Prototype Integration in Off-line Handwriting Recognition Adaptation

Gregory R. Ball and Sargur N. Srihari
Center of Excellence for Document Analysis and Recognition (CEDAR)
University at Buffalo, State University of New York
Amherst, New York 14228
srihari@cedar.buffalo.edu

Abstract

Writer adaptation or specialization is the adjustment of handwriting recognition algorithms to a specific writer's style of handwriting. Such adjustment yields significantly improved recognition rates over counterpart general recognition algorithms.

We present a discussion of a method of prototype integration for writer adaptation and evaluate the results on English and Arabic datasets. The writer adaptation model we use is an iterative bootstrapping model which adapts a writer-independent model to a writer-dependent model using a small number of words achieving a large recognition rate increase in the process, utilizing a confidence weighting method which generates better results by weighting words based on their length.

The Arabic testing set consisting of about 100 pages of handwritten text and was able to improve its recognition rate by over 10% using adaptation. The English testing set consisted of a subset of 3000 pages of handwritten letters written by 1000 authors. Similar to the Arabic set, a corresponding increase in performance was observed.

Keywords: writer adaptation, handwriting specialization, English handwriting recognition, Arabic handwriting recognition, off-line handwriting recognition

1 Introduction

Writer adaptation, specializing from a writer-independent handwriting recognition algorithm to a writer-specific one, has successfully been applied to cursive writing in Latin script both in the context of on-line recognition[16, 5, 4, 17] and off-line recognition[17, 10, 11, 9]. Supplementing the existing results for English handwriting adaptation, we presented the first adaptation results for Arabic in the literature[1]. Arabic, being a naturally cursive language, would seem a prime candidate for the improvements offered by writer adaptation.

In our version of writer adaptation, we use a writer independent system to select samples of the input document for use as unlabeled training data. By combining a traditional hidden Markov model (HMM) recognition engine with a writer adaptation model, we have achieved a significant reduction in our error rate. A key question for any writer adaptation method is how exactly to integrate an author's sampled writing into a writer-dependent model. We explore several different methods for integrating their learned handwriting samples. To evaluate performance for English, we tested on the CEDAR letter database [3]; for Arabic, we tested on same Arabic dataset as in our earlier paper [1].

In addition to evaluating the adaptation on English and Arabic, we explore variation in starting model size and various strategies in combining the newly learned characters with the writer-independent models.

2 Related Research

On-line handwriting recognition can use writer adaptation to create personalized systems by using supervised learning. Hand held devices, for example, can go through a training process to better recognize a writer's handwriting (enrollment adaptation [11]). As Subrahmonia et al mentioned in [16], "for the users who will make extended use of such a system the gain in productivity due to increased accuracy will offset the initial inconvenience of training." They bootstrap from a writer independent model, adapting the character models with writer input to create a writer dependent model. They found that a few hundred words of training data was able to reduce the error rate significantly. Senior and Nathan [13] were able to use a much smaller set of training words, as few as five, in order to reduce the error rate. They used the Maximum Likelihood Linear Regression technique to adapt the means and variances used in their HMM. Connell and Jain[5, 4] identified character styles (lexemes) of individual writers, and specialized the lexeme model to the writer's training data in order to deal with limited training data. More recently, Haluptzok et al [6] discussed several methods for an on-line system to adjust its recognition system to a specific user's writing involving such ideas as

enrollment or incremental adjustment.

Off-line handwriting recognition models can also adapt their independent models with relatively few words. Vinciarelli and Bengio [17] noted that they were able to adapt a writer independent system with 30 words. They attempted to train their system solely on the lone writers' words, and were not able to meet the performance of the adapted system until training with many more words, their lower bound estimate being at least 200. Nosary et al [10, 11, 9] used the recognition output from their system as training data, using batch adaptation as recognition progressed. In batch adaptation, system recognition output is stored and used at a later stage, as opposed to incremental adaptation where each new recognition result is used to improve the system.

