IJDAR (2004) 6: 248–262
Digital Object Identifier (DOI) 10.1007/s10032-003-0114-z

International Journal on Document Analysis and Recognition
IJDAR

Segmentation and recognition of handwritten dates: an HMM-MLP hybrid approach

Marisa Morita[1,2], **Robert Sabourin**[1,2,3], **Flávio Bortolozzi**[3], **Ching Y. Suen**[2]

[1] École de Technologie Supérieure, Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA), Montreal, Canada
[2] Centre for Pattern Recognition and Machine Intelligence (CENPARMI), Montreal, Canada
[3] Pontíficia Universidade Católica do Paraná (PUCPR), Curitiba, Brazil

Received: 17 December 2002 / Accepted: 16 July 2003
Published online: 17 November 2003 –

Abstract. This paper presents an HMM-MLP hybrid system for segmenting and recognizing complex date images written on Brazilian bank checks. Through the recognition process, the system makes use of an HMM-based approach to segment a date image into subfields. Then the three obligatory date subfields (day, month, and year) are processed. A neural approach has been adopted to decipher strings of digits (day and year) and a Markovian strategy to recognize and verify words (month). The final decision module makes an accept/reject decision. We also introduce the concept of metaclasses of digits to reduce the lexicon size of the day and year and improve the precision of their segmentation and recognition. Experiments show interesting results on date recognition.

Keywords: Date processing – Metaclasses – Hidden Markov models – Neural networks – Segmentation – Recognition and verification

1 Introduction

Automatic handwriting recognition has been a topic of intensive research during the last decade. The literature contains many studies on the recognition of isolated units of writing such as characters, words, or strings of digits. Only recently has the recognition of a sentence composed of a sequence of words or different data types been investigated. Some typical applications on sentence recognition are reading texts from pages [4,16,17], street names from postal addresses [13,22,23,25,29], and processing of legal amounts [7,12] and dates [26,28] on checks. In such applications, usually a sentence is segmented into its constituent parts. In the literature, two main approaches of segmentation can be observed. The first and perhaps most frequently used method segments a sentence explicitly into parts usually based on an analysis of the geometric relationship of adjacent components in an image. However, this strategy shows its limits rapidly when the correct segmentation does not fit with the predefined rules of the segmenter, e.g., handwritten sentences that do not have uniform spacing. The second method treats complete handwritten sentences as single units, i.e., the segmentation is integrated into, and obtained as a byproduct of, the recognition process. Nevertheless, this approach requires more computational effort than the first one.

Correspondence to: Marisa Morita
(e-mail: marisa@livia.etsmtl.ca)

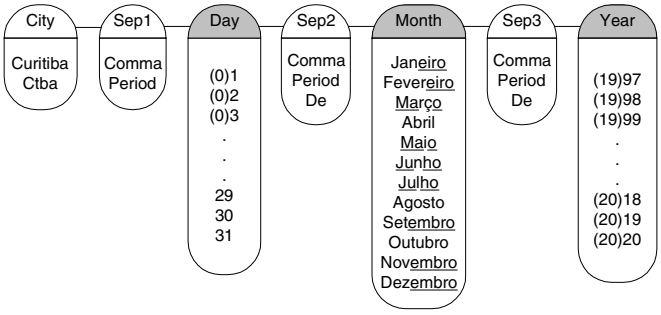

Fig. 1. Lexicon of each date subfield (the common substrings are given emphasis by *underlines*)

In this paper, we present a hybrid system that uses the hidden Markov models (HMMs) and the multilayer perceptron (MLP) to segment and recognize unconstrained handwritten dates on Brazilian bank checks. Through the recognition process, it segments a date into subfields. In such an application, the date from left to right can consist of the following subfields: city name, first separator (Sep1), day, second separator (Sep2), month, third separator (Sep3), and year. Figure 1 details the lexicon of each date subfield present in the laboratory database, and Fig. 2 shows some samples of handwritten dates. In both images, the gray color represents the obligatory date subfields (day, month, and year).

The development of an effective date processing system is very challenging because it has to tackle many levels of complexity. The system must be omniwriter and

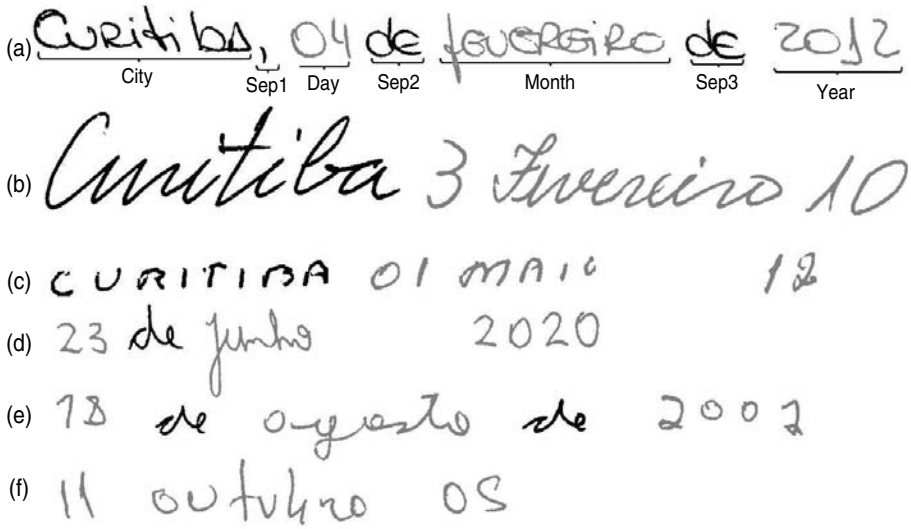

Fig. 2. Samples of handwritten dates on Brazilian bank checks

consider different data types such as strings of digits and words written in different styles (uppercase, lowercase, and mixed). As shown in Fig. 1, although the lexicon size of month words is limited, there are some classes such as "Setembro" (September), "Novembro" (November), and "Dezembro" (December) that contain a common substring ("embro") and can affect the performance of the recognizer. The system must also take into account the variations present in the date field such as 1- or 2-digit day, 2- or 4-digit year, the presence or absence of the city name, and separators. Moreover, it must deal with difficult cases of segmentation since there are handwritten dates where the spaces between subfields (intersubfield) and within a subfield (intrasubfield) are similar as shown in Figs. 2b and 2c. For example, in Fig. 2b, the intrasubfield space between "1" and "0" is almost the same as the intersubfield spaces between "Curitiba" and "3" or "Fevereiro" and "10". Therefore, it will be very difficult to detect the correct intersubfield spaces in this image using a rule-based segmentation method.

Due to the complexity of processing the whole date at the same time, the system discussed in this paper first segments the date subfields and then recognizes the three obligatory ones (day, month, and year) using specialized classifiers. Since separating a date into subfields without resorting to recognition is a difficult task, HMMs have been employed to identify and segment such subfields. We propose to use MLP neural networks to deal with strings of digits (day and year) and the HMMs to recognize and verify words (month). This is justified by the fact that MLPs have been widely used for digit recognition, and the literature shows better results using this kind of classifier [8,21] than HMMs [2]. On the other hand, HMMs have more recently been successfully applied to handwritten word recognition [18,29]. Finally, the last decision module makes an accept/rejection decision.

The development of a hybrid system using HMMs and MLPs is not a new concept for word and digit string recognition, respectively. However, the main contribution of this work focuses on three aspects. The first one lies in the HMM-based approach for separating date subfields. It makes use of the concept of metaclasses of digits to reduce the lexicon size of the day and year and produce a more precise segmentation. The second important feature of this work is the scheme adopted to reduce the lexicon size on digit string recognition to improve the recognition results. Such a strategy uses the output of the HMMs in the form of digit string length, i.e., the information on the number of digits present in a string (day or year), and the metaclasses of digits. Finally, the last aspect is the word verification scheme. We show how important the role of the word verifier is in the system. Experiments show encouraging results on date recognition. In addition to the date database, we have used the NIST SD19 database to validate the strategy employed in digit string recognition. This paper is an enhancement and extension of the work presented in [20].

The paper is organized as follows. Section 2 provides a brief overview of the system. Section 3 presents the definitions related to metaclasses of digits. HMMs and MLPs used in the system are also defined in this section. Section 4 describes each module of the system, and Sect. 5 reports the results of experiments carried out. Finally, Sect. 6 presents some concluding remarks.

2 System overview

Owing to the complexity of this kind of system, it was divided into several modules where each one assumes specific functions to facilitate the construction of the system. Figure 3 shows the modules of the system: segmentation into subfields, digit string recognition, word recognition and verification, and final decision. In Sect. 4, we give a more in-depth description of these modules comprising our date recognition system.

The system discussed in this paper receives a binary date image as input. It takes an HMM-based strategy for separating the date subfields since we have seen that this is a difficult task without resorting to recognition.

