The A2iA Arabic Handwritten Text Recognition System at the OpenHaRT2013 Evaluation

Théodore Bluche*†, Jérôme Louradour*, Maxime Knibbe*,
Bastien Moysset*, Mohamed Faouzi Benzeghiba* and Christopher Kermorvant*
*A2iA, 39 rue de la Bienfaisance, 75008 - Paris - France
†LIMSI CNRS, Spoken Language Processing Group, Orsay - France

***Abstract*—This paper describes the Arabic handwriting recognition systems proposed by A2iA to the NIST OpenHaRT2013 evaluation. These systems were based on an optical model using Long Short-Term Memory (LSTM) recurrent neural networks, trained to recognize the different forms of the Arabic characters directly from the image, without explicit feature extraction nor segmentation. Large vocabulary selection techniques and n-gram language modeling were used to provide a full paragraph recognition, without explicit word segmentation. Several recognition systems were also combined with the ROVER combination algorithm. The best system exceeded 80% of recognition rate.**

Keywords*—*OpenHaRT, Recurrent Neural Networks, ROVER, Large vocabulary Handwriting Recognition

I. Introduction

Handwritten Arabic writing is challenging for automatic recognition systems. This handwriting is highly cursive and the word segmentation is difficult due to the presence of space within words. The written language is inherently ambiguous due to the absence of vowels, and it contains many small marks (diacritics) which are used to disambiguate the different possible meanings. Finally, the Arabic language is morphologically rich, which results in a very large number of possible word forms.

International evaluations of handwritten Arabic recognition systems have been organized since 2005, first on a very simple task: small vocabulary isolated word recognition [1]. On this task, the systems have reached a plateau around 7-8% error rate [2]. The evaluations are now oriented toward large vocabulary text line recognition, since the first OpenHaRT evaluation in 2010 [3].

We present in this paper the systems developed at A2iA for the OpenHaRT2013 evaluation [4]. In 2010, A2iA submitted systems based on word segmentation using a contextual Hidden Markov Model (HMM) with a sliding window feature extraction [5]. In 2013, the word positions were not available: the systems had to perform a full text line recognition. The systems developed by A2iA for the 2013 evaluation were based on recurrent neural networks and used large vocabulary recognition techniques: large lexicon optimization, language models and system combination at sentence level. The different aspects of these systems are described in the following sections.

II. Data and Task Description

The OpenHaRT evaluation series focuses on recognition and translation technologies of Arabic script in document images. For this evaluation, A2iA only addressed the Document Image Recognition (DIR) task which measures the system capability in recognizing the text in the document image given the text line segmentation.

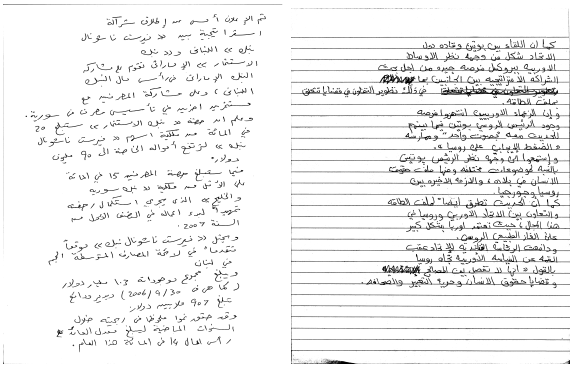

Fig. 1. Two examples of documents from the OpenHaRT dataset.

The documents provided for the evaluation, shown in Figure 1, consisted in single pages containing one or several paragraphs of handwritten text copied from an electronic source. This source could either be from the news, which makes the textual content less spontaneous compared to the Rimes database [6] for example, or from the web, which is less formal.

The documents were written by 455 native Arabic scribes, originating from 16 different countries, as shown on Figure 2. Each scribe contributed to the creation of 120 documents on average, as shown on Figure 3. The test set was composed of documents written by scribes who already contributed to the training set as well as unknown scribes.

The writing conditions of the documents were rather favorable to automatic recognition systems: the scribes wrote many documents (up to 600) and some of them can be found both in the training and in the evaluation set. This situation is closer to what one can encounter in historical document recognition, for example census indexation, where the same scribe has written many documents, than in daily incoming mail processing, where each document is produced by a different writer.