3 Writer Independent Recognition

3.1 Recognition Algorithm

We approach the basic task of recognition with a HMM utilizing a lexicon, similar to the system we used for word spotting[2]. Each word to be recognized is segmented based on ligatures and whitespace separation occurring inside the word, as described by Kim and Govindaraju[8]. Ligatures, as they noted, are strong candidates for segmentation points in cursive scripts. We extract ligatures in a similar way. If the distance between y-coordinates of the upper half and lower half of the outer contour for a x-coordinate is less than or equal to the average stroke width, then the x-coordinate is marked as an element of a ligature. Concavity features in upper contour and convexities in the lower contour are also used to generate candidate segmentation points, which are especially useful for distinct characters which are touching, as opposed to being connected. A ligature will cause any overlapped concavity features to be ignored. For a given x-coordinate, if a concavity and convexity overlap, a segmentation point is added for that x-coordinate.

The segmentation algorithm used over-segments words in the hopes of avoiding incorrectly putting more than a single character into a segment. For dealing with this over-segmentation, the segments need to be rejoined into candidate characters. To do this, all reasonable combinations of candidate characters are considered with a version of the Viterbi algorithm, using the sequence of characters for deciding upon a reasonable range for the number segmentation points in a candidate word. Each subcharacter is assigned to a character appearing in the lexicon word. The subcharacters are placed into consecutive groups corresponding to the letter known to be in the same relative location in the word (i.e. after the group of subcharacters assigned to the first character, the group of subcharacters for the second character begin). The features of the composed characters are compared to features of prototype images of the characters. The groupings are maximized such that the overall score for the characters (groups of subcharacters) is best (the composed characters are a closest match to some of the prototypes). A score that represents the match between the lexicon and the candidate word image is then computed. The score relates to the individual character recognition scores for each of the combined segments of the word image. The word with the smallest average distance overall from its prototype characters is selected as the best candidate.

3.2 Dots and Diacritical Marks

The algorithm that divides the words into segments proceeds horizontally. Arabic words contain a larger number of dots and diacritical marks as compared with English. Rather than attempt to assign dots directly to characters, we instead designed a filter to separate the dot stream from the base word. The dot stream can later be used as an error correction stream to validate or adjust the results of the recognition. A naive approach of assigning the dots to the closest subcharacter resulted in poorer performance than using the filter, presumably because it was more difficult to generalize the character shapes considering the range of positions where the dot might appear and the range of shapes the dots might take. While dots are essential for identifying individual Arabic characters out of context, in the context of a word, the ambiguity introduced by not originally considering the markings is not considerable, the vast majority of the lexicon being pruned out by the initial recognition step. An additional benefit of using this approach is that it is language independent with respect to Latin and Arabic scripts.

3.3 Character Models

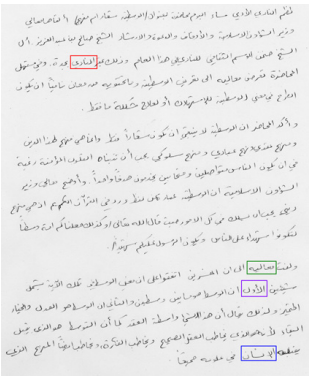
For English recognition, uppercase and lowercase versions of the 26 letters were modeled along with the 10 numerals used in English writing for a total of 62 character classes modeled.

The code words were trained using 21,054 character images extracted from handwritten images of United States city names segmented by the method used in recognition. The initial results were clustered using K-means using a fixed number of clusters.

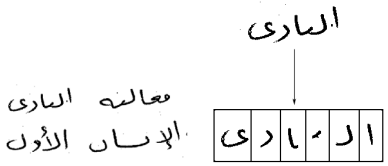
For Arabic, each position-based variation of each Arabic characters were modeled with a prototype grouping. Common character combinations that are not easily horizontally segmented were modeled separately. Based on the predictable nature of the positional based sequence, the lexicon is translated into a predicable sequence of the approximately 200 character classes. The character classes consist of the letters of the Arabic alphabet, with separate classes for each position style (initial, medial, final, separate), as well as classes for several combination

characters such as ”lam-alef” and ”lam-hah”.

