Model Length Adaptation of an HMM based Cursive Word Recognition System

Marc-Peter Schambach
Siemens Dematic AG
78467 Konstanz, Germany
Marc-Peter.Schambach@siemens.com

Abstract

On the basis of a well accepted, HMM-based cursive script recognition system, an algorithm which automatically adapts the length of the models representing the letter writing variants is proposed. An average improvement in recognition performance of about 2.72 percent could be obtained. Two initialization methods for the algorithm have been tested, which show quite different behaviors; both prove to be useful in different application areas. To get a deeper insight into the functioning of the algorithm a method for the visualization of letter HMMs is developed. It shows the plausibility of most results, but also the limitations of the proposed method. However, these are mostly due to given restrictions of the training and recognition method of the underlying system.

1. Introduction

Hidden Markov Model-based techniques (HMM) are a standard technique for recognizing cursive handwritten words. While much attention has been paid to the estimation of letter model parameters, less effort has been spent on automatically determining the model topology. The model topology is mostly determined at the beginning by plausible assumptions that are made about the properties of writing.

Left-to-right models with a fixed number of states are a popular choice. In the recognition system this work is based on, three states are used for every letter model, reflecting the assumed complexity of a letter. Exceptions are made only for special entities like *pause* or punctuation marks, which are modelled by a single state. In this paper an algorithm is proposed which automatically adapts the length of each letter model to an optimal value. Besides the fact that the original systems had already been fine tuned to good performance, improvements could still be reached by optimizing the model length.

In order to analyse the quality of the proposed method, it's useful to have a look at the modelled letters. An easy method for visualizing left-to-right models is presented. It doesn't calculate a model image from model parameters only, but additionally uses classified training data in order to create an image from the abstract model. It proved to be a useful tool for detecting limitations of script models and the algorithms which are used for their adaptation.

In this paper, a quick overview of the recognition system is given first. An easy method for visualization of HMM letter models is presented in section 3, which is used to examine the results of the length adaptation algorithm described in section 4. Experimental results are presented in section 5, and finally a summary and outlook for further work is given.

2. System architecture

The basic handwriting recognition system consists of linear left-to-right HMMs using a semi-continuous (tied-mixture) probability modeling structure. The codebook is given by Gaussian densities with full covariance matrix. The system is characterized by the following features: Writing variants, or *allographs*, are modelled by parallel state paths, sharing an optional pause state. The topology is described in [3]; examples of the resulting "multipath" structure can be seen in Fig. 7. The codebook is computed in two steps: first, an initial codebook is calculated by unsupervised clustering, serving as basis for the training of a first recognition system. Then, a second codebook is determined by assigning one class to each state of the HMMs. After reducing this codebook by clustering of similar classes, the final recognition system is trained.

The preprocessing is similar to the method presented in [1]. The image is normalized and transformed into contour segments. Writing lines are estimated and determine 6 vertical areas. A window is moved from left to right over the image and geometrical features are extracted for the segments in the different areas. Areas and windows have a little overlap in order to smoothen the extracted feature vectors.

The system is well tested and has been in use for years in postal automation systems for cursive handwriting. The

recognition performance (forced recognition) of some configurations is listed in row "baseline" in Table 1.

3. Visualization of HMMs

Owing to their semicontinuous structure, the visualization of the HMMs is performed in two steps. Having calculated visualizations of the codebook classes, images of the model states are created by combining the class images.

Because the feature vectors represented by the codebook are calculated from the geometrical structure of the contour image, it is difficult to get a class image by retransformation. Instead, training data is used to generate the images. Each vector is classified, and the original segments inside the window are rastered into the class image, using the class probability as transparency weight. Results for numbers of system for German ZIP codes can be seen in Fig. 1. Every vertical strip corresponds to a single class. Two observations are noteworthy: Because the class images are arranged next to each other, they form, as a whole, a readable image of the modelled character set. The reason for this is the way the codebook is calculated by assigning states to classes. Second, the character "2" is only represented by two classes, because the number of classes has been reduced by clustering.

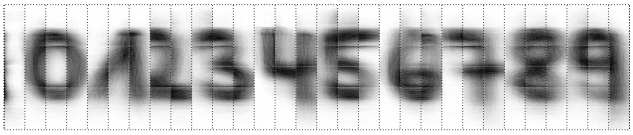

Figure 1. Visualization of classes

In the second step, the images representing the classes are combined into state images by weighting the pixel values by the class emission probabilities. The state images are ordered from left to right to build the model image. No modelling of the state width is performed. In the visualization for the German ZIP code system in Fig. 2, all models are well recognizable. (The small model on the upper left represents the "pause" model; the lower right model is the "joker" model, representing the set of all occuring letters.) The visualization can be used to control the length adaption algorithm in more difficult recognition tasks.