III. System Components Description

A. Image preprocessing

Several algorithms were tested to clean the text line images. In particular, we applied skew and slant correction, morphological denoising, morphological background line removal and

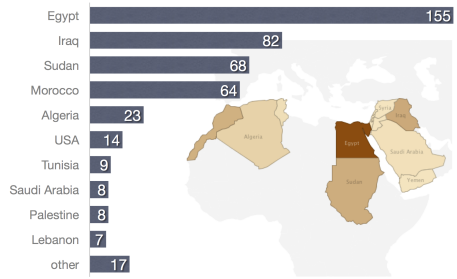

Fig. 2. Geographic origin of the scribes (country where they were born).

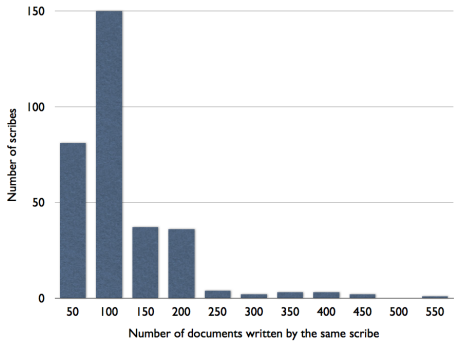

Fig. 3. Histogram of frequencies of the number of documents produced by the same scribe.

removal of ascenders and descenders belonging respectively to the lower and upper text lines. Preprocessing has been done both on images used for the training and for the decoding. However, none of the preprocessing, tested either in isolation or in combination, improved the recognition error rate on a validation set. Therefore, the final system does not include any preprocessing algorithm. We suspect that the database was large enough and homogeneous across the training and test sets, in terms of slant, noise and rules lines, so that the denoising and pre-processing was not needed.

B. Optical Model: Multi-Directional LSTM RNN

At the heart of the text recognition system, the optical model is in charge of estimating the probability of all symbols at different locations sampled accordingly to the decoding direction. All the symbols belong to a predefined set containing the symbol of the alphabet of the target language, the digits and the punctuations marks. The task of training a function which maps an image (2D signal) to a 1D sequence of symbols, *a.k.a.* "Temporal Classification", is different from the static classification problems in that the alignment between the input pixels and the target sequence is unknown. In practice, it happens when the segmentation of each single symbol is not available or is too complex. In [7], the author showed how to effectively sum the contribution of all possible alignments and compute the average Negative Log Likelihood (NLL) of the target sequences, as well as the derivatives. It allows to perform gradient descent and train neural networks that output framewise output probabilities (one normalized value for each symbol and each frame).

Our optical model was a Recurrent Neural Network (RNN) with 2D Long Short-Term Memory units (LSTM) [8] to model both local and scattered dependencies. We used the traditional LSTM units without the so-called peephole connections [9], because preliminary experiments showed that this modeling trick did not improve the results and sometimes made training unstable. However peephole connections could be useful for other applications, *e.g.* when the goal is to finely discriminate between frequencies in periodic patterns.

We used the network architecture described in [10], with exactly the same filter sizes that are well suited to images in 600 dpi. The RNN was Multi-Directional, which means that four separate LSTM layers were used to propagate the information in the 4 possible directions. In our CPU implementation, the recurrences in the four directions were computed on four different threads. This simple speed-up trick was welcome given that RNNs were relatively long to train, due to the fact that recursive operations are costly.

1) RNN optimization details: We trained the RNN to recognize all symbols in the training data, as well as an additional special symbol which stands for the blank, as described in [7]. Besides, we observed that training directly on located lines produces an awkward behaviour of the training process. The training cost function does not really decrease until a large number of updates, and sometimes it does not at all. One of the main reasons is that aligning long sequences is impossible at the beginning of the training process, when the RNN output probability values are uninformative. A natural solution was to train RNN *first on isolated words*, and *then on lines*. This two-step training schedule greatly helped to reduce convergence times and enabled to improve accuracy. The idea of learning first on easiest samples before learning on everything has already be shown to be efficient with Stochastic Gradient Descent (SGD) [11]and has been further developed for training recurrent neural networks in [12].