A set of approximately two thousand manually cut Arabic characters (about 10 per class) were used to initialize the character prototypes. The IFN/ENIT[12] database was then used to train the system. Training was accomplished by executing the described recognition on each word in three of the IFN/ENIT sets with a single word lexicon consisting of the ground truth. The best supersegments of segments were extracted based on the recognizer output. The features of all the candidate letters were extracted and clustered using a K-means clustering algorithm into clusters. The original character prototypes were replaced with set numbers of representative prototypes for each character. The representative prototypes were combined and used as our writer independent model.



(a)



(b) (c)

Figure 1. The page is first recognized as a whole; (a) and the words recognized with most confidence are identified, (b) extracted, (c) split according to the recognized truth and used to adapt the writer-independent model into a writer-dependent one.

3.4 Feature Set

For both English and Arabic, we use the WMR (word model recognition) feature set, which consists of 74 features. Two are global features (aspect and stroke ratio) of the entire character. The remaining 72 are local features. Each character image is divided into 9 subimages. The distribution of the 8 directional slopes for each subimage form this set (8 directional slopes $\times$ 9 subimages = 72 features). $Fl_{i,j} = s_{i,j}/N_i S_j, i = 1, 2, ..., 9, j = 0, 1, ..., 7,$ where $s_{i,j}$ = number of components with slope j from subimage i, N_i = number of components from subimage i, and $S_j = max(s_{i,j}/N_i)$.

4 Writer Adaptation

Our adaptation algorithm makes use of the words the system is most confident of being correct. Since it is an off line algorithm, we cannot ask the user for additional input, but can proceed by treating the sequence of words with highest confidence as a sequence of truthed handwriting samples. The document consists of a set of words W. We create an empty set W' representing words already used for adaptation. The algorithm first recognizes the entire document, assigning the top-1 choice to each word in the document. The related confidence score for each word is also stored. The confidence metric used is the average Euclidean distance of each supersegment from its corresponding prototype.

After processing the entire document, the confidence scores are ranked. The words assigned a better confidence score are more likely to be recognized correctly. Since the word with the best confidence score is most likely to be correct, it is added to W' (see Figure 1. The word's most likely segmentation is created with respect to the recognized sequence of characters by the recognizer. The features are extracted from the characters, and added to the character prototype sets. The process is repeated, taking the word $w \in W$ with the best score such that $w \notin W'$ at each step. As a result of adding the feature set from the component characters, the original author's individual style is incorporated into the prototypes.

4.1 Batch version

The recognition step of this algorithm is the most expensive operation. One way to achieve reasonable efficiency is to adapt to more than simply the best word at each iteration. We reasoned that it was likely that the top several words are recognized correctly. We took the top ten words at each iteration and did not observe a significant change in recognition rate, while increasing the speed of the algorithm by several times.

4.2 Improved confidence metric

When recognized incorrectly, very short words are somewhat likely to still have high confidence scores. For example, a two character word can sometimes be poorly segmented and have the pieces somewhat resemble other characters. This is less likely to happen with long words since very different words are unlikely to have such ambiguous segmentation points. Words which differ by only a character might be recognized incorrectly, but still contribute mostly correct character prototype information due to the other characters which were recognized correctly.

As a result, poorly recognized short words can have a detrimental effect on the the adaptation process. When adapting to an incorrectly recognized word, incorrect prototypes are introduced to the global prototype set. This can have the effect of a worsening recognition rate after their addition.

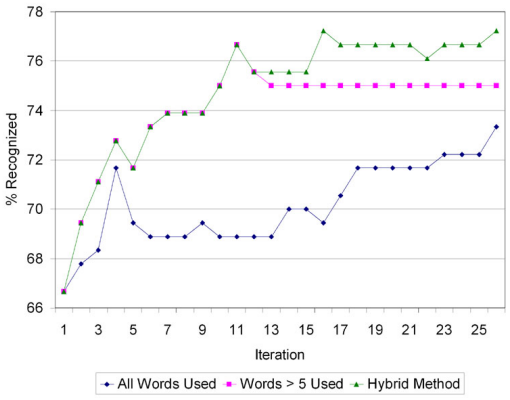
We experimentally verified (see section 5) that this phenomenon was taking place. To adjust for it, we tried two strategies. Our first strategy was to not adapt to words if the recognized words had fewer than five characters. This resulted in adapting to fewer incorrect samples being used in adaptation. However, since there were fewer samples available for training, this meant that the adaptation had to stop at an earlier point than using all words in the document. Our second strategy was to adapt to longer words first, and adapt to shorter words when the initial adaptation stage had stabilized. This second strategy resulted in fewer incorrect samples being used at the beginning of adaptation while still having all recognized words in the document available for adaptation.

