White-Space Models for Offline Arabic Handwriting Recognition

Philippe Dreuw, Stephan Jonas, and Hermann Ney
Human Language Technology and Pattern Recognition, RWTH Aachen University
{*dreuw,jonas,ney*}*@cs.rwth-aachen.de*

Abstract

We propose to explicitly model white-spaces for Arabic handwriting recognition within different writing variants. Position-dependent character shapes in Arabic handwriting allow for large white-spaces between characters even within words. Here, a separate character model for white-spaces in combination with a lexicon using different writing variants and character model length adaptation is proposed. Current handwriting recognition systems model the white-spaces implicitly within the character models leading to possibly degraded models, or try to explicitly segment the Arabic words into pieces of Arabic words being prone to segmentation errors. Several white-space modeling approaches are analyzed on the well known IFN/ENIT database and outperform the best reported error rates.

1. Introduction

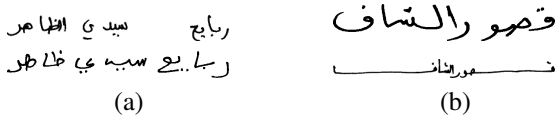
Especially in Arabic handwriting with its position-dependent shapes [5], large white-spaces can occur between isolated-, beginning-, and end-shaped characters (see Figure 1 (b)). As some characters are only connectable from the right side, such words have to be cut into pieces (Piece of Arabic Word (PAW)).

In previous Arabic handwriting recognition competitions [8, 7] it turned out that the relative error of most systems in general follows the frequency of the pieces of Arabic words (PAWs). Similar to silence modeling in automatic speech recognition, we propose to explicitly model these white-spaces in Arabic handwriting by different writing variants. Furthermore, a fast character model length adaptation is presented.

2. System Overview

Our hidden Markov model (HMM) based handwriting recognition system builds on a large vocabulary speech recognition system [4]. The system is Viterbi trained and uses a lexicon with multiple writing variants.

We are searching for an unknown word sequence w_1^N, for which the sequence of features x_1^T best fits to the trained models. We maximize the posteriori probability $p(w_1^N|x_1^T)$ over all possible word sequences w_1^N



(a) (b)

Figure 1. Two examples where each column shows the same Tunisian town name: HMM white-space models (a) and state-transition penalties (b) are important in Arabic handwriting recognition

with unknown number of words N. This is modeled by Bayes' decision rule:

$$\hat{w}_1^N = \underset{w_1^N}{\operatorname{argmax}}\{p(w_1^N)p(x_1^T|w_1^N)\} \quad (1)$$

$$p(x_1^T|w_1^N) \approx \max_{v_1^N}\{p_{\theta_{\mathrm{pm}}}^{\alpha}(v_1^N|w_1^N)p_{\theta_{em,tp}}^{\beta}(x_1^T|v_1^N)\} \quad (2)$$

with v_1^N a sequence of unknown writing variants, α a scaling exponent of the writing variant probability depending on a parameter set θ_{pm}, and β a scaling exponent of the visual character model depending on a parameter set $\theta_{em,tp}$ for emission and transition model.

For our system, we use appearance-based image slice features concatenated with overlapping sliding windows which can either be used directly as features [3], or reduced by linear feature reduction methods like PCA [2] or LDA [6]. Here, we use the image slices X_t and their spatial derivatives in horizontal direction $\Delta = X_t - X_{t-1}$ for the sliding window, which are later reduced by a PCA transformation matrix computed from the corresponding training folds. Opposed to a baseline dependent feature extraction [2], here we propose the extraction of additional virtual training (VT) samples [1] to train baseline robust character models, by simply shifting all training samples by $\pm\delta$ pixels along the y-axis. E.g., for $\delta = 3$, the training corpus is already enlarged by a factor of seven.

3. Visual Modeling

Each character model in our base system is modeled by a 3-state left-to-right HMM with three separate Gaussian mixtures (GMM) and a globally pooled covariance matrix. Additionally, a large stretching of long drawn-out characters occurs often in Arabic handwriting (see

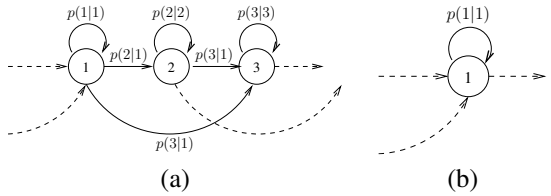


Figure 2. Different HMM topologies and transition penalties are used for character models (a) and white-space models (b).

Figure 1 (b)). Therefore, we use very low loop penalties but higher skip penalties for our HMM state transitions (see Figure 2 (a)).