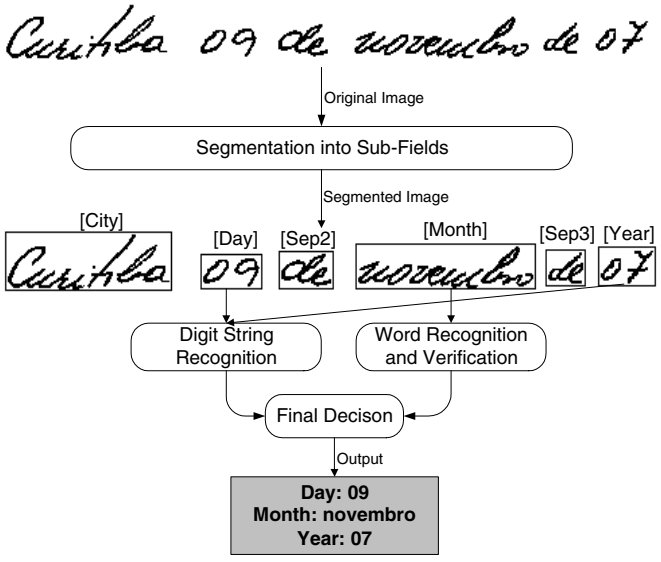

Fig. 3. Block diagram of the date recognition system

After the date subfields have been identified through the HMMs, we can recognize the three obligatory ones (day, month, and year) using specialized classifiers according to their respective known data types. In such cases, an MLP approach has been used to deal with strings of digits (day and year) and an HMM strategy to recognize and verify words (month). The system also contains a final decision module that makes an accept/rejection decision.

3 Definitions

In this section, the definitions related to the date recognition system are introduced. Section 3.1 presents the definition related to the metaclasses of digits. Sections 3.2 and 3.3 describe, respectively, the HMMs and MLPs used in the HMM-MLP hybrid system.

3.1 Metaclasses of digits

We have defined four metaclasses of digits ($C_{0,1,2,3}$, $C_{1,2}$, $C_{0,9}$, and $C_{0,1,2,9}$) based on the classes of digits present in each position of a 1- or 2-digit day and a 2- or 4-digit year as shown in Fig. 4. A metaclass of digits is formed by the union of two or more of the original classes to break down the complexity of their segmentation and the following recognition processes [6,14]. This is possible because the lexicon of the day and year is known and limited, as we observe in Fig. 1. While the class of digits C_{0-9} deals with the ten numerical classes, the metaclasses of digits represented by the shaded boxes in Fig. 4 work with specific classes of digits. The objective is to build HMMs based on these metaclasses in order to reduce the lexicon size of the day and year and improve the precision of their segmentation. In addition, it can be applied to digit string recognition to improve the recognition results since very often confusions between some classes of digits

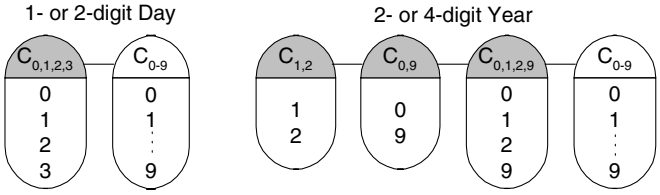

Fig. 4. Classes of digits present in each position of a 1- or 2-digit day and a 2- or 4-digit year

can be avoided (e.g., 4 and 9, 8 and 0). We can observe the efficiency of this strategy in Sect. 5.

3.2 Hidden Markov models

HMMs [24] are finite stochastic processes that have been proven to be one of the most powerful tools for modeling speech and a wide variety of other real-world signals. These probabilistic models offer many desirable properties for modeling characters, words, or sentences. One of the most important properties is the existence of efficient algorithms to automatically train the models without any need for labeling presegmented data. This constitutes a key feature of the approach we developed for segmentation into subfields, word recognition, and verification. In our study, we are dealing with the discrete HMMs. Furthermore, we are explicitly segmenting each image into segments (graphemes) based on cut rules for feature extraction. Explicit methods are more efficient here than in online handwriting recognition or speech recognition due to the bidimensional nature of images and the overlap between characters. Indeed, vertical sampling using sliding windows loses the sequential aspect of the strokes, which is better represented by explicit methods.

As pointed out earlier, a date image can consist of city name, separators (Sep1, Sep2, and Sep3), day, month, and year subfields. Due to the huge size of the vocabulary of dates, our elementary HMMs are built at city, space, and character levels. Thus, the word, date, and date subfield models are formed by the concatenation of appropriate elementary HMMs. In the following sections, we explain the justification behind the proposed models. In Sect. 3.2.1, we present the elementary HMMs used in the system, and in Sect. 3.2.2, we describe the word models employed in word recognition and verification. Finally, in Sect. 3.2.3, we report the date and its subfield models used in segmentation.

3.2.1 Elementary models. In this section, we introduce the topologies of the elementary HMMs used in the system. Such models are built at city, space, and character levels and are employed in the following modules of the system: segmentation into subfields, word recognition, and verification.

The model depicted in Fig. 5a was adopted to represent all the city names and noise (e.g., Sep1). The objective of this model is to identify them in the sentence.

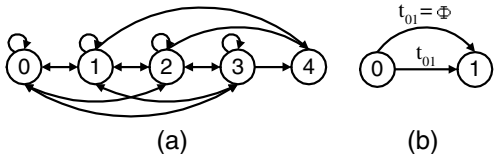

Fig. 5. a Topology of the city model and **b** the space models

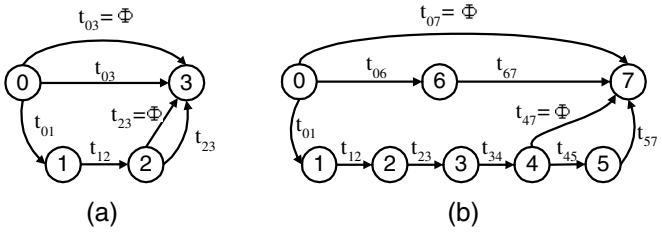

Fig. 6. a,b Topologies of the character models

A left–right topology has been used to model spaces and characters such as digits and letters. The topology of the space models is very simple, as shown in Fig. 5b. It consists of two states linked by two transitions that encode a space (transition t_{01}) or no space (transition $t_{01} = \Phi$). Besides the intersubfield model, we have defined two more HMMs that model the intradigit and intraword spaces.

Two topologies of character models were chosen based on the output of our segmentation algorithm, which may produce a correct segmentation of a character, a character undersegmentation, or a character oversegmentation into two, three, or four graphemes depending on each character (a detailed description of the segmentation algorithm is given in [19]). To cope with these configurations of segmentations, we have designed topologies with three different paths leading from the initial state to the final state. Figures 6a and 6b show examples of both topologies.

The model in Fig. 6a is employed in the segmentation and word recognition modules. In such a topology, the transition t_{03} either (a) models undersegmentation and emits the null symbol Φ or (b) models a character correctly and emits a symbol. The transitions t_{01}, t_{12}, and t_{23} model character segmentation into two or three graphemes. The model in Fig. 6b is used in the word verification module and is based on the previous one, but in this case the model has transitions (t_{12}, t_{34}, t_{57}, and t_{67} of Fig. 6b) that encode the way the graphemes are linked together, e.g., if there is a segmentation point, we can encode whether its vertical position is closer to the upper or lower baselines. Considering uppercase and lowercase letters, we need 40 models since the month alphabet is reduced to 20 letter classes and we are not considering the unused ones. Thus, regarding the two topologies, we have 80 HMMs. For the digit case, we have defined five HMMs based on the topology of the character model shown in Fig. 6a. One model considers the ten numerical classes (M_{0-9}), and the other ones are defined based on the metaclasses of digits (e.g., the $M_{1,2}$ model copes with the class of digits $C_{1,2}$ and so forth) (Fig. 4).

Table 1. Description of the elementary HMMs used in the system. (SSF – segmentation into subfields, WR – word recognition, and WV – word verification)

Model	Topology	Classes	Usage
M_{city}	Fig. 5a	Curitiba and Ctba	SSF
M_{space}	Fig. 5b	Intersubfield space	SSF
M_{space}^{digit}	Fig. 5b	Intradigit space	SSF
M_{space}^{word}	Fig. 5b	Intraword space	SSF,WR
M_a	Fig. 6a	a	SSF,WR
M_A	Fig. 6a	A	SSF,WR
⋮	⋮	⋮	⋮
M_v	Fig. 6a	v	SSF,WR
M_V	Fig. 6a	V	SSF,WR
$M_{a'}$	Fig. 6b	a	WV
$M_{A'}$	Fig. 6b	A	WV
⋮	⋮	⋮	⋮
$M_{v'}$	Fig. 6b	v	WV
$M_{V'}$	Fig. 6b	V	WV
$M_{0,1,2,3}$	Fig. 6a	0,1,2 and 3 ($C_{0,1,2,3}$)	SSF
M_{0-9}	Fig. 6a	0-9 (C_{0-9})	SSF
$M_{1,2}$	Fig. 6a	1 and 2 ($C_{1,2}$)	SSF
$M_{0,9}$	Fig. 6a	0 and 9 ($C_{0,9}$)	SSF
$M_{0,1,2,9}$	Fig. 6a	0,1,2 and 9 ($C_{0,1,2,9}$)	SSF

Therefore, considering one-city, three-space, 80-letter, and five-digit models, the system takes into consideration 89 elementary HMMs that were trained using the Baum-Welch algorithm with the cross-validation procedure [11]. Table 1 describes those models as well as where they are used in the system.