4.3 Starting prototypes

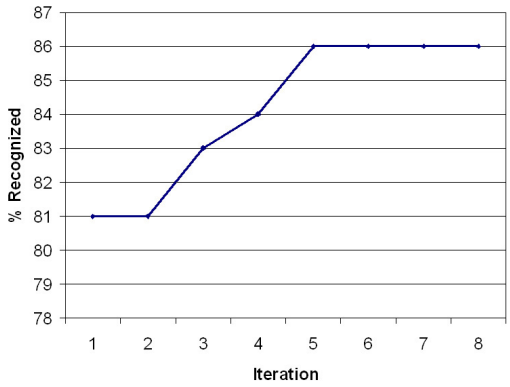
The writer-independent model's prototypes were initialized by clustering the prototype character features using K-means into a set number of clusters. The number of initial clusters as well as how the new prototype features are added affects performance. Another major factor in the performance of adaptation is the method by which the features from the newly learned prototype characters are combined with the original writer-independent prototypes. There are three broad strategies of combining the newly learned characters with the writer-independent models–adding the new feature vectors to the existing cluster centroids, combining the new feature vectors in some way with the writer-independent vectors, or replacing the writer-independent vectors with the writer-dependent features. In batch mode, especially for characters that occur often, several examples of newly learned characters may be present even in a single learning pass. In other cases, or for characters that occur less frequently, multiple passes may yield several cases. These new features may in turn be averaged or clustered.

5 Experiments

For Arabic, the adaptation algorithm was tested on a database of ten documents, each one or two pages long, written by ten different authors[15, 14]. When creating the sample set, the writers were told to make no effort to write neatly. Since no part of any the documents were used in the training phase, all of the approximately 100 documents were available for the testing phase. Although this dataset is somewhat modest in size, there is a general lack of datasets available consisting of pages of Ara-



(a)



(b)

Figure 2. Results comparison over English and Arabic documents; the difference in magnitude of improvement is likely due to the difference in cardinality of the character sets, Arabic affording more room for adaptation due to the larger number of characters (a) Three methods processing a representative Arabic document. Although all methods show significant improvement, giving preference to adapting to longer words gives marked improvement over adapting purely to words recognized with the best confidence. The hybrid method of resorting to shorter length words after exhausting the supply of longer words shows the best performance, (b) Overall results on an English document.

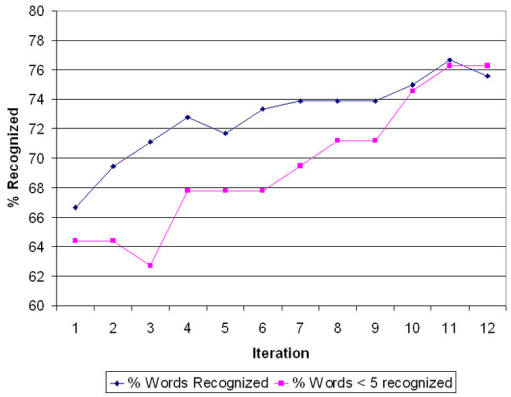


Figure 3. Adapting to words with lengths greater than or equal to five characters achieves a high increase of performance with few words used. The set of longer words itself experiences a larger percentage increase in recognition than the overall set of words in the document.