3.1. White-Space Modeling.

In an HMM based handwriting recognition system, white-spaces usually are modeled within the character models. Instead of implicitly modeling white-spaces within these character models, we propose to explicitly model the white-spaces by training special white-space models. These white-space models are added to the character transcriptions in the lexicon. We propose several different setups for white-space modeling:

- no white-spaces (ns): i.e. the available ground truth annotation
- between word white-spaces (bws): a white-space character model is added only between Tunisian town names consisting of several sub-words
- between word and within word white-spaces (bwws): in addition to the 'bws' writing variant, an additional second writing variant where white-space character models are added between the models of isolated-, beginning-, and end-shaped characters (i.e., the PAWs) is added to the lexicon codebook.

Another possibility for white-space modeling would allow a recognition of PAW sentences instead of words, which is not focussed here.

Our proposed white-space character model uses a special and separate single-state HMM model with separate entry and exit penalties (see Figure 2 (b)). This allows to hypothesize the white-space character model even for very small gaps between characters or PAWs.

3.2. Model Length Adaptation

Due to ligatures and diacritics in Arabic handwriting, obviously some characters like *yaa* ي have a more complex appearance than others, e.g. *alif* ا. Furthermore, some ligatures have a nastaliq font. Especially the letter *kaaf* ك changes dramatically, depending on the position or the ligature (e.g. *kaaf with yaa* كي or *kaaf with alif* كا).

Here, we propose a fast character model length adaptation (MLA), which on the one hand leads to sharper models (i.e., a higher spatial resolution), and on the other hand to fewer white-space features in the character models. Based on an alignment dump of the training data on the *state-level*, it is possible to count the total number of observations which are aligned to the same 0-1-2 character model. In a first pass, these state counts are used to calculate the average length of each character seen in training. Each character in the lexicon codebook is updated by adding additional pseudo-characters to all writing variants depending on their average character length. The new lexicon with proportionally adapted model lengths of all character models, resulting in different number of states per character model, is then used for a second pass training. Opposed to the iteratively adaptation algortihm presented in [10], we adapt our lexicon using a single iteration only.

4. Experimental Results

The experiments are conducted on the IFN/ENIT database [9]. The database is divided into four training folds with an additional fold for testing [8]. The current database version (v2.0p1e) contains a total of 32492 Arabic words handwritten by more than 1000 writers, and has a vocabulary size of 937 Tunisian town names. Additionally, the submitted systems to the ICDAR 2007 competition [7] were trained on all datasets of the IFN/ENIT database and evaluated for known datasets. Here, we follow the same evaluation protocol as in ICDAR 2005 and 2007 competition.

4.1. Base System

Plain image features. For image size normalization, we empirically optimized in preliminary experiments the image scaling of the database images to 16 pixels height, while keeping their aspect ratio. Furthermore, this allows to model each character by a single 3-state model (c.f. section 3).

PCA based sliding window features. We optimized the PCA windowing parameters only on one split of the IFN/ENIT database. The results are shown in Figure 3. For all following experiments, we chose a PCA window-size 7 (i.e., $7 \times (16 + \Delta)$ features) with a window shift of 1 pixel (i.e., maximum overlap).

4.2. White-Space Modeling

The comparison of the three proposed white-space modeling approaches is presented in Figure 4. Additionally, the evaluation of position-independent character models (NS-Nopos) is shown which is obviously worse than the position-dependent models. The word-error-rate for the ground truth annotation task without white-space models (ns) is reduced by all proposed

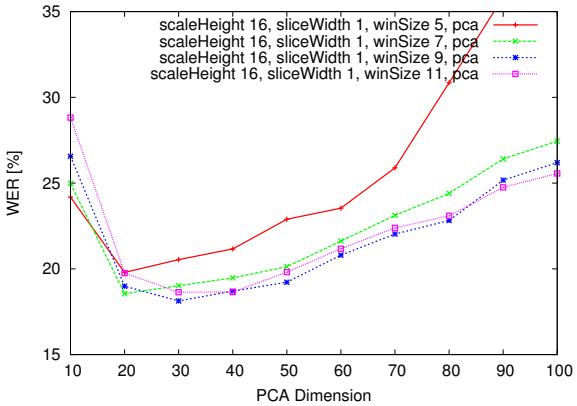


Figure 3. Results for different PCA window sizes and dimensions (for training sets abc and test set d).