3.2.2 Word models. Basically, the word models consist of the concatenation of the foregoing topologies. Since no information on word recognition is available on the handwritten style (uppercase, lowercase), we propose to use the architecture of the word model shown in Fig. 7a, which can deal with the problem of mixed handwritten words. This architecture consists of an initial state (I) and a final state (F), two elementary letter HMMs in parallel, and four elementary intraword space HMMs linked by four transitions: two uppercase letters (UU), two lowercase letters (LL), one uppercase letter followed by one lowercase letter (UL), and one lowercase letter followed by one uppercase letter (LU). The probabilities of these transitions are estimated by their occurrence frequency in the training set. In the same manner, the probabilities of beginning a word by an uppercase letter (0U) or a lowercase letter (0L) are also estimated in the training set.

Figure 7a shows the architecture of the word models adopted in word recognition, while Fig. 7b illustrates the architecture used for word verification. We can observe that the architecture shown in Fig. 7b diverges only in two aspects: it does not consider the space models and its elementary HMMs contain transitions that encode the way the graphemes are linked together.

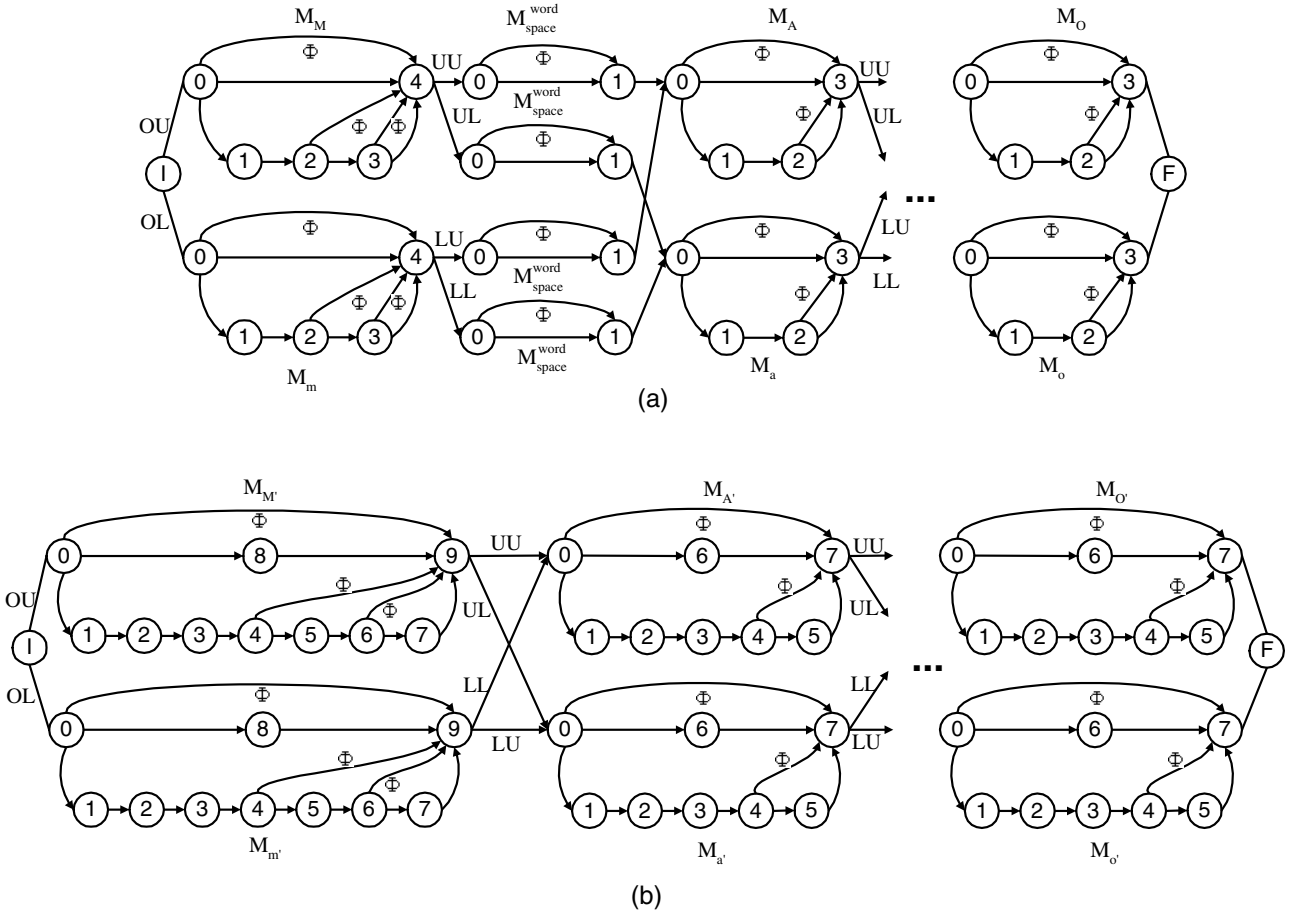

Fig. 7a,b. Word models of class "Maio" (May): **a** recognition and **b** verification

Table 2. Description of the date subfield models

Model	Description
M_{city}	City model (plus Sep1)
M_{day}	1- or 2-digit day model
M_{de}	"De" separator model (Sep2 and Sep3)
M_{month}	Month model (12 word classes)
M_{year}	2- or 4-digit year model

3.2.3 Date and its subfield models. Based on the fact that during the process of segmenting the image into subfields we do not want to recognize the content of each subfield, i.e., their identification is sufficient, the date model has been simplified by reducing the date lexicon size (Fig. 1) to improve the segmentation results.

This is performed by considering one model for representing all the city names and the first separator (Sep1), and the same "De" model for representing the second and third separators (Sep2 and Sep3). In addition, the concept of metaclasses of digits described in Sect. 3.1 has been used to build the day and year models. Figures 8a and 8b show, respectively, the date models without and with lexicon size reduction of an image with all the subfields, e.g., the image presented in Fig. 2a.

We can observe from Fig. 8b that the lexicon size of month words has not been reduced since it can help to identify the other subfields. In addition, the date model shown in Fig. 8b represents an image with all the subfields. Considering that some subfields are optional and there is one model for each subfield, we have eight possible date models that are created by concatenating appropriate subfield models and by considering the intersubfield space as a model. Such date models act as a segmentation engine of the date into subfields. The date subfield models are described in Table 2, and the eight date models are presented in Table 3, where M_{space} stands for the intersubfield space model defined in Table 1.

The month model consists of an initial state (I), a final state (F), and 12 models in parallel that represent the 12 word classes. The architecture of each word model and the "De" separator model are the same as those shown in Fig. 7a. The day model consists of an initial state (I), a final state (F), and the 2-digit day model in parallel with the 1-digit day model as shown in Fig. 9. The 2-digit day model is formed by concatenating the following elementary models shown in Table 1: $M_{0,1,2,3}$, M_{space}^{digit}, and M_{0-9}. The 1-digit day model is related to the M_{0-9} model. The probabilities of being a 1- (1D) or 2-digit (2D) day are estimated in the training set of the date database. The year model is built in the same manner.

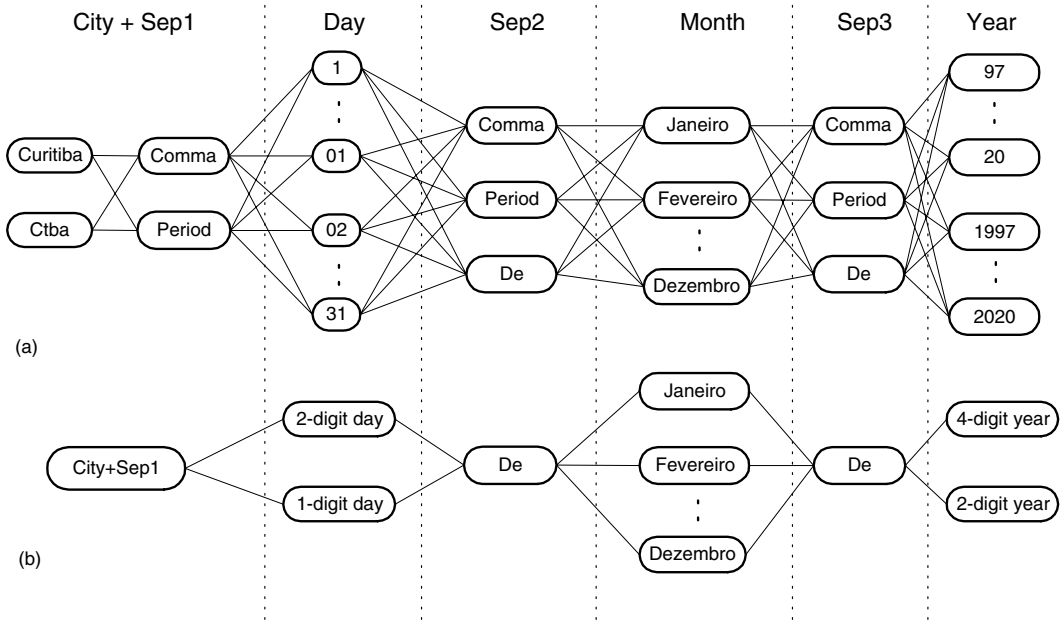

Fig. 8a,b. Date models: **a** without lexicon size reduction and **b** with lexicon size reduction

