OPENHART 2013 EVALUATION: DESCRIPTION OF THE LITIS HANDWRITING RECOGNITION SYSTEM

Kamel Ait-Mohand and Thierry Paquet
Laboratory of computer science, information processing and systems (LITIS)
University of Rouen, France
{kamel.ait-mohand1,Thierry.Paquet}@univ-rouen.fr

***Abstract*—In this paper, we present the Arabic handwriting recognition system that was submitted to the 2013 NIST Open Handwriting Recognition and Translation Evaluation (OpenHaRT 2013). Our baseline recognition system is based on Hidden Markov Models and we also propose a lattice-based framework to combine the outputs from several different recognition engines.**

***Keywords*—Document recognition, Arabic handwriting, OpenHaRT, Hidden Markov Models.**

I. INTRODUCTION

In this paper, we present the Arabic handwriting recognition system developed by LITIS laboratory (laboratory of computer science, information processing and systems) to participate in the DIR task (Document Image Recognition) of the 2013 NIST Open Handwriting Recognition and Translation Evaluation (OpenHaRT 2013 [1]).

We have submitted two DIR systems for the constrained condition (the systems are developed using only the provided data resources) for the OpenHaRT evaluation. The first one is our baseline system which is based on Hidden Markov Models (section II) and the second one uses a lattice-based combination framework to combine the outputs from several HMM-based recognition engines (see section III). We could not participate in the DIR unconstrained condition due to lack of time.

The rest of the paper is organized as follows. In section II, we give details about our HMM-based baseline recognition engine. Section III describes the combination framework that is designed to combine the outputs of different recognition systems. In section IV, we detail the different recognition systems used for combination. Finally, we conclude about the relevance of our system in the last section.

II. BASELINE SYSTEM

The LITIS Arabic handwriting recognition system uses only the ressources given for the OpenHaRT competition. It takes as input the text line images extracted from document images (using the information collected in ground-truth MADCAT files) and outputs the text contained in these lines. Several steps are necessary to achieve this result:

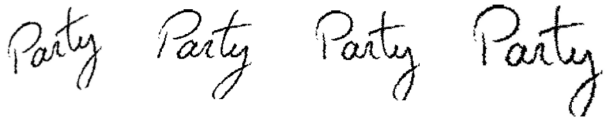

Fig. 1. Line normalization example (original image / deskewed image / deslanted image / normalized image).

A. PRE-PROCESSING

As a first step, we try to improve the image quality by conducting image-based pre-processing operations. The set of image processing operations used are: mathematical morphology operations, Wiener and bilateral filtering, contrast enhancement, adaptive binarization (Sauvola algorithm [2]). Then, a series of processing steps intended to "normalize" the style of writing are applied on each line image (see figure 1):

- Correction of the line slope or *deskew*: the slope of the line is estimated by a regression line calculated using the extrema of connected components. The skew is then corrected by a rotation of the line in the opposite direction.
- Correction of characters slope or *deslant*: the average slope angle of the characters is estimated from the Freeman writing contour direction histogram, then correction is performed by applying a linear transformation that shifts each foreground pixel depending on its position in the image.
- Normalization of the line height (corrected to a standard value of 48 pixels).

B. FEATURE EXTRACTION

We compare several descriptors on this character recognition task to be used by HMM-models. The results obtained show the superiority of the histograms of oriented gradients features described in [3]. The feature extraction module uses a sliding window approach and extracts on each position a set of 128 features based on the image gradient orientation with a 4×4 grid and 8 discrete values for the gradient orientation (see figure 2). We add five more features that capture the information about the position and the size of the connected components. Thus, the dimensionality of the feature vectors used in our experiments is 133.

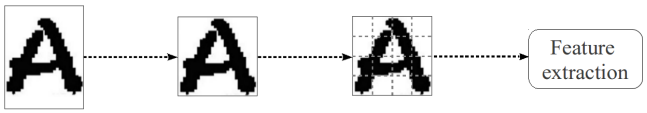

Fig. 2. Illustration (on a whole character image) of the feature extraction steps: (A) original image, (B) white spaces deletion above and below the character shape, (C) application of a 5×5 regular grid, (D) feature extraction.

Feature vectors are fed to a HMM based character recognition engine. For each text block, we use an approporiate set of models (Arabic letters) together with an appropriate language dictionary (60,000 words) and a language model. The textual content is found by searching for the best path in the graph of all possible hypotheses (Viterbi algorithm).

C. MODELING CHARACTERS WITH HMM MODELS

First, we process the OpenHaRT training dataset, which contains about 150,000 line images along with their ground truth files. Each line image is first processed by our pre-processing module before it is converted to a set of feature vectors that serves as input to the recognition engine.
This recognition engine consists of left-right continuous hidden Markov models and it is applied directly to the sequence of feature vectors extracted from an entire line image without pre-segmentation into words. We create line models by concatenating word models separated by an interword space model, making it possible to model all word sequences. Word models are created by concatenating character models (contextual Arabic letters, digits and punctuations, for a total of 144 models).