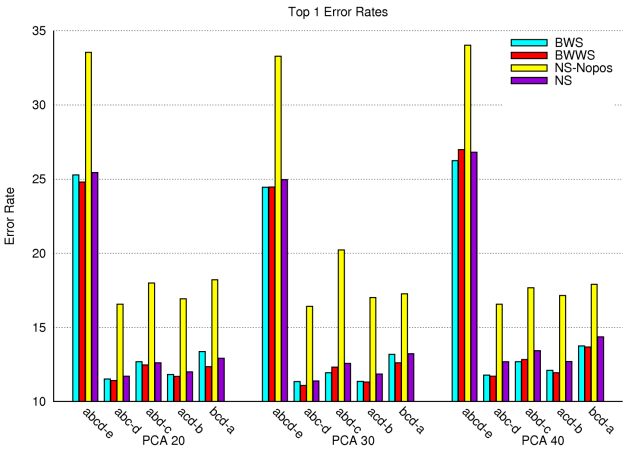


Figure 4. Visualization of different white-space modeling approaches and PCA dimensions: the proposed bwws-modeling (2nd bar, red) outperforms all other approaches

white-space models: The overall best results are obtained for a PCA reduction to 30 components in combination with the proposed between word and within word white-space (bwws) modeling.

After optimization of the transition and writing variants parameters (e.g., α and β in Equation 2) on the d-fold only, the results of our base system without adapted model lengths are shown in Table 1. It can be seen that the optimized system generalizes well on all other folds too.

4.3. Model Length Adaptation

The experiments in Figure 4 show that white-space modeling helps. In order to further reduce the amount of white-spaces in the character models, the system is retrained with a character length adapted lexicon (c.f. subsection 3.2). The average lengths were calculated on the corresponding training folds only. The model length adaptation (MLA) results in Table 1 clearly show that

Table 1. Top 1 word recognition rate (WRR) and corresponding character recognition rate (CRR) results for PCA-30 features and 'bwws' white-space modeling using the base system, the model length adapted system (MLA), and the model length adapted system with additional virtual training samples (MVT)

Train	Test	WRR [%]			CRR [%]		
		Base	MLA	MVT	Base	MLA	MVT
abc	d	89.22	90.71	**92.86**	96.14	95.99	**97.02**
abd	c	88.56	89.87	**91.86**	95.63	95.91	**96.62**
acd	b	89.12	90.95	**92.55**	96.19	96.38	**97.13**
bcd	a	88.14	90.29	**92.32**	95.55	95.74	**96.63**
abcd [8]	e	76.46	79.68	**80.95**	90.99	91.22	**91.75**
abcde [7]	d	92.77	96.56	97.34	97.40	98.71	99.03
abcde [7]	e	82.00	89.91	90.88	92.74	95.90	96.27

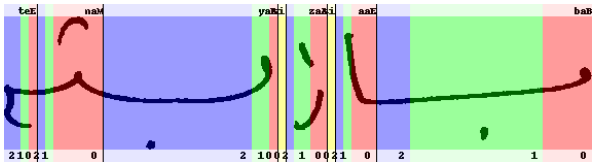


Figure 5. Alignment visualization: low HMM transition loop penalties in Arabic handwriting are important (i.e., blocks with the same background color)

the performance of the system using MLA is improved for the word recognition rate (WRR) *and* the character recognition rate (CRR) on all folds. Additionally, virtual training (VT) samples in combination with the MLA system (i.e., MVT) further improve the system performance in terms of WRR and CRR. Similar improvements are achieved for the base system in combination with VT data (not presented here). In particular, and to the best of our knowledge, these results outperform all error rates reported in the literature.

4.4. Visual Inspection

The visualizations in Figure 5, Figure 6, and Figure 7 show training alignments of Arabic words to their corresponding HMM states, trained with the final HMM base system without any model length adaptation. We use R-G-B background colors for the 0-1-2 HMM states, respectively, from right-to-left. The position-dependent character model names are written in the upper line, where the white-space models are annotated by 'si' for 'silence'; the state numbers are written in the bottom line. Thus, HMM state-loops and state-transitions are represented in Figure 5 by no-color-changes and color-changes, respectively.

In Figure 6 and Figure 7, it can be observed that the

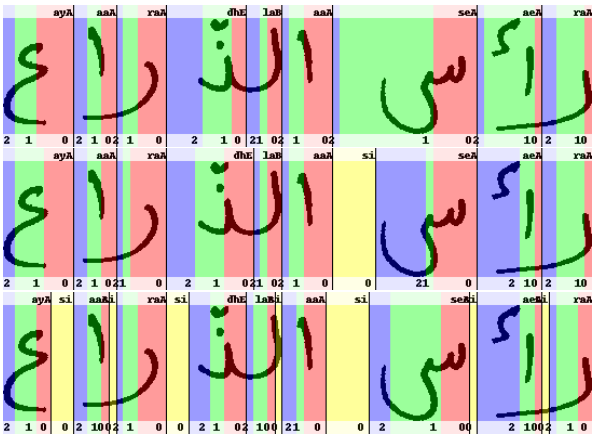


Figure 6. Alignment visualization: first row shows an alignment using ns-modeling, second row bws-modeling, and third row bwws-modeling. Due to the additional white-space models between characters (annotated by 'si', yellow background), the alignment is clearly improved.