Table 3. Date models

No.	Date model										
1	M_{city}	M_{space}	M_{day}	M_{space}	M_{de}	M_{space}	M_{month}	M_{space}	M_{de}	M_{space}	M_{year}
2			M_{day}	M_{space}	M_{de}	M_{space}	M_{month}	M_{space}	M_{de}	M_{space}	M_{year}
3	M_{city}	M_{space}	M_{day}	M_{space}	M_{de}	M_{space}	M_{month}	M_{space}			M_{year}
4			M_{day}	M_{space}	M_{de}	M_{space}	M_{month}	M_{space}			M_{year}
5	M_{city}	M_{space}	M_{day}	M_{space}			M_{month}	M_{space}	M_{de}	M_{space}	M_{year}
6			M_{day}	M_{space}			M_{month}	M_{space}	M_{de}	M_{space}	M_{year}
7	M_{city}	M_{space}	M_{day}	M_{space}			M_{month}	M_{space}			M_{year}
8			M_{day}	M_{space}			M_{month}	M_{space}			M_{year}

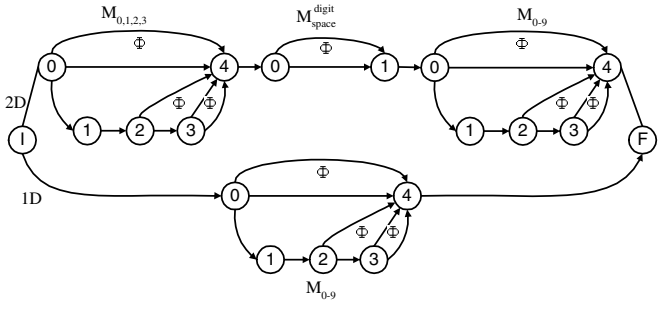

Fig. 9. Day model for 1- or 2-digit strings

3.3 Neural networks

Although many types of neural networks can be used for classification purposes [15], we opted for an MLP, which is the most widely studied and used neural network classifier. Therefore, all classifiers presented in this work are MLPs trained with the gradient descent applied to a sum-of-squares error function [1]. The transfer function employed is the familiar sigmoid function.

To monitor the generalization performance during learning and terminate the algorithm when the improvement levels off, we have used the method of cross validation. Such a method takes into account a validation set that is not used for learning to measure the generalization performance of the network. During learning, the performance of the network on the training set will continue to improve, but its performance on the validation set will improve only to the point where the network starts to overfit the training set, indicating that the learning algorithm is terminated.

Let S be a pattern space that consists of A mutually exclusive sets $S = C_1 \cup \ldots \cup C_A$, each of C_i, $i \in \Lambda = 1, \ldots, A$ representing a set of specified patterns called a class. Let x be an input pattern that should be assigned to one of the A existing classes. e denotes the classifier, and $e(x) = m^i(x) | \forall i (1 \leq i \leq A)$ indicates that the classifier e assigns the input x to each class i with a measurement value $m^i(x)$. This definition is used for all neural classifiers of the system.

We have adopted one ten-numerical-class classifier (e_{0-9}) and four MLPs specialized in the lexicon of the four metaclasses of digits (e.g., the $e_{1,2}$ classifier works

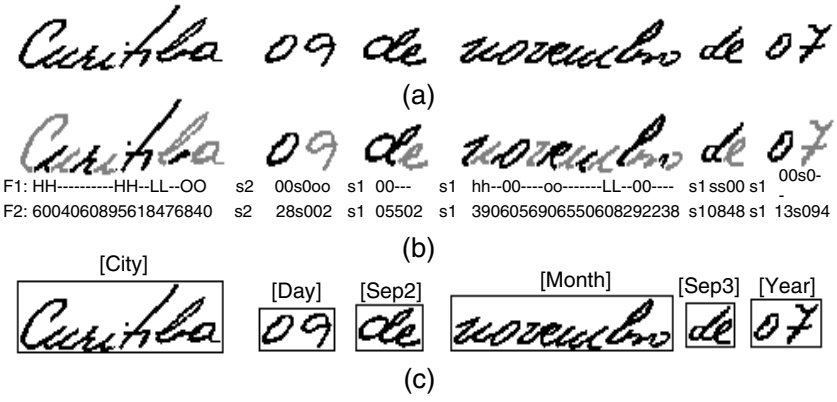

Fig. 10. a Original image. **b** Image segmented into a sequence of graphemes where each one is represented by two feature sets (F_1 and F_2). **c** Result of the segmentation into subfields

Table 4. Description of the MLPs used in the system

Classifier	Hidden units	Classes
$e_{0,1,2,3}$	70	0,1,2 and 3 ($C_{0,1,2,3}$)
e_{0-9}	80	0-9 (C_{0-9})
$e_{0,1,2,9}$	70	0,1,2 and 9 ($C_{0,1,2,9}$)
$e_{0,9}$	70	0 and 9 ($C_{0,9}$)
$e_{1,2}$	70	1 and 2 ($C_{1,2}$)

with the lexicon of the class of digits $C_{1,2}$ and so on) (Fig. 4). In this case, the digit string recognition module will determine which of these classifiers will be used according to the subfield (day or year) and its number of digits obtained by the segmentation module. This scheme aims at reducing the lexicon size on digit string recognition to increase the recognition results (Sect. 5). Table 4 details these classifiers, which have one hidden layer where the units of input and output are fully connected with the units of the hidden layer. The number of hidden units used, which was determined empirically, is also described in this table. The learning rate and the momentum term were set at high values in the beginning to make the weights quickly fit the long ravines in the weight space, and these parameters were then reduced several times according to the number of iterations to enable the weights to fit the sharp curvatures.

The rule that defines how the classifier assigns an input pattern x to a class i is known as the decision rule. In this work, the decision rule applied to all classifiers is defined as

$$e(x) = \max_{i \in \Lambda} m^i(x). \quad (1)$$

4 Description of the modules

In the following sections, we describe in greater depth the modules comprising our date recognition system.

4.1 Segmentation into subfields

As stated earlier, the goal of this module is to provide the segmentation of a date into subfields. Therefore, given an HMM-based segmentation approach, each date image is transformed as a whole into a sequence of observations by the successive application of preprocessing, segmentation, and feature extraction. Preprocessing consists of correcting the average character slant. The segmentation algorithm uses the upper contour minima and some heuristics to split the date image into a sequence of graphemes, each of which consists of a correctly segmented, an undersegmented, or an oversegmented character. A detailed description of these stages is given in [19]. Then, two feature sets are extracted from the sequence of graphemes and combined using a multicodebook-based HMM.

Thus, the segmentation of a date into subfields is delivered by the HMMs as a byproduct of the recognition process. This is done by backtracking the best path produced by the Viterbi algorithm [11]. In this case, the system takes into account the result of the segmentation of the best date model (among the eight possibilities presented in Table 3) that represents a date image. Figure 10 illustrates the entire process.

In the following sections, we describe the two feature sets and the mechanism we have used to train the elementary models used at this stage.

4.1.1 Feature extraction. As we pointed out earlier, the HMMs are fed by two feature sets. The first set targets the recognition of cursive handwriting; it is based on global features such as loops, ascenders, and descenders. The second one targets the discrimination of both letters and digits and is based on concavity measurements. Both feature sets are combined with the space primitives as described below.

The ascenders, descenders, and loops are detected through the local maxima from the upper contour, the local minima from the lower contour, and the closed contour, respectively. These three primitives can be classified as big or small primitives depending on their positions with respect to the upper and lower baselines detected in a date image. The combination of these primitives plus a primitive that determines whether or not a grapheme contains ascender, descender, or loop produces a 20-symbol alphabet. In such a case, each symbol has been evaluated in the training set to prevent it from either occurring or from having many samples.

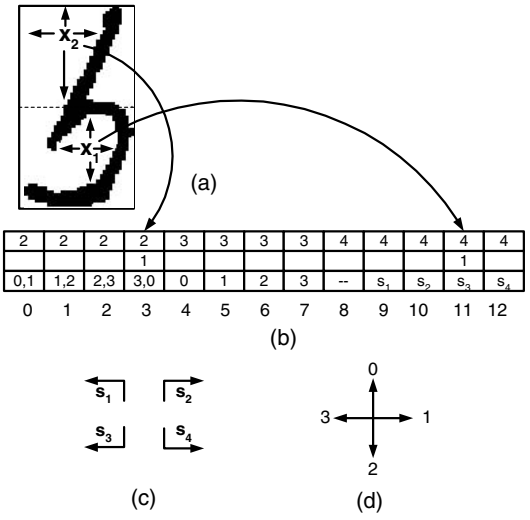

Fig. 11. Concavity measurements: **a** concavities, **b** feature vector, **c** auxiliary directions, and **d** 4-Freeman directions

The basic idea of concavity measurements [9] is the following: for each white pixel in the grapheme, we search in 4-Freeman directions (Fig. 11d) to find out which directions reach black pixels and which directions do not reach any black pixels. When black pixels are reached in all directions (e.g., point x_1 in Fig. 11a), we branch out in four auxiliary directions (s_1 to s_4 in Fig. 11c) to confirm if the current white pixel is really inside a closed contour. Those pixels that reach just one black pixel are discarded.

Then we increment the position in the feature vector according to the results returned by the search (Figs. 11a and b). In Fig. 11b, we represent the feature vector where each component has two labels. The upper label indicates the number of directions that reached black pixels during the search, while the lower label indicates the directions where black pixels were not reached. For example, the pixel x_2 (Fig. 11a) reaches the black pixel in directions 1 and 2. Therefore, position 3 of the feature vector is incremented. For the pixel x_1, position 11 is incremented because it reaches the black pixel in all four directions. However, using the auxiliary direction s_3 we confirm that it is not inside a closed contour. When the pixel is inside a closed contour, the position incremented is the eighth position.

