A Recognition and Verification Strategy for Handwritten Word Recognition

M. Morita[1,2], R. Sabourin[1−3], F. Bortolozzi[3] and C. Y. Suen[2]
[1]École de Technologie Supérieure, Montreal, Canada
[2]Centre for Pattern Recognition and Machine Intelligence, Montreal, Canada
[3]Pontifícia Universidade Católica do Paraná, Curitiba, Brazil
e-mail: marisa@cenparmi.concordia.ca

Abstract

In this paper a word recognition and verification scheme based on HMMs is presented. However, the main contribution of the current work lies in the validation of such a strategy. In order to perform this task, we carried out some experiments on word recognition using a legal amount database and then we compared the results reached with other study which makes use of the same database. The experiments demonstrate the efficiency of the strategy we developed for word recognition and verification.

1 Introduction

Machine simulation of human functions has been a very challenging research field since the advent of digital computers. In some areas which require a certain amount of intelligence, such as chess playing, tremendous improvements are achieved. On the other hand, humans still outperform even the most powerful computers in relatively routine functions such as vision. Machine simulation of human reading is one of these areas. It has been the subject of intensive research especially in unconstrained handwriting recognition. The interest devoted to this field is not explained only by the exciting challenges involved, but also the huge benefits that a system, designed in the context of a commercial application, could bring.

The literature contains many studies on the recognition of isolated units of writing such as characters, words or strings of digits, which are an important subtask of many applications such as reading texts from pages [4], postal addresses [8], and processing of dates [5], courtesy [7] and legal [2] amounts on cheques.

This paper addresses the off-line recognition of isolated handwritten words of legal amounts on Brazilian bank cheques. Although the lexicon size of legal amount words is limited (40 words), there are some classes such as "Sessenta" (Sixty), "Setenta" (Seventy), "Oitenta" (Eighty), and "Noventa" (Ninety) that contain a common sub-string (i.e., "enta") and can affect the performance of the recognizer. The problem is made more difficult once different styles of handwriting (uppercase, lowercase, and mixed) are considered. Figure 1 shows some samples of handwritten words of legal amounts on Brazilian bank cheques.

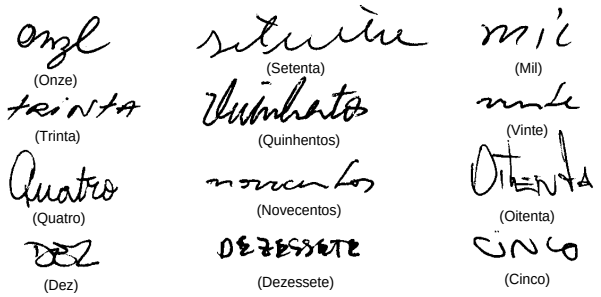

Figure 1. Samples of handwritten words of legal amounts on Brazilian bank cheques

In order to face the foregoing problem, we propose to use a word recognition and verification scheme which we have originally developed for recognizing Brazilian month words of dates on cheques [5]. However, the main contribution of the current work lies in the validation of such an approach since it is very difficult to compare our previous work with others due to its special application, i.e., date recognition on Brazilian bank cheques. Besides, comparison in the same context with other approaches is very delicate when different databases and formats are used, different word classes are involved, and different sizes of databases are considered. In order to assess the recognition and verification scheme, we carried out some experiments on word recognition using a legal amount database and then we compare the results achieved with an other study which makes use of the same database. The proposed strategy is based on the Hidden Markov Models (HMMs), which have been proven to be one of the most powerful tools for modeling speech and later on a wide variety of other real-world signals. Experi-

ments show an increase in the average recognition rate from 70.6% to 88.1%.

2 Description of the Word Recognition and Verification

Given a discrete HMM-based approach, each word 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 segments (graphemes), each of which consists of a correctly segmented, an under-segmented, or an over-segmented character. A detailed description of the preprocessing and segmentation stages is given in [6]. Then, different feature sets are extracted from the sequence of graphemes to feed the word recognizer and verifier. In the following subsections we describe the feature sets, the word models, and the interaction between the word recognizer and verifier.

2.1 Feature Extraction

2.1.1 Word Recognizer

The word recognizer makes use of two feature sets. The first set is based on global features such as loops, ascenders, and descenders, while the second one is based on concavity measurements. Both feature sets are combined with space primitives.

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 base lines detected in a word image. The combination of these primitives plus a primitive that determines whether a grapheme does not contain ascender, descender, and loop produces a 20-symbol alphabet. In such a case, each symbol has been evaluated during the training stage.