4. Length adaptation

Trying to find the optimal state number for each of the letter HMMs, one is faced with following problems: Under the condition that no letter boundaries are labeled in the training data, every change in model length influences the other models during training. Adapting the number of states of each model seperately also results in high computational costs. An algorithm has been developed that is able to adapt all models at once in one iteration, so only a few iterations are necessary.

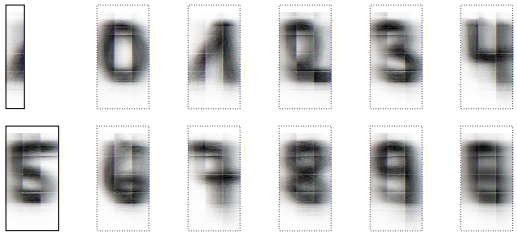

Figure 2. Visualization of character HMMs

4.1. Algorithm

The model length adaptation is done in iteratively, and all letter models are optimized at the same time. The starting point is a trained standard system with default path lengths. In every iteration, different model alternatives are generated, trained together, and finally the best is chosen. The following steps are performed:

- In case of multipath models, assign fixed writing variants to the training data and separate the models into singlepath models.
- Expand each allograph model by adding parallel state paths with ± 1 states. The result is shown in Fig. 3. The new state paths do not represent writing variants but realize alternative model lengths.
- Perform Baum-Welsh training with the expanded models.
- Select the "best" path, representing the new model.
- Retrain the HMM parameters, including a new codebook, without fixed assignment of writing variants.

Two criteria control the abort of iterations. The first criterion is that no more improvement in likelihood of training data takes place. To avoid being stuck in suboptimal configurations, the criterion is widened to stop only when additionally no further change in model topology occurs. It has been shown that better recognition results could be achieved this way. A maximum of eight iterations is performed.

4.2. Path initialization

Two initialization methods for the additional state paths have been tested. In the first, called *"mix states"*, the emission probabilities of the original states are mapped onto the

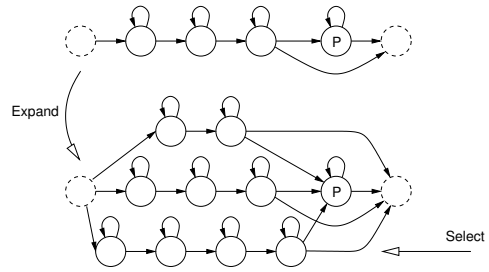

Figure 3. Learning the model length

states in the new paths. Assuming a uniform length of all states, corresponding areas represented by the states of the original and the new paths are defined.

The second initialization method, *"merge and duplicate"*, works more locally by merging similar and duplicating broad states. To shorten a path, the two adjacent states with the the smallest distance are merged. The distance $d(s_1, s_2)$ of two states is defined by the scalar product of the emission probabilities,

$$d(s_1, s_2) = \sum_c b_{s_1}(c) \cdot b_{s_2}(c). \quad (1)$$

The emission probabilities of the joined state are distributed proportional to the length of the original states. The length l_s of a state s is defined by $l_s = \frac{a_{ss}}{1-a_{ss}}$, where a_{ss} is the self transition probability of state s. A path is extended by duplicating the longest state, keeping the emission weights and adjusting the transitions to correspond to half the length.

4.3. Path selection criteria

The choice of the "best" path during iterations has great impact on the algorithm performance. A two-step mechanism has been developed. First, the state path which has been selected most frequently during training is chosen; this is indicated simply by the transition probabilities to the first states of each path. Then, after retraining the paths selected this way, the average likelihood for the new letter models is calculated. Only in cases when this likelihood is improved, is the new model length kept. It has been shown during tests that using the second step is important to gaining better recognition performance.

5. Experimental results

5.1. Project databases

The experimental data is taken from different projects in the area of postal automation. Isolated words from scanned mailpieces from Canada, Germany, and the USA have been used. In the case of German and US data, two different recognition systems (numeric and alphabetic recognizers) have been used; for Canada an alphanumeric recognizer has been trained. Additionally, data from the database of the *Center of Excellence for Document Analysis and Recognition* (CEDAR) has been used, scanned at 300 dpi and also divided into numeric and alphabetic data [2].

For the United Arab Emirates, a recognition system for the Arabic alphabet has been developed. It has been built and trained with synthetic data, but tested with real life data. Only crude adjustments of preprocessing and modelling to the properties of Arabic script have been performed. This way it seem to be a good candidate for improvements with the proposed algorithm.

