2011 International Conference on Document Analysis and Recognition

An Optimized multi-stream decoding algorithm for handwritten word recognition

Yousri Kessentini, Thierry Paquet
Université de Rouen, Laboratoire LITIS EA 4108
Site du Madrillet, St Etienne du Rouvray, France
{*yousri.kessentini,thierry.paquet*}*@univ-rouen.fr*

Ahmed Guermazi
Polytechnique Annecy, Laboratoire LISTIC EA 3703
BP 80439 - 74944 Annecy-le-Vieux cedex - France
ahmed.guermazi@univ-savoie.fr

***Abstract*—This paper is focused on the optimization of the computational efficiency of a multi-stream word recognition system. The aim of this work is to optimize the multi-stream decoding step in order to reduce the recognition time and the complexity to allow combining a large number of streams. Two different multi-stream decoding strategies are compared based on two-level and HMM-recombination algorithms. Experiments carried out on public handwritten word databases show significant speed gains at decoding while keeping the same performances, in addition to new insights for combining a large number of streams.**

***Keywords*-Multi-stream HMM; Decoding; Two-level; Handwriting recognition;**

I. Introduction

Recently, there has been significant interest in the use of multi-stream hidden Markov models (HMMs) for handwriting recognition [3], [4], [5]. The multi-stream paradigm provides a particular way of combining individual feature streams using cooperative HMMs and presents multiple advantages: the topology of HMM can be adapted to each source of information, the combination can be adaptive: some sources of information can be weighted, or even rejected if they are not reliable. In addition, the multi-stream approach combines the likelihoods of multiple information sources at a coarser level than HMM state such as the character or syllable level, thus allowing asynchrony between the state sequences of the streams. Hence, we have demonstrated in [3] the superiority of the multi-stream strategy compared to the classical combination methods in case of handwritten word recognition. However, the gain in recognition performance was achieved at the expense of higher computational complexity due to the separate modeling of the different observation streams. In this previous work, decoding a multi-stream HMM was performed with HMM-recombination algorithm. This algorithm consists in building a product (composite) HMM where the number of states exponentially increases with the number of streams, resulting in a high computational complexity. This algorithm has a limitation for combining more than two streams when dealing with large lexicons [6]. While most of the research efforts on handwriting recognition have focused on improving recognition performance, less works have been devoted to optimizing the recognition time. Some optimization techniques presented in the literature are based on pruning mechanisms that attempt to reduce the lexicon size prior to the recognition [9], [13]. Other approaches attempt to reorganize the lexicon to exploit the presence of common prefixes in words that have similar spellings and avoid repeated computation using a lexicon tree structure [8], [15]. Other techniques, consist in reducing the search effort by introducing heuristics into the search algorithms such as A*, beam search, and two-level [14], [7].

In this paper, we present an optimized multi-stream decoding strategy based on two-level algorithm and we compare its computational complexity to the HMM-recombination algorithm. We demonstrate that the mutli-stream two-level decoding procedure reduces the computational complexity without any loss in the recognition performances. The rest of this paper is organized as follows: in section 2, we describe the multi-stream formalism and the decoding algorithms. Section 3 presents a comparison and a discussion about the computational complexity of the two decoding strategies. In section 4, we present a description of the overall recognition system. Finally, Section 5 provides experimental results followed by a conclusion and future works.

II. Multi-stream paradigm

The multi-stream paradigm [12], [10] provides an adaptive method to combine several individual feature streams using cooperative HMM. This problem can be formulated as follows: assume an observation sequence X composed of K input streams $X^k (k = 1, \ldots, K)$ representing the utterance to be recognized, and assume that the hypothesized model M for an utterance is composed of J sub-unit models $M_j (j = 1, \ldots, J)$ associated with the sub-unit level at which we want to perform the recombination of the input streams (e.g., characters). To process each stream independently of each other up to the defined sub-unit level, each sub-unit model M_j is composed of K models M_j^k (possibly with different topologies). Recombination of the K stream models M_j^k is forced at some temporal anchor states ($\otimes$ in Figure 1). The resulting statistical model is illustrated in Figure 1. Detailed discussion of the mathematical formalism is given in our previous work [3].