We trained up to 11 RNN Optical Models with a view to combining them as described in section V. All these RNNs were first initialized randomly using a Gaussian distribution with 0.1 standard deviation, and picking a different random seed for each RNN. Then all these RNNs were trained by running SGD on the NLL of target sequences with a constant learning rate of 0.001.

2) RNN training data: In the first training step, which consisted in training on isolated words, all the RNNs were trained on different subsets of data (randomly chosen and all disjoint). Then in the second training step, which consisted in training on lines, we used all the data available in "Phase 1/2/3 - Training" [4], iterating on lines in a pseudo-random order. To perform early-stopping and model selection, we used the "Phase 1 - Eval" set. We observed that among the 11 RNNs we trained, two of them were performing significantly worse: we decided to discard them from further combination.

We used the bidi algorithm [13] to convert strings of Arabic characters into strings of Arabic forms, and used these forms as target labels. The motivation was to exploit well-established prior knowledge about the Arabic script, by using labels that are more representative of the visual content. We also flipped the line images horizontally so that the decoding direction is coherent with the reading order and with direction of the language model.

Finally, note that 2D-LSTM RNN [10] did not rely on an

LM ORDER	SMOOTHING	PPX
3-gram	WITTEN-BELL	1180
	WITTEN-BELL INTERPOLATED	1149
	KNESER-NEY	1125
	KNESER-NEY INTERPOLATED	975
4-gram	KNESER-NEY INTERPOLATED	991

TABLE I. PERPLEXITY OF DIFFERENT LM ON THE PHASE 1 EVAL DATA.

SOURCE	NB OF TOKENS	INTER. COEFF.
AFP	121M	0.10
HYT	187M	0.46
NHR	149M	0.21
UMH	1M	0.10
XIN	34M	0.13

TABLE II. NUMBER OF TOKENS AND INTERPOLATION COEFFICIENTS FOR A TRIGRAM LANGUAGE MODEL.

ad hoc feature extraction: they were designed to be directly fed with 2D structure such as images. The only operation on the original input images was the standardization of pixel values. They were originally in the range [0,255], and it was empirically observed that SGD was more effective when inputs were centered and within a sane range. Pixel values were normalized by subtraction of the mean and division by the variance, so as to have zero mean and unit variance. The empirical mean and variance were computed on a subset of training images.

C. Vocabulary selection and Language modeling

The importance of using statistical n-gram language models in handwritten recognition systems has been demonstrated in several studies [14] [15]. To improve the quality of these models, the available text has to be first cleaned and normalized. This normalization consisted of applying some basic rules, including diacritics removal, punctuation marks and brackets from words separation, removing words with noisy characters (i.e., characters that are not supported by the recognizer), Lam-Alif normalization (converting the different written forms of the ligature Lam-Alif to their simple written forms), and splitting numbers into separate digits.

Given the characteristics of the OpenHaRT training data, which was composed of three sets (Phase 1, 2 and 3) with different sizes and potentially different lexicons, the vocabulary was selected using the technique described in [16]. The count (frequency) $C(w_i)$ of a word w_i was estimated as a linear combination (interpolation) of word counts $C(w_{ik})$ (i.e. unigrams) over the three data sets:

$$C(w_i) = \sum_{k=1}^{K} \lambda_k C(w_{ik}) \quad (1)$$

where K was the number of data sets. The weights λ_k were optimized using EM algorithm to minimize the perplexity on the held-out data which is composed of the DevTest Phase 1 and 2. A set of 60K words was then selected based on their counts (frequencies). The Out-of-Vocabulary (OOV) rate estimated on the Eval data set (Phase 1) was equal to 9%. This vocabulary was used to build statistical n-gram language models for both *constrained* and *unconstrained* conditions.

For the *constrained* condition, several language models with different order and smoothing techniques were generated. Table I reports the perplexity of these models on the Eval data set Phase 1. No pruning was performed. The interpolated Kneser-Ney 3-gram model performed the best. This model was then converted to a grammar FST to be used in the Kaldi decoding graph.

With respect to the *unconstrained* condition, the language model was generated using only the Arabic GigaWord database [1]. This database consists of five different Arabic news sources (afp, hyt, nhr, umh and xin). These sources differ by their sizes and their relevance to the target data (OpenHaRT data). In such training condition, the best practice is to build a language model by interpolating several source-dependent language models.

