 2006.

[8] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[9] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[10] L. Breiman, "Bagging Predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[11] G. Fumera, F. Roli, and G. Giacinto, "Reject option with multiple thresholds," *Pattern Recognition*, vol. 33, pp. 2099–2101, 2000.

[12] L. Oliveira, R. Sabourin, F. Bortolozzi, and C. Suen, "Automatic recognition of handwritten numerical strings: A recognition and verification strategy," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1438–1454, 2002.