International Workshop on Frontiers in Handwriting Recognition,
University of Essex, Wivenhoe Park, September 2-5, 1996,
Colchester, England

A Fast Search Technique for Large Vocabulary On-Line Handwriting Recognition

Stefan Manke, Michael Finke and Alex Waibel

University of Karlsruhe, Computer Science Department,
D-76128 Karlsruhe, Germany

Carnegie Mellon University, School of Computer Science,
Pittsburgh, PA 15213-3890, USA

Abstract

State-of-the-art on-line handwriting recognition systems should be able to handle dictionary sizes of at least 25,000 words or more to be useful for real-world applications. Using dictionaries of this size requires fast search techniques to achieve reasonable recognition times. In this paper we present a search approach yielding recognition times, which are virtually independent of the dictionary size. This approach combines a tree representation of the dictionary with efficent pruning techniques to reduce the search space without loosing much recognition performance compared to a flat exhaustive search through all words in the dictionary. The tree search with pruning is about 15 times faster than a flat search and allows us to run the **NPen**$^{++}$ on-line handwriting recognition system in real-time with dictionary sizes up to 100,000 words.

1. INTRODUCTION

During the last decade several systems with promising recognition performance for on-line handwriting recognition have been proposed [1, 2]. These systems differ in many ways: the kind of input (hand-printed, cursive, mixed handwriting), preprocessing (e.g. temporal feature sequences vs. bitmaps), and recognition techniques (separate vs. integrated recognition and segmentation). Systems making intensive use of temporal writing information and using integrated recognition and segmentation usually outperform other approaches.

In order to achieve high recognition performance most systems constrain their search space to a fixed set of words (dictionary) that can be recognized (i.e. it is not possible to recognize arbitrary sequences of characters, but only sequences which are listed in the dictionary). The size of such a dictionary influences both the recognition performance and response time of the system and therefore also the degree of user acceptance. While

even with dictionary sizes up to 100,000 words reasonable good recognition performance can be achieved [1], response time of the system becomes an important problem. With dictionary sizes of 20,000 words or more an exhaustive search through the dictionary is no longer possible, if the system must come up with the answer in real-time. For the exhaustive search approach the search costs increase linearly with the dictionary size. Therefore a search technique which reduces the search space drastically is needed.

In this paper we present the search component of **NPen**$^{++}$, whose run-time is virtually independent of the dictionary size. This postprocessing component combines a tree representation of the dictionary with efficient pruning techniques for reducing the search space without loosing much recognition performance compared to other exhaustive search techniques (i.e. without pruning). The following section gives a short description of the **NPen**$^{++}$ system. Section 3 presents the search tree architecture and search techniques used in **NPen**$^{++}$. Recognition times and results for different dictionary sizes are shown in Section 4.

2. THE NPen^{++} SYSTEM

NPen^{++} [1] is a system for writer independent, large vocabulary on-line cursive handwriting recognition. The neural network architecture, which was orginally proposed for continuous speech recognition tasks [5], and the preprocessing techniques of **NPen**$^{++}$ are designed to make heavy use of the dynamic writing information, i.e. the temporal sequence of data points recorded on a LCD tablet or digitizer. During preprocessing this sequence of data points is transformed into a still temporal sequence of n-dimensional feature vectors $\boldsymbol{x}_0^L = \boldsymbol{x}_0 \ldots \boldsymbol{x}_L$, which combine local information (e.g. writing direction or curvature) for a data point with so-called context bitmaps [3], which are basically low resolution, bitmap-like descriptions of the coordinate's proximity.

The **NPen**$^{++}$ recognition component integrates recognition and segmentation of words into a single network architecture, the so-called Multi-State Time Delay Neural Network (MS-TDNN). This architecture combines the high accuracy pattern recognition capabilities of a TDNN [6, 4] with a postprocessing algorithm (tree search) for non-linear time alignment of stroke, character and word boundaries in handwritten words or sentences. This tree search architecture is described in the following sections.