The basic idea of concavity measurements [3] is the following: for each white pixel in the grapheme, we search in 4-Freeman directions (Figure 2(d)), to find out which directions reach black pixels, as well as which directions do not reach any black pixels. When black pixels are reached in all directions (e.g. point x_1 in Figure 2(a)), we branch out in four auxiliary directions (s_1 to s_4 in Figure 2(c)) in order to confirm if the current white pixel is really inside a closed contour. Those pixels that reach just one black pixel are discarded.

Thereafter, we increment the position in the feature vector according to the results returned by the search (Figures 2(a) and (b)). In Figure 2(b) we represent the feature vector where each component has two labels. The superior label means the number of directions which reached black pixels during the search, while the inferior label means the directions where black pixels were not reached. For example, the pixel x_2 (Figure 2(a)) reaches the black pixel in directions 1 and 2. Therefore, the position 3 of the feature vector is incremented. For the pixel x_1, the 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 8^{th}.

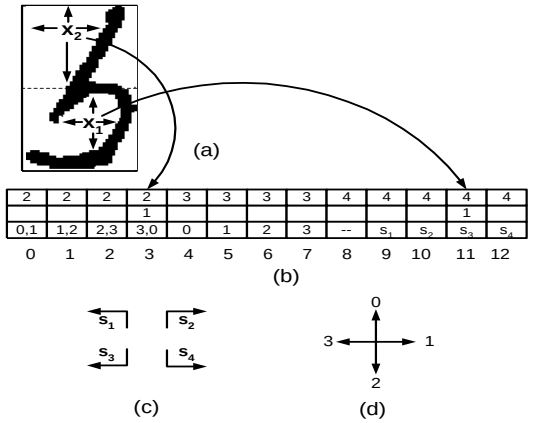

Figure 2. Concavity measurements: (a) Concavities, (b) Feature vector, (c) Auxiliary directions, and (d) 4-Freeman directions

Since we are dividing each grapheme into two zones, we consider two feature vectors of 13 components each. Therefore, in the example presented above, 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 (13×2) 26 components normalized between 0 and 1.

The spaces between two connected components have been extracted from a word image and then they are combined with the global and concavity features. The space values and the concavity vectors are clustered into symbols by a vector quantization algorithm. In the former case, the codebook size we have adopted was 8 and in the latter case it was 100. These numbers were chosen after several tests on the validation set.

2.1.2 Word Verifier

The word verifier is also fed by two feature sets. The first set is based on global features and the second one is a mixture of concavity and contour features. Both feature sets are combined with segmentation primitives. The global and concavity features are basically the same as we have used in word recognition. However, here the concavity features differ in the size of concavity vector.

Since we are dividing a grapheme into two zones, we have two concavity feature vectors of 9 components each. For each vector, we have introduced 8 more components related to the information about the contour image in order to increase the discrimination between some pairs of letters (e.g. "s" ("sete") and "o" ("oito")). 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. The number of line segments of each orientation is counted. In this manner, the final feature vector has $(2 \times (9+8))$ 34 components. The feature vectors are clustered into symbols by a vector quantization algorithm. We have used a codebook with the size of 100 chosen after carrying out several tests on the validation set.

The segmentation features have been used to reduce confusions such as "u" (duzentos) and "r" (trezentos) since they try to reflect the way that the graphemes are linked together. For connected graphemes, we encode the nature of segmentation points in two ways depending on whether its vertical position is closer to the upper or lower base lines. We have also defined a primitive to indicate no segmentation point between two graphemes.

2.2 Markoving Modeling of Words

Basically, the word models are formed by the concatenation of appropriate elementary HMMs, which are built at letter and space levels.

The topology of space model shown in Figure 3(a) consists of 2 states linked by two transitions that encode a space (transition t_{01}) or no space (transition $t_{01} = \Phi$).

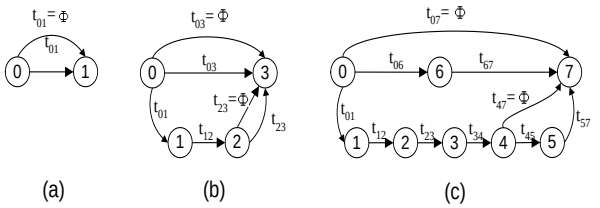

Figure 3. Topologies of (a) space, (b) and (c) letter models

Two topologies of letter models were chosen based on the output of our grapheme-based segmentation algorithm which may produce a correct segmentation of a letter, a letter under-segmentation or a letter over-segmentation into two, three, or four graphemes depending on each letter. In order 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 3(b) and 3(c) show examples of both topologies.