Since we are always dividing each grapheme into two equal zones (horizontal), as shown in Fig. 11a, we consider two feature vectors of 13 components each. Therefore, in this figure, the pixel x_2 will update the first vector while the pixel x_1 will update the second one. Finally, the overall concavity feature vector is composed of 26 (13×2) components normalized between 0 and 1.

The spaces between two connected components have been extracted from a date image and are then combined with the global and concavity features. The space values and the concavity vectors are clustered into symbols by a vector quantization algorithm [10]. In the former case, the codebook size we adopted was eight, and in the latter case it was 100. These numbers were chosen after several tests carried out on the validation set.

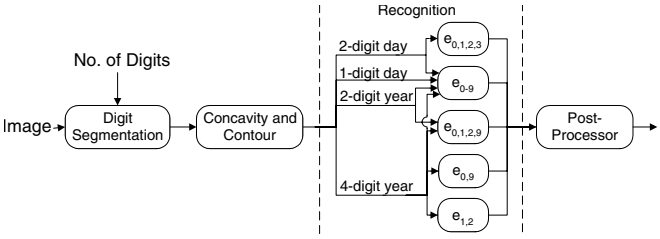

Fig. 12. Block diagram of the digit string recognition system

Figure 10b shows an example of a segmented date image represented by these two feature sets. While F_1 corresponds to a mixture of global and space features, F_2 is related to a mixture of concavity and space features.

4.1.2 Training mechanism. The mechanism we used to train the elementary models employed in subfield segmentation and word recognition (Table 1) consists of two training steps. In the first step, the city model is trained using images of two different city names extracted from the date database (Fig. 1). The number of states was determined after several tests considering different numbers of states. We used about 980 and 370 images for training and validation, respectively. In the second step, besides the date database, we also consider the legal amount database, which is composed of isolated words, to increase the training and validation sets. This is possible once our system takes into consideration one model for each letter. The use of metaclasses of digits also allows the sharing of data during training and consequently increases the training and validation sets. In this case, the parameters of the city model are initialized based on the parameters obtained in the first step of training. Then the other elementary models presented in the date and word images are trained systematically by concatenating them. We used about 1200 and 400 date images as well as 8300 and 1900 word images for training and validation, respectively.

4.2 Digit string recognition

After segmenting a date image into its constituent parts, the subimages of the day and year are used as input to the digit string recognizer. Moreover, the number of digits supplied by the HMMs is used as a priori information on digit string recognition to determine which classifiers will be employed depending on the subfield (day or year). As discussed in Sect. 3.3, we have defined five neural classifiers. e_{0-9} copes with the ten numerical classes, and the other classifiers – $e_{0,1,2,3}$, $e_{0,1,2,9}$, $e_{0,9}$, and $e_{1,2}$ – are specialized in the lexicon of the metaclasses of digits $C_{0,1,2,3}$, $C_{0,1,2,9}$, $C_{0,9}$, and $C_{1,2}$, respectively. This strategy aims at reducing the lexicon size on digit string recognition to improve the recognition results. Figure 12 shows the block diagram of the digit string recognition system to be described below.

Table 5. Description of the classifiers

Classifier	Classes	TR	VL	TS	RR VL	RR TS
$e_{0,1,2,3}$	0,1,2 and 3 ($C_{0,1,2,3}$)	8,300	1,250	2,500	99.7%	99.4%
e_{0-9}	0-9 (C_{0-9})	14,000	3,000	5,000	99.0%	98.9%
$e_{0,1,2,9}$	0,1,2 and 9 ($C_{0,1,2,9}$)	8,300	1,250	2,500	99.7%	99.4%
$e_{0,9}$	0 and 9 ($C_{0,9}$)	3,400	500	1,000	99.9%	99.8%
$e_{1,2}$	1 and 2 ($C_{1,2}$)	4,400	700	1,400	99.8%	99.5%

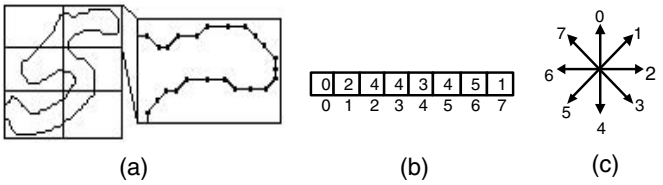

Fig. 13a–c. Contour measurement: **a** contour image of the upper right corner zone, **b** feature vector, and **c** 8-Freeman directions

The digit segmentation module shown in Fig. 12 is based on the relationship of three complementary sets of structural features, namely, contour, profile, and skeletal points. The segmentation hypotheses are generated through a segmentation graph, which is decomposed into linear subgraphs representing the segmentation hypotheses. More details on such an algorithm can be found in [21].

For each segmentation hypothesis a mixture of concavity/contour is extracted. We have used six concavity feature vectors of 13 components each since we are dividing the image into six zones. In this way, the overall concavity feature vector is composed of 78 (13×6) components normalized between 0 and 1. The concavity features employed in digit string recognition are very similar to the concavity features described in Sect. 4.1.1; they differ in the size of vector and the zoning used.

The contour information is extracted from a histogram of contour directions. For each zone, the contour line segments between neighboring pixels are grouped into 8-Freeman directions (Fig. 13c). The number of line segments of each orientation is counted (Fig. 13b). Therefore, the contour feature vector is composed of 48 (8×6) components normalized between 0 and 1. Finally, the last part of the feature vector is related to the character surface. We simply count the number of black pixels in each zone and normalize this value between 0 and 1. Thus, the final feature vector, which feeds our classifiers, has 132 ($78 + 48 + 6$) components.

To train those classifiers, we have used images of isolated digits extracted from the courtesy amount and date databases. Table 5 describes the databases used for training (TR), validation (VL), and testing (TS) as well as the recognition rates achieved on validation (RR VL) and test (RR TS) sets.

Since we are dealing with multiple hypotheses of segmentation and recognition, the generation of k best hypotheses of a string of digits is carried out by means of a modified Viterbi algorithm, which ensures the calculation of the k best paths of a segmentation–recognition graph [21]. Thus, the final probability for a hypothesis of segmentation–recognition is given by the product of the probabilities produced by the classifiers, as shown in Fig. 14. For simplicity, this figure presents just one hypothesis of segmentation. Afterwards, each hypothesis is submitted to the postprocessor module depicted in Fig. 12, which verifies whether it belongs to the lexicon of the day or year depending on the subfield.

4.3 Word recognition and verification

In this section, we describe the HMM-based word recognition and verification approach. While the recognizer is devoted to the word segmentation and recognition aspects, once it uses the same models employed in segmentation into subfields, the proposed verifier is specialized in the word recognition problem. This verifier deals with the loss in terms of recognition performance brought by the word recognition module and aims at improving the word recognition results and reliability of the system. The word models employed in word recognition and verification are discussed in Sect. 3.2.2.

The word recognizer receives as input the two sequences of observations related to the word image identified in the sentence that were extracted in the segmentation into subfield module. The word recognizer then computes the word probabilities for the 12 word models using the Forward procedure [11]. Then, only the two best hypotheses generated by the word recognizer are confirmed by the word verifier.

The word verifier takes a word image as input, which is transformed into a sequence of observations. This is done by segmenting the image into graphemes and then extracting two feature sets from the sequence of graphemes. The first set is based on global features, while the second one is a mixture of concavity and contour features. Both feature sets are combined with the segmentation primitives. Since the segmentation algorithm and the global features are the same described throughout this paper, here we explain the concavity and contour features, which differ in the size of concavity and contour vector, and the segmentation primitives.

Since we are dividing a grapheme into two equal zones (horizontal) and for the concavity features we are considering only those white pixels that reach three or more black pixels, we have two concavity vectors of 9 components each instead of 13 components as employed in date