Decoding multi-stream models requires a more sophisticated procedure than the standard Viterbi search. Two

1520-5363/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDAR.2011.47

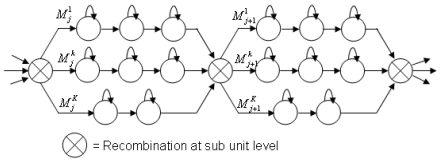

Figure 1. General form of K-stream model with anchor points between sub-units models

different algorithms have been proposed to solve the problem of decoding.

A. HMM-recombination algorithm

Here, a composite (or product) HMM is built by merging a n-tuple of states from the n stream HMMs [10]. The topology of this composite model is defined so as to represent all possible state paths given the initial HMM topologies. Figure 2 shows an example of a multi-stream HMM with 2 streams and its corresponding product HMM.

The product HMM parameters are determined as follows: The transition probabilities of the product HMM are derived from the transition probabilities of the 2 single stream HMMs assuming independence of the models between 2 recombination states. For example:

$$P(a-B|a-A) = P(a|a) \times P(B|A)$$

The conditional observation likelihoods of the composite HMM are obtained using a combination of the observation likelihoods of the single-stream components, for example:

$$P\left(X^1(t), X^2(t)\,|a-A\right) = P\left(X^1(t)\,|a\right)^{\alpha} P\left(X^2(t)\,|A\right)^{(1-\alpha)}$$

where $X^1(t)$ (similarly, $X^2(t)$) is the observation vector corresponding to stream 1 (similarly, stream 2) and α the reliability of stream 1 ($0 \leq \alpha \leq 1$). Decoding under such a model requires computing a single best path using the well known Viterbi decoding algorithm. We have demonstrated in [3] that this algorithm have a high complexity especially when dealing with a large number of streams. An alternative consists in using the two-level decoding algorithm.

B. Two level dynamic programming

This decoding strategy was initially proposed in [11] to optimize the Viterbi algorithm in the case of large vocabulary recognition problems. The decoding takes place in two steps. A first dynamic programming process is applied at the sub-unit level (phoneme or character models) and each sub-model is scored on arbitrary portions of the frame data. Secondly, sub-models are merged together in order to find the best overall score, during a second dynamic programming stage at word level.

In the case of multi-stream HMMs, the first level of this algorithm is slightly modified and each stream HMM is independently decoded for each possible portion of the frame

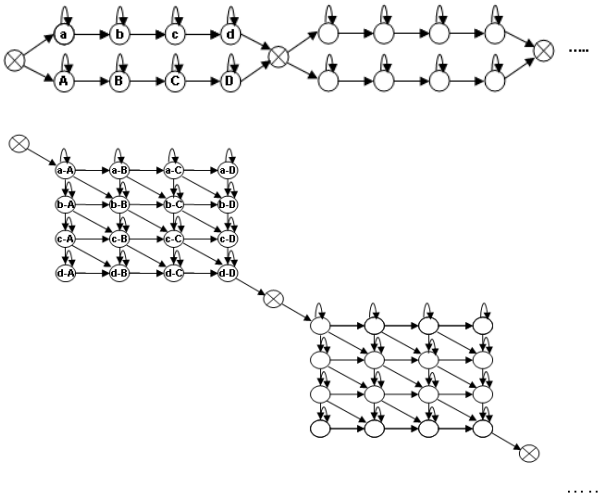

Figure 2. Example of a multi-stream HMM with 2 streams and its corresponding product (composite) HMM

data, the individual stream scores are then combined for further use at the second level. In the second level, only the character boundaries are decoded without the necessity of going through the HMM states. The details of the first level are presented as follows:

Level 1 : Two-level algorithm

1: For each stream S_k and each observation stream $X_t^k (k = 1, \ldots, K)$
2: For each character model $M_j^k (j = 1, \ldots, C)$
3: For each beginning frame $b, (b = 1, \ldots, T-1)$
4: For each end frame $e, (e = b+1, \ldots, T)$ compute the best score corresponding the best state sequence.
5: Store the obtained likelihood in $\Psi(M_j^k, b, e)$
6: Combine the stream likelihoods using a combination function f, $\Psi(M_j, b, e) = f(\Psi(M_j^1, b, e), \ldots, \Psi(M_j^K, b, e))$

The first decoding level can be viewed as a standard Viterbi algorithm which is used to decode the best character alignment for all possible positions of the character within the observtion stream. The last step of the first level consists in combining the different stream scores using a combination function as a weighted sum of log-likelihood. Once scores are computed for all character stream models, they can be reused to decode any lexicon, hence we avoid repeating character decoding for each hypothetized occurrence of the character at the some position while decoding each word of the lexicon. Given these scores, the second level of the computation pieces together the individual character scores to maximize the overall accumulated score over the entire word. This can be accomplished using dynamic programming as follows:

$$L(c_1,\ldots,c_l,e) = \max_{1\leq b\leq e}\left[L(c_1,\ldots,c_{l-1},b-1)\times\Psi(c_l,b,e)\right]$$

where $L(c_1,\ldots,c_l,e)$ is the score of the best path ending at frame e using the character sequence $c_1,c_2,\ldots,c_l$. The best path ending at frame e using exactly l character models is the one with maximum score over all possible beginning frames, b, of the concatenation of the best path ending at frame $b-1$ using exactly $l-1$ character models fold the best score of the character model c_l from frame b to frame e (calculated at the first level). Therefore, the second level consists as follows:

Level 2 : Two-level algorithm

1: For each word in the lexicon
2: For each character composing the word
3: For each end frame e $(2\leq e\leq T)$
4: Compute the score of the best concatenation of character until frame e: $L(c_1,c_2,\ldots,c_l,e) = \max_{1\leq b\leq e}\left[L(c_1,c_2,\ldots,c_{l-1},b-1)\times\Psi(c_l,b,e)\right]$

III. Computational complexity

We compare the computational complexity of the two decoding algorithms. In the case of the two-level decoding algorithm, the first level is independent of the lexicon size and its computational complexity is given by $\mathcal{O}(T^2N^2KC)$ where T is the length of a sequence of observations, N the number of states per character model, C the number of character models and K the number of streams. In the second level, the computation depends on the word in the lexicon, and the complexity is $\mathcal{O}(T^2LV)$, where L is the average length of the words in the lexicon and V is the number of words in the lexicon. The approximate computational complexity for the two-level decoding algorithm is then $\mathcal{O}(T^2N^2KC+T^2LV)$.

In the case of the HMM-recombination algorithm, the Viterbi algorithm is applied to decode the product HMM. The complexity of the Viterbi algorithm is $\mathcal{O}(T(LN)^2V)$. In the product HMM, the number of states increases to LN^K. Therefore, the complexity of the HMM-recombination algorithm becomes $\mathcal{O}(T(LN^K)^2V)=\mathcal{O}(TL^2N^{2K}V)$

Note that by just looking the complexity expressions of the two algorithms, it is hard to see which strategy is better. To get a feeling on the computational complexity of each decoding strategy, typical values of $T=100$, $L=5$, $N=4$, $K=4$, $V=100$, $C=26$ results in $\mathcal{O}(16384\times10^7)$ for HMM-recombination algorithm, and in $\mathcal{O}(2.164\times10^7)$ for two-level algorithm. Tables I, II present respectively, the variation of the the approximate computational complexity of the two algorithms with respect to the number of streams K and the lexicon size V. By analyzing the values in these tables, it is possible to have a better idea of the computational complexity of each decoding strategy. It is clear that the two-level algorithm is more advantageous when dealing with a large lexicon size and a great number of streams.