The model in Figure 3(b) is employed in word recognition. In such a topology, the transition t_{03} either (a) models under-segmentation 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 Figure 3(c) is used in word verification and it is based on the previous one, but in this case the model has transitions (t_{12}, t_{34}, t_{57} and t_{67} of Figure 3(c)) that encode the nature of segmentation points. Considering uppercase and lowercase letters, we need 42 models since the legal amount alphabet is reduced to 21 letter classes and we are not considering the unused ones. Thus, regarding the two topologies, we have 84 HMMs which are trained using the Baum-Welch algorithm with the Cross-Validation procedure.

Since no information on recognition is available on the writing style (uppercase, lowercase), the word model shown in Figure 4(a) consists of two letter HMMs in parallel and four 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 frequency of occurrence 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. This architecture handles the problem related to the mixed handwritten words detecting implicitly the writing style during recognition using the Backtracking of the Viterbi algorithm.

Figure 4(a) shows the architecture of the word models adopted on word recognition, while Figure 4(b) illustrates the architecture used for word verification. We can observe that the architecture shown in Figure 4(b) diverges only in two aspects: it does not consider the space models and its character HMMs contain transitions that encode the nature of segmentation points.

2.3 How the Word Verifier Interacts with the Word Recognizer

As pointed out earlier, the word recognizer receives as input two sequences of observations extracted from the word image. Then, the word recognizer computes the word probabilities for the word models using the Forward proce-

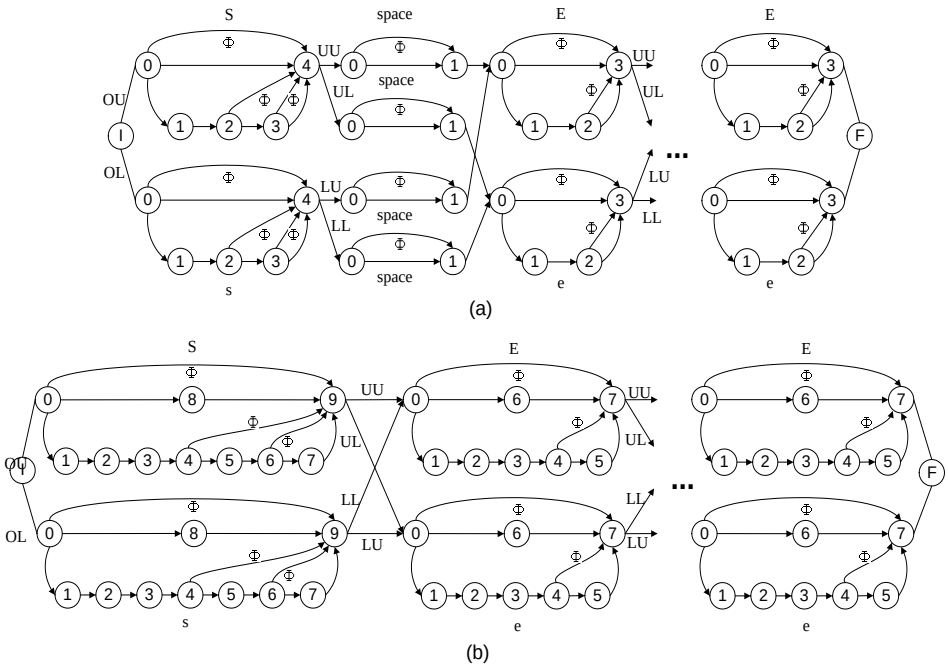

Figure 4. Word models of class "Sete" (Seven): (a) Recognition and (b) Verification

dure. Then, only the two best hypotheses generated by the word recognizer are confirmed by the word verifier. The objective of the word verifier is to re-rank the output of the word recognizer in order to improve the recognition rate and reliability of the system. This verifier deals with the loss in terms of recognition performance brought by the word recognizer.

Thus, 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. Then, we multiply the probabilities produced by the word recognizer and verifier. In Figure 5, 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 which contain the correct one ("Oitenta"), but it is not in the top of the list. On the other hand, the word verifier succeeds in re-ranking 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 in order to better illustrate the problem. In Section 3 we will see the improvements produced by this scheme of verification.

3 Experimental Results

This section is devoted to the experiments conducted on the legal amount database which contains 11,000 isolated images of handwritten words. It was divided into three sets: 6,600, 2,200, and 2,200 images for training, validation, and testing respectively.

Table 1 reports the two best recognition rates (Top1 and Top2) on the test set using our word recognizer without considering the word verifier. In this case, the objective is to compare the performance of our recognizer with the one developed by Freitas et al in [1]. This was possible since we have used the same database. Their work considers one global Markov model for each class of words and makes use of global, concavity, and convexity features. However, in this case modeling characters is better than modeling words due to the small number of images for training some classes of words which do not have a uniform distribution. Thus, by considering character models we can increase the training set and improve the performance on word recognition. We can observe from Table 1 the improvements on word recognition using our approach. The recognition rates for Top1 and Top2 were increased by 15.2% and 9.9% respectively.