3. TREE SEARCH

Let $W = \{w_1, \ldots w_K\}$ be a dictionary consisting of K words. Each of these words w_i is represented as a sequence of characters $w_i \equiv c_{i_1} c_{i_2} \ldots c_{i_k}$ where each character c_j itself is modelled by a three state hidden markov model $c_j \equiv q_{j_0} q_{j_1} q_{j_2}$. Within these models only self-loops $q_{j_k} \rightarrow q_{j_k}$ or state transitions $q_{j_k} \rightarrow q_{j_{k+1}}$ are allowed. The self-loop probabilities $p(q_{i_j}|q_{i_j})$ and the transition probabilities are both defined to be $\frac{1}{2}$. The idea of using three states per character is to model explicitly the initial, middle and final section of the characters. Thus, w_i is modelled by a sequence of states $w_i \equiv q_{i_0} q_{i_1} \ldots q_{j_{3k}}$.

3.1 ISOLATED WORD RECOGNITION

Given an unknown pen trajectory $\boldsymbol{x}_0^L = \boldsymbol{x}_0 \ldots \boldsymbol{x}_L$ we have to find the word $w_i \in W$ in the dictionary that maximizes the a-posteriori probability $p(w_i|\boldsymbol{x}_0^L, \theta)$ given a fixed set of parameters θ, i.e.:

$$w_i = \text{argmax}_{w_j \in W} p(w_j|\boldsymbol{x}_0^L, \theta).$$

The problem of modeling the word posterior probability $p(w_i|\boldsymbol{x}_0^L, \theta)$ is simplified by using Bayes' rule which expresses that probability as

$$p(w_i|\boldsymbol{x}_0^L, \theta) \quad = \quad \frac{p(\boldsymbol{x}_0^L|w_i, \theta)P(w_i|\theta)}{p(\boldsymbol{x}_0^L|\theta)}.$$

Thus, instead of referring to $p(w_i|\boldsymbol{x}_0^L, \theta)$ directly the MS-TDNN is supposed to model the log-likelihood $\log p(\boldsymbol{x}_0^L|w_i, \theta)$ of the pen trajectory using the following approximation [1]:

$$\log p(\boldsymbol{x}_0^L|w_i, \theta) \quad = \quad \max_{q_0^L} \sum_{l=1}^{L} \log p(\boldsymbol{x}_l, q_l|q_{l-1}, w_i)$$

The maximization is done over all possible sequences through the states of the word model. To find that maximum over all state sequences given $\boldsymbol{x}_0^L$ we define the quantity

$$\begin{aligned} s_{i_j}(l+1) &= \max_{q_0^l, q_{i_j}} \log p(\boldsymbol{x}_0^{l+1}, q_0^l, q_{i_j}|w_i) \\ &= \max_k \{s_k(l) + \log p(q_{i_j}|q_k)\} + \log p(\boldsymbol{x}_{l+1}|q_{i_j}, w_i) \end{aligned} \tag{1}$$

which is the likelihood of observing $\boldsymbol{x}_0^l$ going through the states q_0^{l-1} and ending in state q_{i_j}. Thus, the likelihood of the word w_i is given by $s_{i_{3k}}(L)$. In order to find the most likely word written we have to find this approximated likelihood for each word in the dictionary and select the highest scoring word. Figure 1 a) shows which kind of model is evaluated in order to find that word.

The problem with this kind of *flat dictionary* MS-TDNN approach is that it works fine as long as the number words in the dictionary is small (like in digit or very limited dictionary tasks). But since the run-time for a recognition pass scales linearly with the number of words in the dictionary, this approach is not feasible to build a very large dictionary neural network recognizer. Figure 2 gives an impression of how many three state HMMs we have to evaluate (i.e. run the Viterbi algorithm delineated above) given the flat structure in order to find the best matching word in the dictionary.