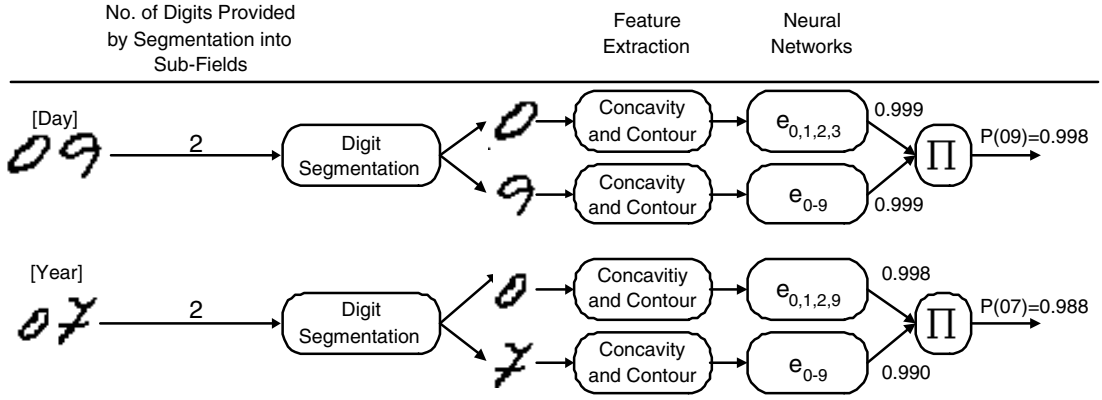

Fig. 14. Digit string recognition through an example

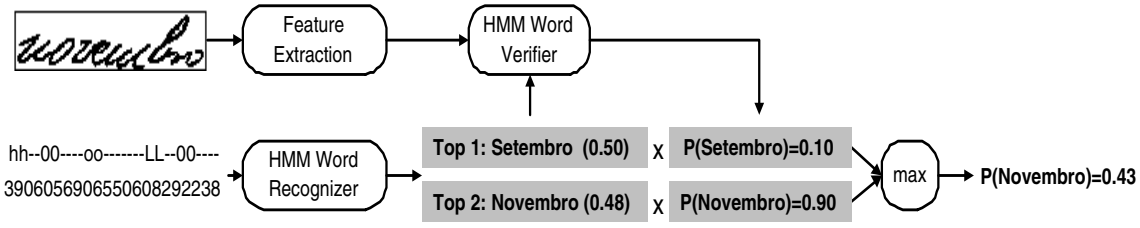

Fig. 15. Example of how the word verifier interacts with the word recognizer

segmentation. For each vector, we have introduced 8 more components related to the information about the contour image. They have been used to increase the discrimination between some pairs of letters (e.g., "L" (JULHO) and "N" (JUNHO)). In this manner, the final feature vector has 34 $(2 \times (9 + 8))$ components. The feature vectors are clustered into symbols by a vector quantization algorithm. We have used a codebook with a size of 100 chosen after carrying out several tests on the validation set.

The segmentation features have been used to reduce confusions such as "n" (junho) and "l" (julho) and "i" (maio) and "r" (marco) since they try to reflect the way the graphemes are linked together. For connected graphemes, we encode the nature of segmentation points in two ways depending on whether the point is closer to the upper or lower baselines. We have also defined a primitive to indicate no segmentation point between two graphemes.

Therefore, given an input word image, the output of the feature extraction process is a pair of symbolic descriptions of equal length, each consisting of an alternating sequence of grapheme shapes and associated segmentation point symbols. To train the elementary models employed in word verification, we have used about 9500 and 2300 word images for training and validationextracted from, respectively, the date and legal amount databases.

The objective of the word verifier is to rerank the output of the word recognizer. The word verifier computes the probabilities for two word models that correspond to the two best hypotheses (Top1 and Top2) generated by the word recognizer using the Forward procedure. We then multiply the probabilities produced by the word recognizer and verifier. In Fig. 15, we present an example of how the word verifier interacts with the word recognizer. We can see in this figure that the word recognizer generates the list of hypotheses that contain the correct one ("Novembro"), but it is not at the top of the list. On the other hand, the word verifier succeeds in reranking the correct hypothesis to the top of the list $(0.48 \times 0.90 > 0.50 \times 0.10)$. In such an example, we have used fictitious probabilities to better illustrate the problem. In Sect. 5, we will see the improvements produced by this scheme of verification.

4.4 Final decision

The final decision module accepts the recognition result when all three obligatory date subfields are correctly classified, otherwise it rejects it. The goal of rejection is to minimize the number of recognition errors for a given number of rejects. The scheme of rejection we have used is based on Fumera et al. [5].

This technique suggests the use of multiple reject thresholds for the different data classes $(T_0, \ldots, T_n)$ to obtain the optimal decision and reject regions. In such a case, we have considered one threshold for each class of digits and for each month word. To define such thresholds, we have developed an iterative algorithm that takes into account a decreasing function of the threshold variables $R(T_0, \ldots, T_n)$ and a fixed error rate T_{error}. We start from all threshold values equal to 1, i.e., the error rate equal to 0, since all images are rejected. Then at each step the algorithm decreases the value of one of the thresholds to increase the accuracy until the error rate exceeds T_{error}. The error rate is defined in Eq. 3.

Table 6. Segmentation results

Set	City	Day	Sep2	Month	Sep3	Year	No. of digits	
							Day	Year
VL	94.9%	94.1%	94.6%	97.9%	98.9%	100.0%	91.1%	100.0%
TS	95.7%	96.2%	95.5%	99.5%	100.0%	100.0%	92.2%	100.0%

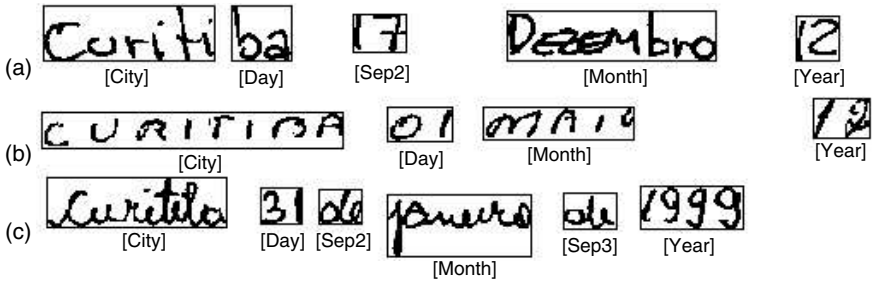

Fig. 16. a Example of missegmented date image. **b,c** Examples of well-segmented date images

Thereafter the rejection of an image is straightforward. We just compare the probability of its components (month and digits) with their corresponding thresholds. If any of the components has a probability less than its corresponding threshold, the entire string is rejected, otherwise it is accepted.

We will see in the next section that this strategy of rejection produces interesting error-reject trade-offs. We will present experimental results considering different values of T_{error}.

5 Experiments and analysis

This section is devoted to the experiments conducted on two databases to assess our approach. The first database contains about 2000 binary images of handwritten dates and aims at evaluating the performance of the system on the recognition of dates written on Brazilian bank checks. It was divided into three sets: 1200, 400, and 400 images for training, validation, and testing, respectively. This omniwriter database was collected at the campus of the Pontifical Catholic University of Paraná (PUCPR) in Curitiba, Brazil, where writers were students of the campus. Each date was written in a separate blank sheet of paper that was put under a form of a Brazilian bank check. In this way, it was not necessary to consider any extraction process of this field from the background of checks. There were no constraints on the writing style. All images were acquired in 300 dpi. The second database is the NIST SD19 (hsf_7 series), and it aims at validating the concept of metaclasses of digits on digit string recognition on a well-known database.

For all reported results, we used the following definitions of the recognition rate, error rate, rejection rate, and reliability rate. Let B be a test set with N_B string images. If the recognition system rejects N_{rej}, correctly classifies N_{rec}, and misclassifies the remaining N_{err}, then

$$\text{Recognition rate} = \frac{N_{rec}}{N_B} \times 100 \tag{2}$$

$$\text{Error rate} = \frac{N_{err}}{N_B} \times 100 \tag{3}$$

$$\text{Rejection rate} = \frac{N_{rej}}{N_B} \times 100 \tag{4}$$

$$\text{Reliability} = \frac{N_{rec}}{N_{rec} + N_{err}} \times 100. \tag{5}$$

Therefore, the recognition rate, error rate, and rejection rate sum up to 100%.

5.1 Experiments on date

5.1.1 Date segmentation results. The first step of our system is to find the best date model (among the eight possibilities presented in Table 3) that represents a date image. At this level, the system reached 93.6% and 95.5% on the validation and test sets, respectively. Table 6 details the segmentation rate of each date subfield on the validation (VL) and test (TS) sets as well as the result when the number of digits present in a string (day or year) is correctly estimated by the HMMs. The results shown in this table were evaluated automatically by the system. This was possible because we have a labeled database that also contains information about the position of each date subfield.

Figure 16a shows an example of when the date subfields are missegmented, and Figs. 16b and 16c demonstrate difficult cases of segmentation, where the spaces between subfields and within subfields are very similar. However, in such cases our approach succeeded in segmenting the date subfields correctly.

5.1.2 Date recognition results. Table 7 reports the improvements on date recognition using the word verifier on the validation (VL) and test (TS) sets (zero-rejection level). A date image is counted as correctly classified if the three obligatory subfields are correctly classified. In

Table 7. Performance of the system (−: without verification and √: with verification)

Set	Word verifier	Date	Month	1-digit day	2-digit day	2-digit year	4-digit year
VL	−	79.5%	89.6%	60.0%	93.2%	97.2%	–
	√	82.5%	93.1%	60.0%	93.2%	97.2%	–
TS	−	80.7%	89.5%	71.4%	92.6%	97.7%	100.0%
	√	82.5%	91.5%	71.4%	92.6%	97.7%	100.0%

addition, this table presents the results on digit string recognition and word recognition with verification.

Note in Table 7 that the verification brings an increase in the recognition rates on date by 3.0% and 1.8% on the validation and test sets, respectively. In this case, it is very difficult to compare our approach with other sentence recognition engines due to the special application of our work.