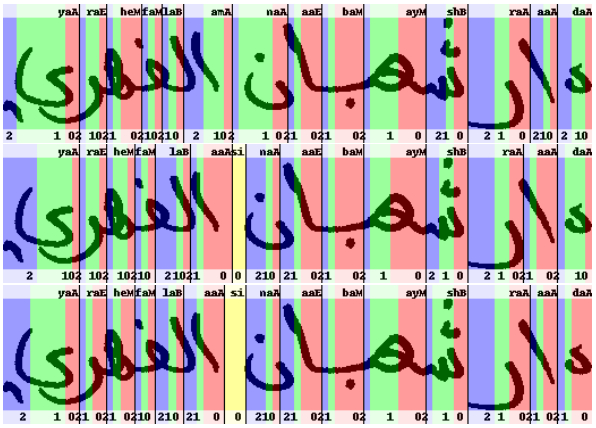


Figure 7. Alignment visualization (from top to bottom: ns, bws, bwws): due to the cursive handwriting style, the insertion of additional white-space models could worse the alignment. Here, the system opts for the correct writing variant without additional white-spaces between characters, i.e., the alignments for bws- and bwws-modeling are similar

white-spaces are implicitly modeled within the character models, if no white-space modeling is applied (first rows). The alignments are clearly improved in Figure 6, if additional white-space models are trained (second and third row). On the other hand it can be seen in Figure 7 that, due to the cursive handwriting style, the bwws-modeling approach selected the correct writing variant *without* additional white-space models between the characters (i.e., the 'bws' writing variant).

5. Conclusions

We presented an HMM based system for offline Arabic handwriting recognition which explicitly models the white-spaces between characters and pieces of Arabic words. The proposed novel white-space models for Arabic handwriting could improve on all cross folds the system accuracy and outperforms the best reported error rates. A visual inspection of the trained models showed the need for an accurate modeling and adaptation of the character lengths. The usage of additional virtual training samples again strongly improved the results on the IFN/ENIT database.

6. Acknowledgment

We would like to thank Thomas Deselaers and Christian Gollan for their support.

References

[1] C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector machines. In *NIPS 97*, volume 9, pages 375–385, Vancouver, Canada, dec 1997.

[2] R. El-Hajj, L. Likforman-Sulem, and C. Mokbel. Arabic handwriting recognition using baseline dependant features and hidden markov modeling. In *ICDAR*, volume 2, pages 893–897, Seoul, Korea, Aug. 2005.

[3] D. Keysers, T. Deselaers, C. Gollan, and H. Ney. Deformation models for image recognition. *IEEE PAMI*, 29(8):1422–1435, Aug. 2007.

[4] J. Lööf, C. Gollan, S. Hahn, G. Heigold, B. Hoffmeister, C. Plahl, D. Rybach, R. Schlüter, and H. Ney. The RWTH 2007 TC-STAR evaluation system for european english and spanish. In *Interspeech*, pages 2145–2148, Antwerp, Belgium, Aug. 2007.

[5] L. M. Lorigo and V. Govindaraju. Offline Arabic handwriting recognition: A survey. *IEEE PAMI*, 28(85):712–724, May 2006.

[6] Z. A. Lu, I. Bazzi, A. Kornai, J. Makhoul, P. S. Natarajan, and R. Schwartz. A robust language-independent OCR system. In *AIPR Workshop: Advances in Computer-Assisted Recognition*, volume 3584 of *SPIE*, pages 96–104, 1998.

[7] V. Märgner and H. E. Abed. ICDAR 2007 Arabic handwriting recognition competition. In *ICDAR*, volume 2, pages 1274–1278, Sept. 2007.

[8] V. Märgner, M. Pechwitz, and H. Abed. ICDAR 2005 Arabic handwriting recognition competition. In *ICDAR*, volume 1, pages 70–74, Seoul, Korea, Aug. 2005.

[9] M. Pechwitz, S. S. Maddouri, V. Mägner, N. Ellouze, and H. Amiri. IFN/ENIT-database of handwritten Arabic words. In *Colloque International Francophone sur l'Ecrit et le Document (CIFED)*, Hammamet, Tunis, Oct. 2002.

[10] M.-P. Schambach. Model length adaptation of an hmm based cursive word recognition system. In *ICDAR*, Edinburgh, Scotland, UK, Aug. 2003.