Table 1. The two best recognition rates on word recognition on the test set

Word Recognizer	Top1	Top2
Current	85.8%	92.3%
[1]	70.6%	82.4%

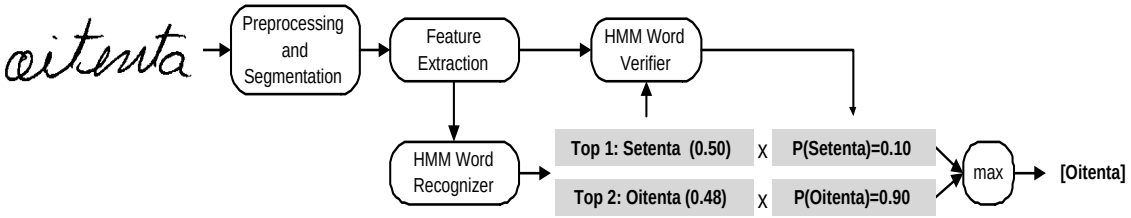

Figure 5. Example of how the word verifier interacts with the word recognizer

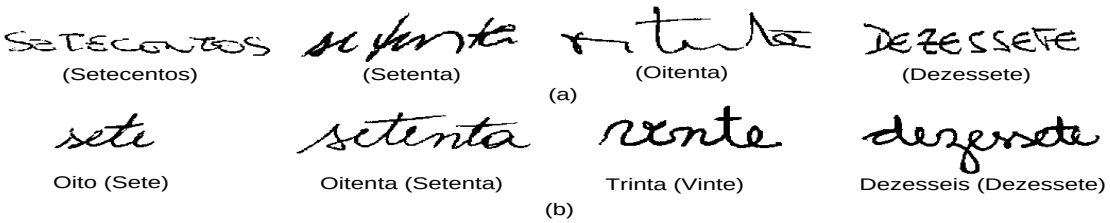

Figure 6. (a) Examples of well-classified images and (b) Examples of misclassified images (the correct string is the one in parentheses)

In the top of it, the word verifier brings an improvement of the recognition rate on word recognition on the test set from 85.8% to 88.1% (Top1). The results show the efficiency of the strategy we have developed for word recognition and verification.

Figure 6(a) illustrates examples of well-classified images. Figure 6(b) shows some recognition errors. As we expected, the main confusions of the classifiers are the presence of a common sub-string among some classes of words (e.g., "Setenta","Oitenta", and "Noventa") or similarities among them (e.g., "Seis", "Sete", and "Oito").

4 Conclusion

In this paper an HMM-based recognition and verification scheme for word recognition has been presented. It was originally developed for recognizing Brazilian month words of dates on cheques [5]. However, since it is very difficult to compare our previous work with others due to its special application, i.e., date recognition on Brazilian bank cheques, in this study we propose to assess such an approach. In order to perform this task, we carried out some experiments on word recognition using a legal amount database and then we compared the results reached by other researchers which makes use of the same database. The experiments demonstrate the efficiency of the strategy we developed for word recognition and verification since the average recognition rate was increased from 70.6% to 88.1%.

References

[1] C. Freitas, F. Bortolozzi, and R. Sabourin. Handwritten isolated word recognition: An approach based on mutual information for feature set validation. In *Proc. 6^{th} ICDAR*, pages 665–669, 2001.

[2] N. Gorski, V. Anisimov, E. Augustin, O. Baret, and S. Maximov. Industrial bank chech processing: the A2iA checkreader. *IJDAR*, 3:196–206, 2001.

[3] L. Heutte, J. Moreau, B. Plessis, J. Plagmaud, and Y. Lecourtier. Handwritten numeral recognition based on multiple feature extractors. In *Proc. 2^{nd} ICDAR*, pages 167–170, 1993.

[4] U. Marti and H. Bunke. Text line segmentation and word recognition in a system for general writer independent handwriting recognition. In *Proc. 6^{th} ICDAR*, pages 159–163, 2001.

[5] M. Morita, R. Sabourin, F. Bortolozzi, and C. Suen. Segmentation and recognition of handwritten dates. In *Proc. 8^{th} IWFHR*, pages 105–110, 2002.

[6] M. Morita, A. E. Yacoubi, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Handwritten month word recognition on Brazilian bank cheques. In *Proc. 6^{th} ICDAR*, pages 972–976, 2001.

[7] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Automatic recognition of handwritten numerical strings: A recognition and verification strategy. *IEEE PAMI*, 24(11):1438–1454, November 2002.

[8] A. E. Yacoubi, M. Gilloux, and J. Bertille. A statistical approach for phrase location an recognition within a text line: An application to street name recognition. *IEEE PAMI*, 24(2):172–188, February 2002.