Altogether, eight recognition projects have been studied. The size of the data is between 10000 and 20000 words and has been divided into training and test data in a ratio of 1:10. The recognition rates of the existing baseline systems are listed in Table 1.

5.2. Model length

The behavior of the system during iterations has been analysed. Interesting questions comprise the evolution of model length, average likelihood and recognition performance. To get a deeper insight, two different experimental sets have been used. Beside the standard recognition systems, systems with only two states per model were taken as starting point for the iterations. The aim was to test whether similar model topologies could be found in both configurations.

Analysing the model topologies generated by the algorithm, it could be noticed that the "merge and duplicate" algorithm produces longer paths than the "mix states" strategy. Starting from standard system, the average final path length was 3.46, compared to 3.27. The reason for this could be that duplicating long states results in better models than just mixing the emission probabilities. Starting from two states only, the final lengths were 2.99 (merge and duplicate) and 2.48 (mix states). However, a maximum of only eight iterations has been performed, while the average path length was still growing in all projects. So more iterations have to be performed to check whether the algorithm would converge to the same configurations.

5.3. Likelihood and recognition rate

Fig. 4 shows the evolution of likelihood and recognition rate for the Arabic system, starting from 3 states per model (left) and 2 states (right). The Arabic system has been chosen because it shows different properties of the adaptation process. Furthermore, as the base system is not fine tuned already, it has the highest potential for improvements.

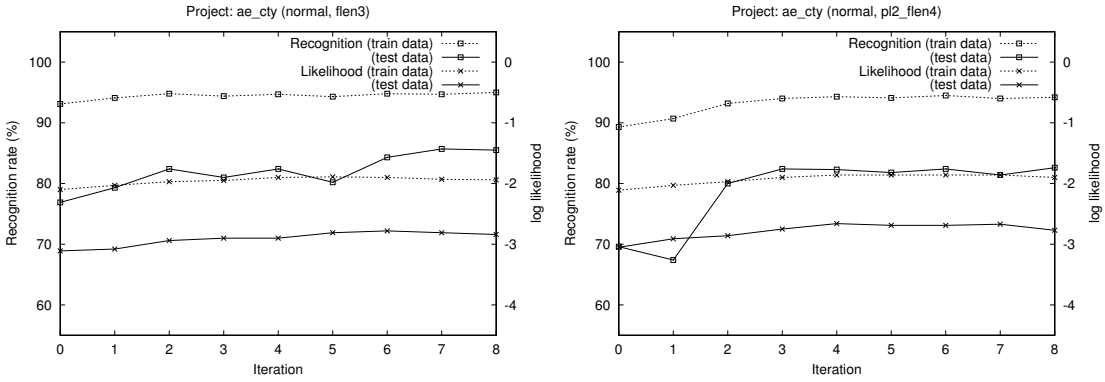

Figure 4. Plots of likelihood and recognition rate during model length adaptation iterations

- Only a weak correlation exists between likelihood and recognition rate. The recognition rate of the training data seems to be a better indicator for the performance of the recognition system.
- Likelihood and recognition rate drop locally during iterations. Therefore it is useful to continue iterations as long as changes in model topology happen.
- The generalization capability of the systems is improved during iterations.

Still, the Arabic recognition system generalizes worse in comparison to the other systems, because training data is synthetic and test data is real, but also because of the preprocessing, which does not really fit to Arabic script.

Two methods for selecting the best topology from the iterations have been tested: by likelihood, or by recognition rate of the training data. The test recognition rates are listed in Table 1. In nearly all cases, it is better to choose the topology by the training recognition rate. This way, an average improvement in recognition rate of 2.72 percent could be obtained. The highest improvements could be reached with those systems that haven't been well tuned at the start, like the Arabic system, or those that permit easy segmentation and therefore allow detailed modelling, like the ZIP code systems.

The two different initialization methods generate comparable recognition rates, besides the great impact they have in the resulting model length. Each has it advantages in some projects.

5.4. Visual inspection

The visualization of the resulting models gives a good impression of the effects of model length adaptation. The adapted numbers of the German ZIP recognition system are shown in Fig. 5. It can be seen that the characters are modelled with much more detail. The same is true for the Arabic

Figure 5. Model images after length adaptation of German numbers

models shown in Fig. 6. A selection of three characters is shown. Starting from a system with three states for each model (upper row), more details can be seen in the length-adapted models (lower row), which leads to an improved recognition rate. (The stripes inside the frames are quantization effects due to low resolution of the word images.)