Table I
VARIATION OF THE OPERATIONS NUMBER ($\times10^4$) WITH RESPECT TO THE NUMBER OF STREAMS K, $T=100, L=5, N=4, V=100, C=26$

	Two-level	HMM-recombination
K=2	1332	6400
K=3	1748	102400
K=4	2164	1638400
K=5	2580	26214400

Table II
VARIATION OF THE OPERATIONS NUMBER ($\times10^4$) WITH RESPECT TO THE LEXICON SIZE V, $T=100, L=5, N=4, K=3, C=26$

	Two-level	HMM-recombination
V=100	1748	102400
V=1000	6248	1024000
V=2000	11248	2048000
V=5000	26248	5120000

Note that we can reduce the complexity of the two-level algorithm. In fact, T^2 can be reduced to $T(T-D)$, where D is an estimation of the duration of the character models, without loss of accuracy. The complexity of the first level can be also reduced assuming that the same HMM topology is used for all streams, in this case, all stream character HMMs can be simultaneously decoded and the complexity is reduced to T^2N^2C. Consequently, the overall complexity of the two-level algorithm can be reduced to $\mathcal{O}(T(T-D)(N^2C+LV))$.

IV. Recognition system description

The recognition system is based on a multi-stream HMM and proceeds on different stages as presented in figure 3. In the first step, pre-processing operations like slant correction, smoothing and normalization are applied to the word image in order to eliminate noise and to simplify the procedure of feature extraction [3]. The next step is to extract features from the input word image. Two types of features are considered: (i) contour based features and (ii) density based features. Contour based features are extracted from the lower and the upper contours, and density based features are computed on two different sliding windows with different width. Therefore, each feature type (contour or density feature) defines two feature streams representing the input word image. A complete description of features is given in [3]. Each stream model is then separately trained using embedded training where all character models are trained in parallel using Baum-Welch algorithm applied on

word examples. The last step is recognition during which the HMM models are simultaneously decoded according to the multi-stream formalism presented above.

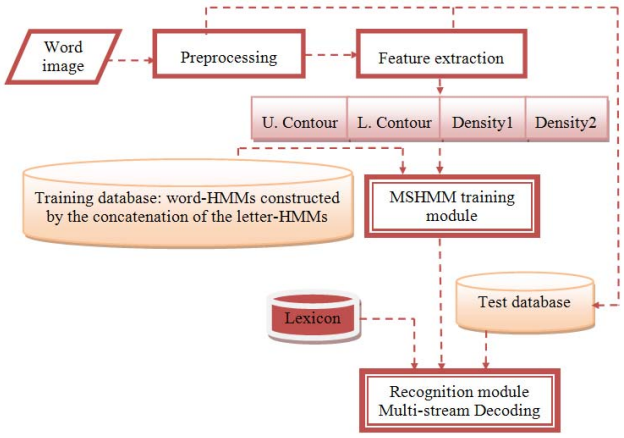

Figure 3. The multi-stream recognition system architecture

V. Experimental results

Experiments have been conducted on two publicly available databases: IFN/ENIT benchmark database of arabic words and RIMES database for latin words. The IFN/ENIT [2] contains a total of 32492 handwritten words (Arabic script) of 946 Tunisian town/village names written by 411 different writers. Some town/village names occur in the database with slightly different writing style according to the presence or absence of "shadda" for example. It follows that our lexicon is made of about 2,100 valid entries. Four different sets (a, b, c, d) are predefined in the database for training and one set (e) for testing. The RIMES database [1] is composed of isolated handwritten word snippets extracted from handwritten letters (Latin script). In our experiments, 36000 snippets of words are used to train the different HMM classifiers and 7464 words are used in the test. The dictionary is composed of 1612 words. During these experiments, we are interested in evaluating the two aspects: recognition accuracy and recognition time.