We use the HTK toolkit[1] (*Hidden markov models ToolKit* [4]) to create hidden Markov models used in our recognition engine. In order to optimize the learning of the models, we have to find the appropriate values for three hyper-parameters:

1) Number of states for each HMM.
2) Number of Gaussians per state.
3) Number of learning iterations.

For this purpose, we use the heuristic method of Zimmermann and Bunke procedure [5] in which the number of states of each left/right HMM character model is variable and proportional to the *average duration* of each character in the learning dataset. The optimal value of the number of states of each model, along with the number of Gaussians mixtures elements and learning iterations, are jointly optimized to maximize the recognition rate on a validation dataset.

Once the hyper-parameters are fixed, we apply the Baum-Welch algorithm to perform HMMs parameters estimation on the OpenHaRT training dataset. The values of the parameters that yields the best results are:

- Number of states per HMM ranging from 8 to 24 depending on the model
- Gaussian mixtures containing 20 components
- 20 iterations of the Baum-Welch algorithm

[1] http://htk.eng.cam.ac.uk/

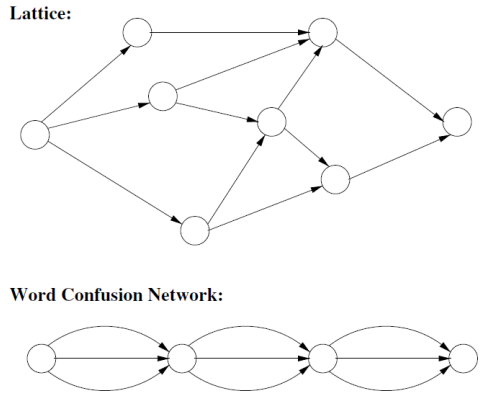

Fig. 3. Schematic representation of a word lattice (top) and a word confusion network (bottom).

Once the character models are learned, they can be used to find the text content of a line image. During the recognition phase, the line images to be recognized are inputted to the feature extractor. The obtained sequences of feature vectors are decoded using a two-pass forward-backward search [6]. The first forward pass uses a frame-synchronous beam search algorithm, with lexical constraints imposed by a bigram language model. While the backward pass is based on a stack decoding search and a trigram language model. To implement this decoder, we use the API (Application programming interface) of the Julius [7] software for its speed (the overall processing chain takes less than 2 minutes on an average for one document).

The accuracy of the HMM-based baseline system on the OpenHaRT evaluation dataset is quite low (23.33% word recognition rate). This can be explained primarily by the insufficient discriminating capabilities of our classifier. HMMs are not enough powerful to recognize with sufficient precision handwritten Arabic words such as those of the OpenHaRT database. Secondly, we lack some pre-processing operations like line-removal : our feature extractor is very sensitive to the presence of lines in the image and we noticed a much higher error rate on documents with lines in the background.

III. RECOGNITION SYSTEMS COMBINATION FRAMEWORK

We design a recognition framework to combine the outputs of several recognition engines in order to achieve improved recognition results. This combination method is the only novelty of this work but is still under development. When processing a line image, each recognition engine outputs a word lattice. This word lattice is a structured representation of the N-best recognition hypotheses along with the words confidence scores and time boundaries (see figure 3).

The first step of the combination consists of vertically concatenating lattices from all systems by merging their start and end nodes. This larger word lattice is further converted into a Confusion-Network (CN).

A confusion network (CN) is a weighted directed graph that forms a compact representation of word lattices in which competing word hypotheses, along with their posterior probabilities, are organized in time-ordered sets [8]. Each set (or

class, or node) contains one or more competing word hypothesis (figure 3). The probability of all the words contained in one set sum up to 1. Each set can also contain at most one empty word (ϵ). Due to the use of these empty words, a CN contains more paths than the lattice from which it is derived. These empty words also allow paths through the CN to have different lengths.

The advantage of this conversion of lattices to CNs is that we create new paths that combine words coming from the outputs of different engines. It also permits to sum the confidence scores of a word hypothesis if it is recognized by two or more systems, thus reinforcing the score of (a supposedly) good hypothesis. The use of CN representation, instead of word lattices or 1-best recognition, is also shown to reduce word error rate in automatic speech recognition [9].

To build these CNs from lattices, we use the SRILM toolkit [10] which implements an algorithm that is inspired by the original work in [8] but has lower complexity. In this algorithm, the word mesh is initialized with the highest-probability path through the lattice (this path is divided in distinct adjacent classes, so that each word forms a new class), and then successively aligns remaining partial lattice paths to the CN until all word hypotheses are processed (empty words are inserted where necessary to make these alignments one-to-one).