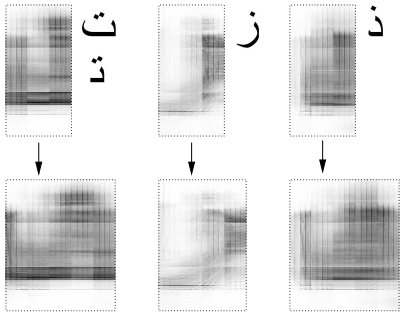

Figure 6. Length adaption of Arabic letters

The US data displayed in Fig. 7 also shows evidence of a better modelling of wide uppercase letters. The figure also gives an impression of the multipath structure of the letter HMMs.

The visualization tool proved to be very useful in the analysis of the city recognition system of CEDAR. Even

Project	baseline	mix states		merge & duplicate	
		by likelihood	by rec. rate	by likelihood	by rec. rate
Arab Emirates (emirates)	76.9 %	85.5 %	85.5 %	83.2 %	82.2 %
Canada (address)	93.4 %	93.1 %	93.4 %	92.7 %	93.4 %
Germany (address)	92.8 %	91.1 %	93.3 %	91.5 %	93.1 %
Germany (ZIP)	69.1 %	72.4 %	72.1 %	71.9 %	71.9 %
USA (address)	81.2 %	81.6 %	82.7 %	81.5 %	82.6 %
USA (ZIP)	60.8 %	63.1 %	62.9 %	63.5 %	62.9 %
CEDAR (cities)	84.2 %	83.9 %	84.2 %	82.0 %	84.1 %
CEDAR (ZIP)	58.2 %	59.3 %	59.5 %	61.8 %	61.8 %
average	77.1 %	78.8 %	79.2 %	78.5 %	79.0 %
relative improvement		+2.21 %	+2.72 %	+1.82 %	+2.46 %

Table 1. Recognition rates after length adaptation for different adapatation methods

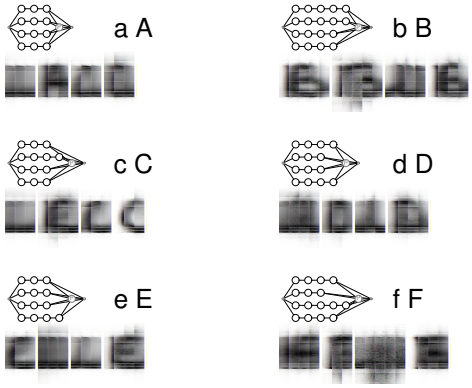

Figure 7. Model structure and images after length adaptation of US data

after length adaptation, it showed an unexpected image of the letter “u”, which can be seen in Fig. 8. The reason is that the training data is dominated by “Buffalo”, so in 62 percent of all occurrences of letter “u”, it comes in the context of “Bu”. This fault is due to the recognition concept of unsegmented training data and therefore can’t be corrected by the length adaptation algorithm.

6. Summary and outlook

In this paper, a method for visualization of HMM script recognition systems and an algorithm for determining model length of linear models has been presented. The model length adaptation showed to work well and was able to improve the recognition performance of the systems that have been studied. The average recognition improvement was about 2.72 percent. Some more work has to be done to determine the best model selection criterion, in order to realize the full potential of the underlying algorithm.

Besides its simplified calculation, the visualization tool proved to be very useful for analysing the output of model adaptation algorithms. It also facilitated the choice of the best model selection strategy in the proposed model selection algorithm. The output of the final model length selection could be justified by the complexity of the generated model images. On the other hand, it showed the limitations of the given prerequisites of unsegmented training data.

Acknowledgments

This work has been partially funded by the BMBF - German Federal Ministry of Education and Research (project Adaptive READ).

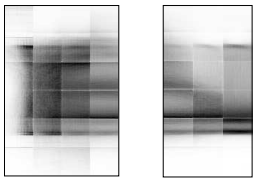

Figure 8. Visualizations of models “B” and “u” in the CEDAR city system

References

[1] T. Caesar, J. Gloger, and E. Mandler. Preprocessing and feature extraction for a handwriting recognition system. In *Proc. of the 2nd Int. Conf. on document analysis and recognition*, pages 408–411, Tsukuba Science City, Japan, Oct. 1993. IEEE Computer Society Press.

[2] J. J. Hull. A database for handwriting recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, May 1994.

[3] A. Kaltenmeier, T. Caesar, J. Gloger, and E. Mandler. Sophisticated topology of hidden Markov models for cursive script recognition. In *Proc. of the 2nd Int. Conf. on document analysis and recognition*, pages 139–142, Tsukuba Science City, Japan, Oct. 1993. IEEE Computer Society Press.