The text from these sources was first normalized using the rules described above. Language models with different order were then generated for each data source using Kneser-Ney smoothing, and an interpolated LM was finally generated. Interpolation coefficients were estimated on a development data composed of the three OpenHaRT training data sets.

Table II reports the number of tokens (after text normalization) and the estimated interpolation coefficients for a trigram language model.

Due to computational constraints, in particular the limited memory size, the interpolated big LM was severely pruned using entropy pruning scheme [17]. However, in [18], it has been shown that such pruning techniques with a Kneser-Ney smoothed LM might results in a poor language model quality. This is because in the Kneser-Ney smoothed LM, the probability of a word in the lower-order model is not estimated according to the maximum likelihood criterion, but according to the number of different contexts in which the word appears. As a remedy to this issue, an entropy pruning variant, proposed in [19] and implemented in the SRILM toolkit, was applied. In this case, a target language model is used to estimate the lower-order word probabilities. This model should not be a Kneser-Ney model. For the A2iA system, the target language was trained on the OpenHaRT data sets using Witten-Bell smoothing.

IV. DECODING

The decoding was performed using the Kaldi toolkit [20], which is based on Finite-State Transducers (FST). This procedure required the construction of a decoding graph that integrates the main system components.

A. Decoding graph construction

The recurrent neural network provided a sequences of predictions at character (or Arabic forms) level. These predictions were combined with a lexicon and language model to produce the final recognition result. In order to achieve this, we composed Finite-State Transducers (FST) and used Kaldi to find the best transcription from the character predictions.

HMM transducer: Each character or form was represented as an Hidden Markov Model (HMM) with only one state, a self-loop transition, and a transition to the next model. The emission model of the HMMs was the RNN.

[1]LDC Catalog No.:LDC2006T02

Lexicon FST: This FST transformed a sequence of characters and blank symbols into words. We took into account the particular behavior of the last layer of the RNN (the so-called Connectionnist Temporal Classification (CTC) layer) in the structure of the FST. Each character was modeled as one transition preceded by an optional transition, which input is the blank symbol and output is ϵ . When a character was doubled in a word, the blank transition is not optional anymore. Note that for arabic, the decomposition of a word into presentation forms was performed with the open-source *fribidi* [13] algorithm.

Grammar FST: The language model was created with the SRILM toolkit [21] and transformed into a FST with Kaldi.

B. Decoding strategies

Once built and composed, the final FST was a decoding graph, taking as input the characters - plus blank - predictions provided by the optical model and outputting the recognized sequence of words. We used the decoder provided within the Kaldi toolkit.

An HMM-based recognition system expected likelihoods $p(observation|state)$, while the RNN output are posteriors $p(state|observation)$. We divided the state posteriors by the state priors, weighted by $0 < \alpha < 1$: $p(state|observation)/p(state)^{\alpha}$.

In practice, we did not have the prior for the *blank* label. We explored two strategies. First, we estimated the priors as the average of the posteriors in the training set. We observed that the *blank* label had a high frequency (almost 80%). The results with this method showed a high number of deletions.

In an attempt to reduce the deletion rate, we adopted a second strategy. It consisted in assigning a fixed prior for the blank label, $p(blank) = \kappa$, and an uniform prior for the other labels $p(label \neq blank) = \frac{1-\kappa}{N_{labels}}$. This model yielded a lower deletion rate, at the expense of a higher global word error rate. This kind of model was useful when combined with other systems.

Two kinds of decoding were performed, corresponding to the two evaluation constraints. The first one was for the *constrained* evaluation, where only OpenHaRT data are used to build the trigram language model - which thus has a limited size. The decoding could be performed in a single pass.

For the *unconstrained* evaluation, the language model was significantly larger, so we first extracted lattices using a unigram LM, which we rescored with the full trigram.

V. System Combination

Overall, 9 RNNs were trained and kept, three of which explicitly model spaces between words. We had two *"prior models"* (estimated from training set and blank-penalty). The accuracies of individual systems are reported on table III.