A. Recognition accuracy

Table III shows the experimental results of the performance of our recognition system using 4 different single streams (upper contour, lower contour and density with two windows of different width; Density1 and Density2 correspond to the windows of 8-pixel and 14-pixel widths, respectively) as a function of the size of the list of word hypothesis.

We present in table IV the best performances obtained by combining 2, 3 and 4 streams. Multi-stream two-level decoding algorithm is used for these experiments. As expected, we obtain exactly the same performances obtained by the HMM-recombination algorithm. The main advantage

Table III
Individual performances using single stream features

	IFN/ENIT		RIMES	
	Top 1	**Top 2**	**Top 1**	**Top 2**
Upper contour	70.50	78.60	54.10	66.40
Lower contour	63.50	73.10	38.93	51.57
Density1	65.10	73.00	53.23	65.83
Density2	68.70	78.10	52.23	65.40

of the two-level algorithm is to reduce the computational complexity which allows combining more than 2 streams when dealing with a large lexicon size, this was impossible using HMM-recombination algorithm due to the high computational cost.

Table IV
Multi-stream recognition performances

	IFN/ENIT		RIMES	
	Top 1	**Top 2**	**Top 1**	**Top 2**
2-stream	79.60	85.70	74.10	79.40
3-stream	79.83	86.93	74.32	79.67
4-stream	80.23	87.57	74.68	80.12

It must be stated that the main purpose of these experiments was to establish the contribution of additional streams to the recognition performances. Regarding this investigation, the experimental results show in both cases (RIMES and IFN/ENIT) interesting improvements by combining 2 streams. In fact, the best 2-stream recognition rate is 79.6% in Top 1 on IFN/ENIT and is obtained by combining upper contour and density2 features. The gain is 9.1% compared to the best single stream recognition rate. Similarly, the best 2-stream recognition rate on RIMES database is 74.10% by combining upper contour and density1 features, which corresponds to a gain of 20% compared to the best single stream recognition rate. On the contrary, when combining more than two streams, the experimental results show in both cases (IRONOFF and IFN/ENIT) the same moderate improvement of recognition performance. We think that this can be explained by the nature of the combined feature streams which do not provide sufficient complementary information.

B. Recognition time

We are interested in this section in comparing the recognition time of the two decoding strategies. The recognition time is defined as the time in seconds required to recognize one word and it is measured in CPU-seconds, which is the time the recognition process has exclusive use of the central processing unit of a computer with a multitasking operating system. In these experiments, the recognition time covers only the recognition process, excluding preprocessing and feature extraction steps. The machine used for these tests is Intel E7340, 2.4 GHz processor, 2GB

of RAM memory. Table V compares the word recognition time of the two decoding strategies with respect to the number of streams and the lexicon size. The performance of the HMM-recombination algorithm is completely flawed on large vocabulary tasks especially when dealing with large number of streams. In counterpart, the two-level algorithm presents a significant improvement in recognition time and appears less sensitive to the variation in the number of stream and the lexicon size. Despite these improvements, the recognition time of the two-level decoding algorithm is still high, and many investigations are still needed to improve its performance. This can be performed by using a lexicon tree instead of a flat lexicon, or by introducing an estimation of the character duration on the decoding step. The simultaneous decoding of all stream character HMM in the first step of the two-level can also reduce the computational cost.

Table V
WORD RECOGNITION TIME (IN SECOND) WITH RESPECT TO THE NUMBER OF STREAMS K AND THE LEXICON SIZE V

	Two-level			**HMM-recombination**		
V=	**100**	**1000**	**1600**	**100**	**1000**	**1600**
K=2	4.2	5	7.5	1.2	13.2	23.8
K=3	9.4	10.1	13.4	15.2	162.8	298.6
K=4	10.9	12	17.5	182.4	1836	3283.2

VI. CONCLUSION

This paper has focused on the optimization of the computational efficiency of a multi-stream handwritten word recognition system. Two decoding strategies are compared and experimental results show that the multi-stream two-level decoding algorithm allows to speed up significantly the recognition process while maintaining the recognition accuracy. Consequently, it is now possible to combine many feature streams when dealing with large lexicons. Future works will focus on the improvement of the computational efficiency of the two-level algorithm as described in the previous section. We are also interested in combining other kinds of features providing more complementary information to further improve the results using a N-streams approach. Finally, we plan to exploit the multi-stream paradigm to combine a large number of HMMs resulting from Boosting/Bagging ensemble methods.