Once the CN is built, we rescore the word hypothesis using a 4-gram language model. Then the best path through the word CN is found by extracting, in each set of the CN, the best hypothesis (the highest probability word) which is also called consensus hypothesis.

IV. Designing several different recognition systems

In order that the combination of the outputs of different systems can improve their initial performance, it is important that the combined systems are significantly different, in terms of technology (type of classifier) or representation of the data (type of feature extraction).

Our combination framework is still under development at the beginning of the OpenHaRT evaluation campaign and requires significant improvements to be fully functional. Initially, we intended to use several HMM-based recognizers using different feature extractors. However, because of the lack of time, we are not able to develop several sufficiently powerful features extractors (we only have the histogram of oriented gradient features extractor detailed above). Therefore, we have decided to combine the outputs coming from the same system but applied several different image resolution levels (we used 3 different image resolution values, each resolution level is obtained by applying a Lanczos interpolation [11] on the initial image). The aim is that to get different HMM alignments on these feature frames and, therefore, different outputs depending on the used resolution.

Another problem due to the lack of time is the number N of sequences provided as output by each system, which we were forced to fix to a very small value (because obtaining the N-best sequences is quite time-consuming). We limit this value to only the 3 best sequences for each recognition engine. This partly explains the low accuracy of the results that was submitted at the end of the OpenHaRT evaluation campaign (21,60% of recognition rate). In addition, our system lacks a precise modeling of the language, in order to be more appropriate to the type of documents used in this evaluation campaign. We created a 60,000 words language model, estimated on a large text corpus of about 10,000,000 words, which is a mix of press articles (corpora extracted from Al Khaleej and Al Watan newspapers[2]) and some Arabic literature texts (retrieved from the Internet). We believe that this language model is too large to have good recognition accuracy on a handwritten recognition task and is also responsible of the high running time of the combination system (several hours for each document).

V. Conclusion

We designed a handwriting recognition engine based on Hidden Markov Models. To achieve a result improvement, we added a combination framework that uses the word lattices (that contain the N-best word sequences) outputted by several HMM-based recognition engines to build a large lattice containing all their word hypothesis. This large lattice is further converted into a confusion network which is decoded in order to get, as a final output, its highest-probability path.

Due to the lack of time, we have not been able to implement several feature extractors that are necessary for the combination framework to give significant improvement of accuracy. To overcome this deficiency, we obtain different outputs from the same HMM-based recognition engine by putting as input the line image at different resolution levels. Also because of the lack of time, we limited to 3 the number of N-best word sequences outputted by each recognition engine. All these shortcomings explain the poor accuracy of our final system. Another shortcoming of this recognition engine is the inappropriate language model used during the decoding process. We believe that this language model is too large (60,000 words) to have good recognition accuracy on a handwritten recognition task. In order to get better results, we plan to use smaller and topic-specialized language models. We also plan to modify the recognition engine by replacing the HMM models with more efficient classifiers. We believe that the use of discriminative classifiers such as neural networks can significantly improve the accuracy of our baseline system.

References

[1] A. Tong, M. Przybocki, V. Maergner, and H. El Abed, "Nist 2013 open handwriting recognition and translation (openhart'13) evaluation," in *Proceedings of the NIST 2013 Open Handwriting and Recognition Workshop*, 2013, in press.

[2] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern Recognition*, vol. 33, no. 2, pp. 225–236, 2000.

[3] J. Rodriguez and F. Perronnin, "Local gradient histogram features for word spotting in unconstrained handwritten documents," in *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, ser. ICFHR'08, Montreal, Canada, 2008, pp. 7–12.

[4] S. J. Young, G. Evermann, M. Gales, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, "The htk book version 3.4," 2006.

[2] https://sites.google.com/site/mouradabbas9/corpora

[5] M. Zimmermann and H. Bunke, "Hidden markov model length optimization for handwriting recognition systems," in *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition*, ser. IWFHR'02, 2002, pp. 369–.

[6] A. Lee, T. Kawahara, and S. Doshita, "An efficient two-pass search algorithm using word trellis index," in *Proc. iCSLP*, vol. 98, 1998, pp. 1831–1834.

[7] A. Lee and T. Kawahara, "Recent development of open-source speech recognition engine julius," in *Proceedings: APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference*. Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference, International Organizing Committee, 2009, pp. 131–137.

[8] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks," *eprint arXiv:cs/0010012*, Oct. 2000.

[9] G. Tur, D. Hakkani-Tur, and G. Riccardi, "Extending boosting for call classification using word confusion networks," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, vol. 1. IEEE, 2004, pp. I–437.

[10] A. Stolcke, J. Zheng, W. Wang, and V. Abrash, "Srilm at sixteen: Update and outlook," in *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, 2011.

[11] K. Turkowski, "Filters for common resampling tasks," in *Graphics gems*. Academic Press Professional, Inc., 1990, pp. 147–165.