We used the ROVER technique [22] to combine the recognition results of the systems. One option was to include all systems in the combination. This combination yielded 76.0% accuracy, which is a 1% absolute improvement over the best systems.

A second option was to try to select a subset of all systems. Indeed, all systems were not equally useful in compensating for the errors of the others. Moreover, running 15 systems in order to combine their output was time-consuming. We adopted an heuristic iterative approach. In the first pass, we added each system in turn to the combination, in descending order of accuracy. If the resulting combination improved the accuracy, we kept the considered system in the combination and record the relative improvement brought. Otherwise, we discarded it. We repeated the process in subsequent passes, ordering systems by their relative improvement rather than accuracy, until no system is thrown away.

RNN	Priors	Accuracy
Without space label		
RNN1	training set	74.8%
RNN2	training set	74.9%
RNN3	training set	75.0%
RNN4	training set	73.9%
RNN5	training set	74.0%
RNN6	training set	75.0%
RNN1	blank penalty	72.5%
RNN2	blank penalty	72.5%
RNN3	blank penalty	73.2%
RNN4	blank penalty	71.1% *
RNN5	blank penalty	72.1%
RNN6	blank penalty	73.3%
With space label		
RNN7	training set	75.0% *
RNN8	training set	75.0% *
RNN9	training set	75.0% *
ROVER Combinations		
All systems		76.0%
Selected systems (*)		**76.7%**

TABLE III. ROVER Combination (systems marked with * correspond to the selected combination)

We do not argue that the combination obtained by this algorithm is the best possible one. In particular, we observed that the order in which the systems are added in the combination has an effect on the resulting accuracy. We may throw away some systems too soon, and a more elaborate search strategy could probably result in a better combination. Yet this method brought the accuracy up to 76.7% on Phase 1 - Eval, improving the previous result by 3% relative.

VI. Results

The recognition rates of the A2iA systems at the 2013 evaluation are shown on Table IV. According to NIST rules, the results of the other systems in the evaluation can not be reported in this paper but can be found on the NIST web site [4] or in each participant's paper [23].

The first conclusion is that the recognition level has been greatly improved since the 2010 evaluation. In 2010, the best A2iA system reached a recognition rate of 62.3%, whereas in 2013 it reached 81.6% in the same conditions (unconstrained).

Second conclusion, the combination of several recognition systems improved the recognition rate from 79.4% to 79.9%, which is somewhat of a limited improvement considering the burden of training and combining several systems. This result can be explained by the training of the RNN : after a different initialization and training on different sets of words, they were trained on the same set of lines. They may have converged to very similar systems, not counterbalancing the recognition errors in a combined system.

System	HART13 Eval		
	Newswire	Web	All
Single RNN constrained	82.9	75.7	79.4
Multiple RNN constrained	83.4	76.2	79.9
Single RNN unconstrained	86.4	76.6	81.6

TABLE IV. RECOGNITION RATES OF THE DIFFERENT A2IA SYSTEMS AT THE OPENHART 2013 EVALUATION.

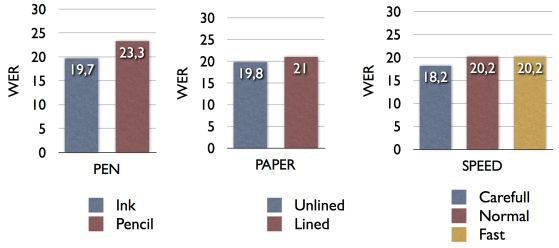

Fig. 4. Recognition results under different writing conditions.

The detailed results reveal that the documents copied for news sources are better recognized and also take more advantage of larger language models than text form web sources. This result can be explained by the fact that texts from the web are less formal, therefore more difficult to capture with language models, making their recognition by the whole system more complicated.

The recognition results under different writing conditions are shown on Figure 4. The results are conform with what was expected : the recognition is better when the text is carefully ink-written on unruled paper. However, the difference with other writing conditions (pencil, ruled paper and fast writing) is limited : it seems that the database was large enough to learn the different writing conditions efficiently.