REFERENCES

[1] E. Grosicki, M. Carre, J. Brodin, E. Geoffrois: Results of the rimes evaluation campaign for handwritten mail processing. International Conference on Document Analysis and Recognition, pp 941-945, 2009.

[2] M. Pechwitz, S. Maddouri, V. Maegner, N. Ellouze, IFN/ENIT-DataBase for Handwritten Arabic words, CIFED'02, pp. 129-136, 2002.

[3] Y. Kessentini, T. Paquet, A. B. Hamadou, Off-line handwritten word recognition using multi-stream hidden markov models. Pattern Recognition Letters, vol 30 No. 1, pp 60-70, 2010.

[4] Y. Kessentini, T. Paquet, A. B. Hamadou, A Multi-Stream HMM-based Approach for Off-line Multi-Script Handwritten Word Recognition. 11th International Conference on Frontiers in Handwriting Recognition ICFHR'08, vol 1, pp. 147-152 ,2008.

[5] T. Artires, N.Gauthier, P.Gallinari, B.Dorizzi, A Hidden Markov Models combination framework for handwriting recognition, International Journal on Document Anlysis and Recognition (IJDAR), vol 5, N 4 pp 233-243, 2003.

[6] Y. Kessentini, T. Paquet, A. B. Hamadou. Multi-Script Handwriting Recognition with N-Streams Low Level Features. International conference on pattern recognition, International Conference on Pattern Recognition ICPR. pp 1-4, Tampa, Floride 2008.

[7] A. L. Koerich, R. Sabourin, and C. Y. Suen. Large vocabulary off-line handwriting recognition: A survey. Pattern Analysis and Applications, 6(2):97-121, 2003.

[8] A. L. Koerich, R. Sabourin, C. Y. Suen, and A. El-Yacoubi. A syntax-directed level building algorithm for large vocabulary handwritten word recognition. In Proc. 4th International Workshop on Document Analysis Systems, pp 255-266, Rio de Janeiro, Brazil, 2000.

[9] S. Madhvanath, V. Krpasundar, and V. Govindaraju. Syntatic methodology of pruning large lexicons in cursive script recognition. Pattern Recognition, vol 34, pp 37-46, 2001.

[10] H. Bourlard, S.Dupont. Sub-band-based Speech Recognition. In IEEE Int. Conf. on Acoust., Speech, and Signal Processing, pp. 1251-1254, 1997.

[11] H. Sakoe, Two-level DP matching - A dynamic programming-based pattern matching algorithm for connected word recognition, IEEE Transactions of the IECE of Japan, vol 27, pp 588-595, 1979.

[12] C. J. Wellekens, J. Kangasharju, C. Milesi, The use of meta-HMM in multistream HMM training for automatic speech recognition, Proc. of Intl. Conference on Spoken Language Processing (Sydney), pp. 2991-2994, 1998.

[13] S. Wshah, V. Govindaraju, Y. Cheng, and H. Li. A Novel Lexicon Reduction Method for Arabic Handwriting Recognition. In Proceedings of the 2010 20th International Conference on Pattern Recognition (ICPR '10). pp 2865-2868, 2010.

[14] J. T. Favata. Offline general handwritten word recognition using an approximate beam matching algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(9):1009-1021, 2001.

[15] R. Bippus and V. Margner. Script recognition using inhomogeneous p2dhmm and hierarchical search space reduction. In Proc. 5th International Conference on Document Analysis and Recognition, pp 773-776, 1999.