Comparison with other approaches is very delicate when we consider bank check recognition systems since different databases and formats are used, different word classes are involved, and different sizes of databases are considered. Furthermore, the literature indicates few studies in this area. For example, Suen et al. in [26] present a system for segmenting date images into subfields written on Canadian bank checks. In this application, each date image can appear in any one of two patterns: $MM\ S\ DD\ S\ 19\ YY$ and $DD\ S\ MM\ S\ 19\ YY$, where MM, S, DD, and YY refer to month, separator, day, and year subfields, respectively. The month subfield can be written in different data types (digits or words in English or French), while the day and year subfields can be written only in digits. The authors claim a segmentation rate of 83.19% on the test set of 310 French and 499 English checks from the CENPARMI_IRIS database. Xu et al. in [28] present an extension of the previous work. It focuses on the more difficult task of separating the day and month subfields. The result achieved was 89.90% using 1000 English date images from the CENPARMI_IRIS database. Considering our approach, we reached an interesting segmentation rate of 95.5% on the test set.

With respect to month word recognition, Xu et al. in [27] also report a modified product rule to combine two MLP classifiers and an HMM classifier. Their goal is to recognize month words in English and French. The recognition rate obtained was 85.36% using a test set of 2063 samples from a separate set of the CENPARMI_IRIS database. De Oliveira Jr. et al. in [3] consider the same classes of month words we have used, but they make use of a different database. Their best result on month word recognition was 87.2% by combing two MLP classifiers on the test set composed of 1200 isolated images of month words. Concerning our system, we achieved a recognition rate of 91.5% on month word recognition on the test set considering the word verifier. We believe that this result is quite promising since we must take into consideration the segmentation aspects. We note from Table 7 that by using the word verifier we were able to increase the recognition rates by 3.5% and 2.0% on the validation and test sets, respectively. We observed on the validation set that the presence of common substrings among some word classes can affect the performance on month word recognition such as:

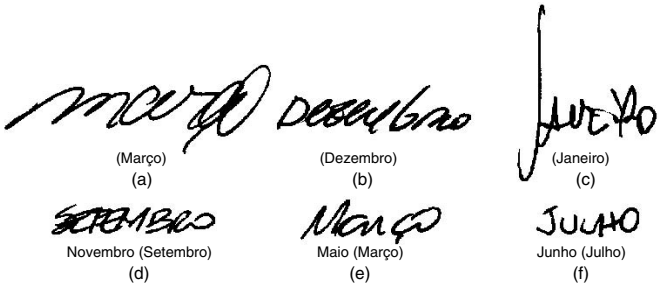

Fig. 17. a–c Examples of well-classified images. **d–f** Examples of misclassified images (the correct string is the one in parentheses)

- the termination in "eiro" for "Janeiro" and "Fevereiro";
- the termination in "embro" for "Setembro", "Novembro", and "Dezembro";
- almost all characters between "Junho" and "Julho" and between "Maio" and "Março".

Figures 17a, 17b, and 17c illustrate some examples of well-classified images. Figures 17d, 17e, and 17f show some recognition errors that correspond to undersegmentation problems due to the lack of local minima and confusion of the word classifiers.

In this application, the year segmentation is less complex than the day due to the low frequency of the "De" separator before the year and its location (i.e., the year is the last subfield present in the date field) (Table 6). This explains why the results shown in Table 7 on year recognition are higher for 2-digit strings than those achieved on day recognition for 2-digit strings. In this table, the low recognition rates for 1-digit days on the test and validation sets, and the difference of about 11% between them can be explained by the fact that we have few images for 1-digit days on the test (21 images) and validation (25 images) sets. In the validation set, for example, one misclassified image means 4% error.

By analyzing the errors on digit string recognition on the validation set, we noted that they could be classified into five classes: errors caused by recognition (Rec.), fragmentation (Frag.), digit segmentation (Digit Seg.), and date segmentation when our HMM-based approach does not correctly segment the day and year subfields (Seg.

Table 8. Error analysis on digit string recognition on the validation set

Subfield	Seg. into subfields	Rec.	Frag.	Digit seg.	No. of digits
Day	57.1%	17.2%	8.5%	–	17.2%
Year	–	58.3%	16.7%	25.0%	–

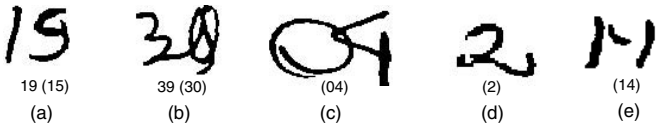

Fig. 18a–e. Examples of errors that come from **a,b** recognition, **c** digit segmentation, **d** number of digits, and **e** fragmentation (the correct label is the one in parentheses)

Table 9. Recognition rates (RR), rejection rates (RJ), and reliability rates (RL) for various error rates on date recognition

Error = 1.2%			Error = 2.0%		
RR(%)	RJ(%)	RL(%)	RR(%)	RJ(%)	RL(%)
62.1	36.6	98.1	65.3	32.6	97.0

into subfields) or it does not properly identify the number of digits present in a string (day or year) (No. of digits), e.g., in Fig. 18d two digits were identified instead of one. The confusions produced by fragmentation are found basically when the digit segmentation algorithm groups the fragmented part with the wrong neighbor. Usually, it fails for images that have neighbors (left and right) with similar distances to the fragmented part (Fig. 18e) and for images with poor quality. The digit segmentation errors can be caused either by undersegmentation (Fig. 18c), which is due to a lack of basic points in the neighborhood of the connection stroke, or wrong segmentation. These errors can be visualized in Table 8, while some examples are illustrated in Fig. 18.

Since bank check systems demand low error rates, two experiments were carried out on date recognition using the test set where the error rates were fixed at 1.2% and 2.0%, respectively. Table 9 presents recognition (RR), rejection (RJ), and reliability (RL) rates at these two error levels, while Fig. 19 shows the evolution of the recognition rate, error rate, and reliability as a function of the rejection rate. We can observe from this figure that the strategy of rejection that considers multiple reject thresholds produces interesting error-reject trade-offs.

5.2 Experiments on NIST SD19

To validate the concept of metaclasses of digits on digit string recognition, we performed two experiments using a subset of the validation set of the NIST SD19 database (hsf_7). In such experiments, we considered images of 2-digit strings related to the lexicon of 2-digit days. Table 10 summarizes the recognition rates (RR) achieved on

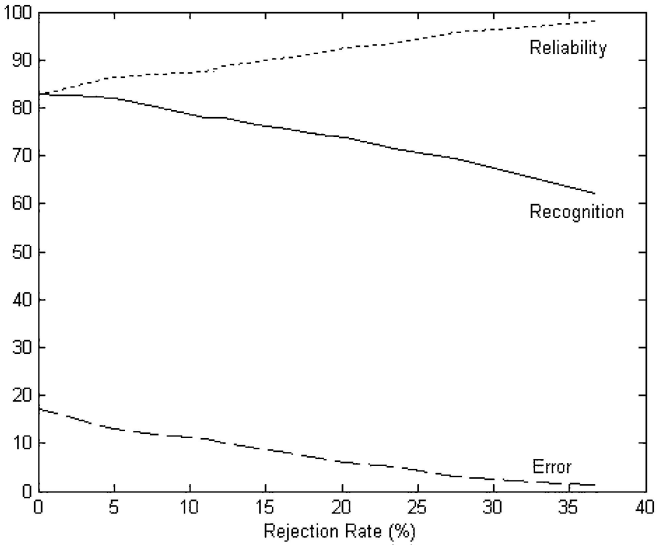

Fig. 19. Error rate vs. rejection rate for date recognition

Table 10. Recognition rates on NIST database for 2-digit strings

Experiment	No. of images	RR
Exp I	986	99.2%
Exp II	986	97.1%

these experiments. The former (Exp I) considers the concept of metaclasses of digits using the classifiers $e_{0,1,2,3}$ and e_{0-9}, while the latter (Exp II) makes use of the e_{0-9} classifier, i.e., without the concept of metaclasses. The use of this concept on digit string recognition seems to be a good strategy when the lexicon is known and limited since it enhanced the recognition results from 97.1% to 99.2%, as shown in Table 10. To train the classifiers $e_{0,1,2,3}$ and e_{0-9}, we used 78,000 and 195,000 images of isolated digits, respectively, from hsf_{0,1,2,3}.

6 Conclusion

In this paper, we presented an HMM-MLP hybrid system for segmenting and recognizing date images written on Brazilian bank checks. The system evolves by dealing with many sources of variability such as heterogeneous data types and styles, variations present in the date field, and difficult cases of segmentation that make the recognizer task particulare hard to do.

The proposed system takes an HMM-based strategy for separating the date into subfields since we have seen

that this is a difficult task without resorting to recognition. Thus it is not necessary to perform a priori segmentation, and premature errors can be avoided. In addition, by identifying each date subfield through the HMMs we can recognize the three obligatory ones using specialized classifiers according to their respective known data types. However, the main strength of such an approach lies in the modeling phase. We have built HMMs based on the concept of metaclasses of digits in order to reduce the lexicon size of the day and year and improve the precision of their segmentation. We have shown difficult cases of segmentation in which our strategy works well.

To deal with heterogeneous data types, we propose to use HMMs and MLPs to work with words and strings of digits, respectively. This is justified by the fact that HMMs have been successfully applied to handwritten word recognition and MLPs to digit recognition. Moreover, the literature has shown better results on digit recognition using MLPs [8,21] than HMMs [2]. Although such methods are very well known, this work presents an interesting strategy for lexicon size reduction that makes use of the output of the HMMs (the number of digits present in a string) and the concept of metaclasses of digits, which are beneficial to the digit string component. We have seen that the use of metaclasses of digits on digit string recognition for 2-digit strings achieved encouraging results on the NIST SD19 database. We have also shown that the proposed word verification scheme brought an increase in the overall recognition rate of the system. We consider that our system has achieved good performance taking into account its complexity. However, we believe that it can be improved by working on features.