VII. CONCLUSION

This paper describes the systems submitted by A2iA to the Arabic handwriting recognition evaluation OpenHaRT2013. This system used an optical model based on recurrent neural networks and many techniques borrowed from large vocabulary speech recognition: large vocabulary selection, language models interpolation, ROVER system combination. The recognition rate has been greatly improved since the first evaluation in 2010 and exceeded 80%. Further improvement can be found in out-of-vocabulary word recognition and in better modeling texts coming form web sources.

ACKNOWLEDGMENT

This work was partly achieved as part of the Quaero Program, funded by OSEO, French State agency for innovation and was supported by the French Research Agency under the contract Cognilego ANR 2010-CORD-013.

REFERENCES

[1] V. Märgner, M. Pechwitz, and H. El Abed, "Arabic Handwriting Recognition Competition," in *International Conference on Document Analysis and Recognition*, 2005.

[2] V. Märgner and H. El Abed, "ICDAR 2011 Arabic Handwriting Recognition Competition," in *International Conference on Document Analysis and Recognition*, 2011.

[3] NIST, "NIST 2010 Open Handwriting Recognition and Translation Evaluation Plan - version 2.8," pp. 1–6, 2010. [Online]. Available: http://www.nist.gov/itl/iad/mig/hart2010.cfm

[4] ——, "NIST 2013 Open Handwriting Recognition and Translation Evaluation Plan - version 1.7," pp. 1–9, 2013. [Online]. Available: http://www.nist.gov/itl/iad/mig/hart2013.cfm

[5] A.-L. Bianne-Bernard, F. Menasri, L. Likforman-Sulem, C. Mokbel, and C. Kermorvant, "Variable length and context-dependent HMM letter form models for Arabic handwritten word recognition," in *Document Recognition and Retrieval Conference*, 2012.

[6] F. Menasri, J. Louradour, A.-L. Bianne-Bernard, and C. Kermorvant, "The A2iA French handwriting recognition system at the Rimes-ICDAR2011 competition," in *Document Recognition and Retrieval Conference*, vol. 8297, 2012.

[7] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *International Conference on Machine learning*, 2006, pp. 369–376.

[8] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning Precise Timing with LSTM Recurrent Networks," *Journal of Machine Learning Research*, vol. 3, pp. 115–143, 2002.

[10] A. Graves and J. Schmidhuber, "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks," in *Advances in Neural Information Processing Systems*, 2008, pp. 545–552.

[11] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *International Conference on Machine Learning*, no. 1330, 2009.

[12] J. Louradour and C. Kermorvant, "Curriculum Learning for Handwritten Text Line Recognition," in *submission*, 2014.

[13] "GNU FriBidi." [Online]. Available: http://fribidi.org

[14] A. Vinciarelli, S. Bengio, and H. Bunke, "Offline recognition of unconstrained handwritten texts using HMMs and statistical language models." *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 709–20, Jun. 2004.

[15] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–68, May 2009.

[16] W. Wang, "Techniques for effective vocabulary selection," in *European Conference on Speech Communication and Technology*, 2003.

[17] A. Stolcke, "Entropy-based Pruning of Backoff Language Models," in *In Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, VA, 1998, pp. 270–274.

[18] C. Chelba, T. Brants, W. Neveitt, and P. Xu, "Study on Interaction between Entropy Pruning and Kneser-Ney Smoothing," in *International Conference on Speech Communication and Technology*, 2010, pp. 2–5.

[19] A. Stolcke, J. Zheng, W. Wang, and V. Abrash, "SRILM at Sixteen : Update and Outlook," in *Workshop on Automatic Speech Recognition and Understanding*, 2011.

[20] D. Povey, A. Ghoshal, N. Goel, M. Hannemann, Y. Qian, P. Schwarz, J. Silovsk, and P. Motl, "The Kaldi Speech Recognition Toolkit," in *Workshop on Automatic Speech Recognition and Understandin*, 2011.

[21] A. Stolcke, "SRILM An Extensible Language Modeling Toolkit," in *International Conference on Spoken Language Processing*, 2002.

[22] J. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER)," in *Workshop on Automatic Speech Recognition and Understanding Proceedings*. IEEE, 1997.

[23] M. Hamdani, P. Doetsch, M. Kozielski, A. El-Desoky Mousa, , and H. Ney, "The rwth large vocabulary arabic handwriting recognition system," in *Submitted to DAS2014*, 2014.