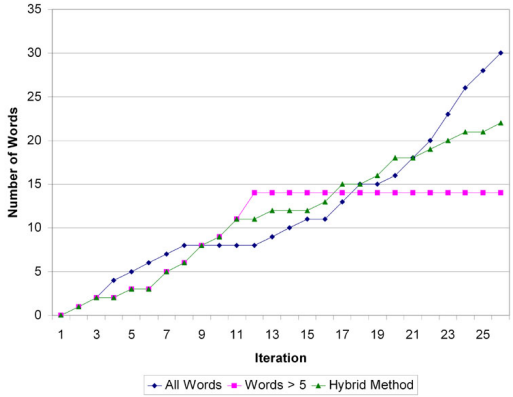


Figure 4. While adapting using the three methods, the hybrid and longer word methods achieve use fewer incorrect words for adaptation. The hybrid method achieves a greater recognition increase than the general method while adapting to fewer incorrect words.

bic text. After an initial recognition pass, the recognition rate was about 67%. After the (unrestricted) adaptation was finished, the recognition rate had improved to 73.3%, about a 6.3% improvement with 170 words used for adaptation, when the maximal correct score had been reached. About 35 words were used for adaptation incorrectly. When using only words with lengths greater than or equal to five characters for adaptation, adaptation stabilized after adapting to about 60 words, which resulted in a document recognition rate of 75%. The recognition rate for the set of words greater than or equal to five characters itself was initially 64.4% and grew to 76.3%. Only about 14 words were adapted to incorrectly. After adapting to as few as about 20 words, with only about two incorrect words used for adaptation, document recognition had already grown to about 73%. When using the hybrid approach, the maximum recognition rate was achieved after adapting to 130 words, about 22 of them incorrectly. The recognition rate was 77.2%, greater than either of the other two approaches. A representative document was processed and the results of the various methods shown in Figure 2(a) and 5.

For English, the adaptation algorithm was tested on a subset of the CEDAR letter dataset, truthed by the transcript mapping described in [7]. Increase in English performance is shown in Figure 2(b). One interesting result was that the magnitude of improvement seen over Arabic documents was larger than the English documents, likely due to the larger number of characters in Arabic. In either case, the interesting feature is not the raw performance, but the magnitude improvement seen with the prototype integration.

5.1 Starting prototypes

We ran all experiments for a varying number of starting prototypes ranging from one to 20 centroids generated by the K-means algorithm. The performance peaked at about five starting prototypes, with significantly less performance increase observed as the number of prototypes continued to increase. The rapid increase in performance was found while increasing from one to five prototypes is likely a reflection of the incorporation of multiple common styles for writing each character. The slow decrease in performance increase based on adaptation as the number of prototypes continued to increase is likely due to the less weight the newly learned information holds. In other words, when there are 20 prototypes available for a given letter, reflecting many different variations of styles for a given letter, the model is so broad that adding yet another prototype carries little weight. Utilizing alternate strategies for incorporating the features yielded less of an effect on performance, especially when an equivalent number of writer-dependent prototypes were present in the models (i.e., when the replacement method had five learned prototypes vs. the addition method had five learned prototypes). Clustering the learned features allowed the system to realize some incorrectly recognized words were in fact recognized incorrectly; however, work is preliminary as to how much of an effect rebuilding the models with these new results might have on overall recognition.

6 Conclusions and Future Directions

Writer adaptation is useful in both English and Arabic when we have several words written by the same author,

for example in the case where a page of text was written. Unfortunately for Arabic recognition, the publicly available datasets currently are mostly word oriented as opposed to being of pages of text. While these databases are also necessary, research of this type depends on having larger samples written by a single individual. We have shown that adaptation to the longer words first following by adaptation to the shorter words achieves better results than adapting to the entire set of words in a document or to only the longer words. Adaptation of even a few words illustrated a significant improvement in recognition. Adapting with as little as part of a page of text results in significant increases of recognition performance. This technique is designed with a character model based approach in mind. In our case, we cluster feature vectors for each character, but it would be interesting to extend this method to other recognition algorithms utilizing character-based models. To do so, it would be necessary to combine the existing pretrained, writer independent model with the newly discovered features at runtime.

While we have explored gradually or totally replacing prototype characters with new features extracted solely from the writer's sample, there may exist more complicated methods of combining the features with the writer-independent models that yield better results. Another direction to explore are alternative confidence metrics or further hybrid techniques. Furthermore, clustering learned samples allows words that were previously learned to be "unlearned" as new information becomes available casting doubt on an earlier recognition result. This helps deal with the problem of incorrectly learned words, but needs further investigation as to the exact effect this can have when reconstructing the models to remove the newly added characters. In addition to the performance increases that more complex adaptation techniques might bring, one interesting future result might be whether or not the difference in magnitude of performance increase between English and Arabic is preserved with other approaches.