Acknowledgements. This work was supported by Fundação Araucária, CENPARMI, and NSERC of Canada. The authors would like to thank Luiz S. Oliveira for sharing the digit string recognizer system and the anonymous reviewers for their helpful comments in improving the earlier draft of this paper.

References

1. Bishop CM (1995) Neural networks for pattern recognition. Oxford University Press, Oxford, UK
2. Britto Jr A, Sabourin R, Bortolozzi F, Suen CY (2001) A two-stage HMM-based system for recognizing handwritten numeral strings. In: Proceedings of the 6th ICDAR, Seattle, September 2001, pp 396–400
3. De Oliveira Jr JJ, de Carvalho JM, de A Freitas CO, Sabourin R (2002) Feature sets evaluation for handwritten word recognition. In: Proceedings of the 8th IWFHR, Niagara on the Lake, CA, August 2002, pp 446–451
4. Favata JT, Srihari SN, Govindaraju V (1998) Off-line handwritten sentence recognition. In: Proceedings of the 6th IWFHR, Taegon, South Korea, August 1998, pp 171–176
5. Fumera G, Roli F, Giacinto G (2000) Reject option with multiple thresholds. Patt Recog 12:2099–2101
6. Ghosh J (2002) Multiclassifier systems: back to the future. In: Proceedings of the 3rd international workshop on multiple classifier systems, Cagliari, Italy, June 2002, pp 1–15
7. Gorski N, Anisimov V, Augustin E, Baret O, Maximov S (2001) Industrial bank check processing: the A2iA checkreader. Int J Doc Anal Recog 3:196–206
8. Ha TM, Bunke H (1997) Off-line, handwritten numeral recognition by perturbation method. IEEE Trans Patt Anal Mach Intell 19(5):535–539
9. Heutte L, Moreau J, Plessis B, Plagmaud J, Lecourtier Y (1993) Handwritten numeral recognition based on multiple feature extractors. In: Proceedings of the 2nd ICDAR, Tsukuba, Japan, October 1993, pp 167–170
10. Huang XD, Ariki Y, Jack MA (1990) Hidden Markov models for speech recognition. Edinburgh University Press, Edinburgh, UK
11. Jelinek F (1997) Statistical methods for speech recognition. MIT Press, Cambridge, MA
12. Kaufmann G, Bunke H (2000) Automated reading of cheque amounts. Patt Anal Appl 3:132–141
13. Kim G, Govindaraju V (1998) Handwritten phrase recognition as applied to street name images. Patt Recog 31(1):41–51
14. Kumar S, Ghosh J, Crawford MM (2002) Hierarchical fusion of multiple classifiers for hyperspectral data analysis. Patt Anal Appl 5(2):210–220
15. Lippmann RP (1989) Pattern classification using neural networks. IEEE Commun Mag 27(11):47–64
16. Marti U, Bunke H (2001) Text line segmentation and word recognition in a system for general writer independent handwriting recognition. In: Proceedings of the 6th ICDAR, Seattle, September 2001, pp 159–163
17. Marti U, Bunke H (2001) Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. Int J Patt Recog Artif Intell 15(1):65–90
18. Mohamed MA, Gader P (1996) Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques. IEEE Trans Patt Anal Mach Intell 18(5):548–554
19. Morita M, El Yacoubi A, Sabourin R, Bortolozzi F, Suen CY (2001) Handwritten month word recognition on Brazilian bank cheques. In: Proceedings of the 6th ICDAR, Seattle, September 2001, pp 972–976
20. Morita M, Sabourin R, Bortolozzi F, Suen CY (2002) Segmentation and recognition of handwritten dates. In: Proceedings of the 8th IWFHR, Niagara on the Lake, CA, August 2002, pp 105–110
21. Oliveira LS, Sabourin R, Bortolozzi F, Suen CY (2002) Automatic recognition of handwritten numerical strings: a recognition and verification strategy. IEEE Trans Patt Anal Mach Intell 24(11):1438–1454
22. Park J, Govindaraju V (2002) Use of adaptive segmentation in handwritten phrase recognition. Patt Recog 35:245–252
23. Park J, Govindaraju V, Srihari SN (1999) Efficient word segmentation driven by unconstrained handwritten phrase recognition. In: Proceedings of the 5th ICDAR, Bangalore, India, September 1999, pp 605–608
24. Rabiner LR (1989) A tutorial on hidden Markov models and selected applications in speech recognition. IEEE 77(2):257–286
25. Srihari SN, Govindaraju V, Shekhawat A (1993) Interpretation of handwritten address in us mailstream. In:

Proceedings of the 2nd ICDAR, Tsukuba, Japan, October 1993, pp 291–294
26. Suen CY, Xu Q, Lam L (1999) Automatic recognition of handwritten data on cheques – fact or fiction? Patt Recog Lett 20(13):1287–1295
27. Xu Q, Kim JH, Lam L, Suen CY (2002) Recognition of handwritten month words on bank cheques. In: Proceedings of the 8th IWFHR, Niagara on the Lake, CA, August 2002, pp 111–116
28. Xu Q, Lam L, Suen CY (2001) A knowledge-based segmentation system for handwritten dates on bank cheques. In: Proceedings of the 6th ICDAR, Seattle, September 2001, pp 384–388
29. El Yacoubi M, Gilloux M, Bertille JM (2002) A statistical approach for phrase location and recognition within a text line: an application to street name recognition. IEEE Trans Patt Anal Mach Intell 24(2):172–188

Marisa Morita received her B.S. in computer science from Pontifícia Universidade Católica do Paraná, Curitiba, PR, Brazil and her M.Sc. in electrical engineering and industrial informatics from the Centro Federal de Educação Tecnológica do Paraná (CEFET-PR), Curitiba, PR, Brazil in 1994 and 1998, respectively. From 1994 to 1998 she was a systems analyst at HSBC Bank and Paradigm Systems, Curitiba, PR, Brazil, where she worked on financial systems. Currently she is a Ph.D. candidate at École de Technologie Supérieure, Université du Québec, Montreal, Canada and visiting scientist at the Centre for Pattern Recognition and Machine Intelligence (CENPARMI). Her current interests include pattern recognition, unsupervised learning, HMMs, and image analysis.

Robert Sabourin received his B.ing., M.Sc.A., and Ph.D. in electrical engineering from the École Polytechnique de Montréal in 1977, 1980, and 1991, respectively. In 1977, he joined the physics department of the Université de Montréal where he was responsible for the design and development of scientific instrumentation for the Observatoire du Mont Mégantic. In 1983, he joined the staff of the École de Technologie Supérieure, Université du Québec, Montréal, P.Q., Canada, where he is currently a professeur titulaire in the Département de Génie de la Production Automatisée. In 1995, he also joined the Computer Science Department of the Pontifícia Universidade Católica do Paraná (PUCPR, Curitiba, Brazil), where he was coresponsible since 1998 for the implementation of a Ph.D. program in applied informatics. Since 1996 he has been a senior member of the Centre for Pattern Recognition and Machine Intelligence (CENPARMI). His research interests are in the areas of handwriting recognition and signature verification for banking and postal applications.

Flávio Bortolozzi received his B.S. in mathematics in 1977 from Pontifícia Universidade Católica do Paraná (PUC-PR), Brazil, his B.S. in civil engineering in 1980 from PUC-PR, and his Ph.D. in 1990 in systems engineering (computer vision) from the Université de Technologie de Compiègne, France, where his work was concerned with trinocular vision. From 1994 to 1999 he was chair of the Department of Informatics and dean of the College of Exact Sciences and Technology at PUC-PR. Currently he is a full professor in the Computer Science Department and the prorector for research at PUC-PR. His research interests include computer vision, handwriting recognition, document image analysis, educational multimedia, and hypermedia.

Ching Y. Suen received his M.Sc.(Eng.) from the University of Hong Kong and Ph.D. from the University of British Columbia, Canada. In 1972, he joined the Department of Computer Science of Concordia University where he became professor in 1979 and served as chair from 1980 to 1984 and as associate dean for research in the Faculty of Engineering and Computer Science from 1993 to 1997. He has advised more than 30 visiting scientists and postdoctoral fellows and about 50 doctoral and master's students. Currently he holds the distinguished Concordia Research Chair of Artificial Intelligence and Pattern Recognition and is director of CENPARMI, the Centre for PR & MI. Professor Suen is the author/editor of 11 books and more than 300 papers on subjects ranging from computer vision and handwriting recognition to expert systems and computational linguistics. He is the founder and editor-in-chief of a journal and associate editor of several journals related to pattern recognition. A Fellow of the IEEE, IAPR, and Academy of Sciences of the Royal Society of Canada, he has served several professional societies as president, vice president, or governor. He is also the founder and chair of several conference series including ICDAR, IWFHR, and VI. He has served as general chair of numerous international conferences including the International Conference on Document Analysis and Recognition held in Montreal in August 1995 and the International Conference on Pattern Recognition held in Quebec City in August 2002. Dr. Suen is the recipient of numerous awards, including the ITAC/NSERC Award (Information Technology Association of Canada and the Natural Sciences and Engineering Research Council of Canada) in 1992 and the Concordia "Research Fellow" award in 1998.