Acknowledgement

This work was supported in part by the U.S. Department of Justice, Office of Justice Programs, NIJ grant 1043383-1-34210. Views expressed are those of the authors and not of the DoJ.

References

[1] G. R. Ball and S. N. Srihari, "Writer adaptation in offline Arabic handwriting", *Proc. Document Recognition and Retrieval XV, IST/SPIE Annual Symposium*, January 2008, San Jose, CA.

[2] G. R. Ball, S. N. Srihari and H. Srinivasan, "Segmentation-Based And Segmentation-Free Methods for Spotting Handwritten Arabic Words", *Proc. Int. Workshop on Frontiers of Handwriting*, October 2006, pp 53–59, La Baule, France.

[3] S.-H. Cha and S. N. Srihari, "Writer identification: Statistical analysis and dichotomizer", *Proceedings of SS&SPR 2000 WVCS - Advances in Pattem Recognition*, 2000, volume 1876, pp 123–132.

[4] S. Connell and A. Jain, "Writer adaptation for online handwriting recognition", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24:329–346, March 2002.

[5] S. D. Connell and A. K. Jain, "Writer Adaptation of Online Handwriting Models", *ICDAR '99: Proceedings of the Fifth International Conference on Document Analysis and Recognition*, 1999, pp 434–437, Washington, DC, USA. IEEE Computer Society.

[6] P. Haluptzok, M. Revow and A. Abdulkader, "Personalization of an Online Handwriting Recognition System", *Tenth International Workshop on Frontiers in Handwriting Recognition*, 2006.

[7] C. Huang and S. N. Srihari, "Mapping Transcripts to Handwritten Text", *Proc. Int. Wkshop Frontiers in Handwriting Recognition(IWFHR-11)*, October 2006, pp 15–20, La Baule, France.

[8] G. Kim and V. Govindaraju, "A lexicon driven approach to handwritten word recognition for real-time applications", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(4):366–379, 1997.

[9] A. Nosary, L. Heutte and T. Paquet, "Unsupervised writer adaptation applied to handwritten text recognition.", *Pattern Recognition*, 37(2):385–388, 2004.

[10] A. Nosary, L. Heutte, T. Paquet and Y. Lecourtier, "Defining writer's invariants to adapt the recognition task", *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, 1999, pp 765–768.

[11] A. Nosary, T. Paquet, L. Heutte and A. Bensefia, "Handwritten Text Recognition Through Writer Adaptation", *IWFHR '02: Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, 2002, pp 363–368, Washington, DC, USA. IEEE Computer Society.

[12] M. Pechwitz, S. S. Maddouri, V. Maergner, N. Ellouze and H. Amiri, "IFN/ENIT - database of handwritten Arabic words", *Proc. of CIFED'02*, 2002, pp 129–136.

[13] A. Senior and K. Nathan, "Writer Adaptation of a HMM Handwriting Recognition System", *ICASSP '97: Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)-Volume 2*, 1997, volume 2, pp 1447–1450, Washington, DC, USA. IEEE Computer Society.

[14] S. N. Srihari, H. Srinivasan, P. Babu and C. Bhole, "Handwritten Arabic Word Spotting using the CEDARABIC Document Analysis System", *Proc. Symposium on Document Image Understanding Technology (SDIUT-05)*, November 2005, pp 123–132, College Park, MD.

[15] S. N. Srihari, H. Srinivasan, P. Babu and C. Bhole, "Spotting Words in Handwritten Arabic Documents", *Proceedings SPIE*, 2006, pp 606702–1–606702–12, San Jose, CA.

[16] J. Subrahmonia, K. Nathan and M. Perrone, "Writer dependent recognition of on-line unconstrained handwriting", *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, 1996, volume 6, pp 3478–3481.

[17] A. Vinciarelli and S. Bengio, "Writer adaptation techniques in HMM based off-line cursive script recognition", *Pattern Recogn. Lett.*, 23(8):905–916, 2002.