3.2 SEARCH TREE

Instead of having this flat organization of the dictionary where each word is represented as a sequence of 3-state hidden markov models we extended the MS-TDNN approach to a tree-based TDNN. For each letter a search tree is built which represents all words starting with this character. The nodes in each tree consist of HMMs representing individual letters (see figure 1b). Thus, according to figure 2 we can reduce the ratio of the number of HMMs to the size of the dictionary from 10 for the flat structure to 3. E.g. for the 100k dictionary system there were only 277382 models left to be evaluated compared to 968005 different HMMs in the flat search MS-TDNN approach.

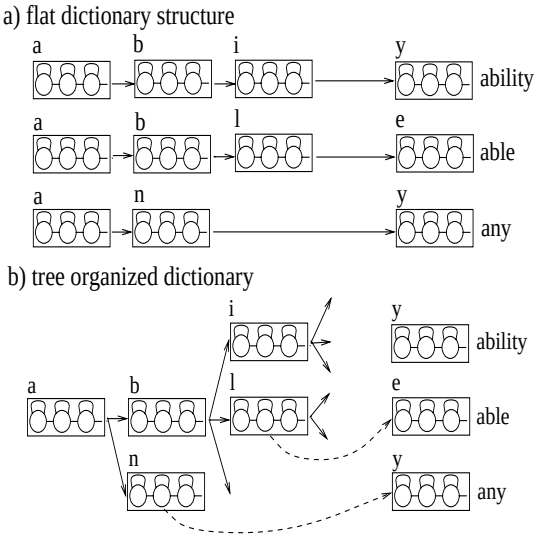

Figure 1: Flat vs. tree structured dictionary.

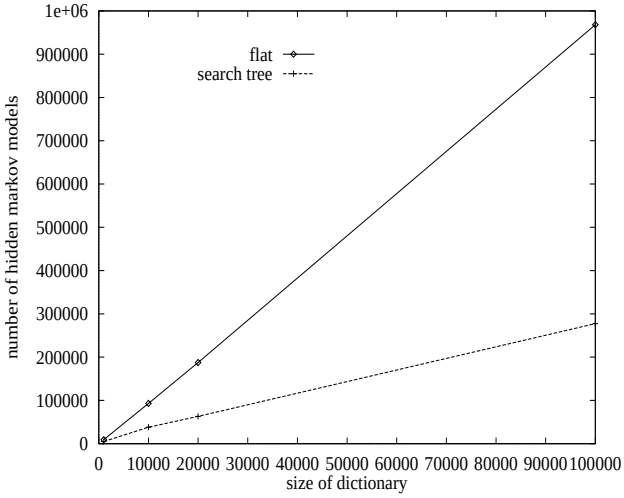

Figure 2: Number of hidden markov models to be evaluated versus dictionary size.

3.3 SEARCH ENGINE

The tree structure in itself does not yield enough of a benefit with respect to run-time efficiency. There is still a linear scaling of run-time with the dictionary size. In order to achieve real-time performance for very large dictionary systems we had to give up applying exact viterbi where for each frame each state in all HMMs in the trees is updated according to equation (1). Instead of that we introduced the concept of active and inactive HMMs and defined a set of pruning rules which specify when to turn on an inactive HMM and when to turn off an active one.

The **NPen**$^{++}$ search engine is based on the following data structures and algorithms: each HMM-node in the tree can be marked as being either active or inactive. When the search is initialized only the roots of the trees will be turned on whereas all other nodes are set to be inactive. There are two lists whose elements are pointing to active nodes, one is pointing to the nodes active in the current frame and the other one is used to gather pointers to those nodes which are supposed to be active in the next frame. Based on these lists the search algorithm goes for each frame through the following four steps:

1. **Evaluation**: for each active hidden markov model a viterbi step (1) is computed to find the accumulated scores s_{i_j} for the next frame. We also compute the best state score $\hat{s}_i$ within each node and the best score $\hat{s} = \max \hat{s}_i$ over all evaluated models.

2. **Pruning**: deactivate (turn off) all currently active nodes in the search tree where the following pruning criterion is fulfilled

   $$\hat{s}_i \quad < \quad \hat{s} - \text{beam}.$$

   That means, all nodes whose best accumulated score is below a certain threshold (called beam) will become inactive in the next frame.

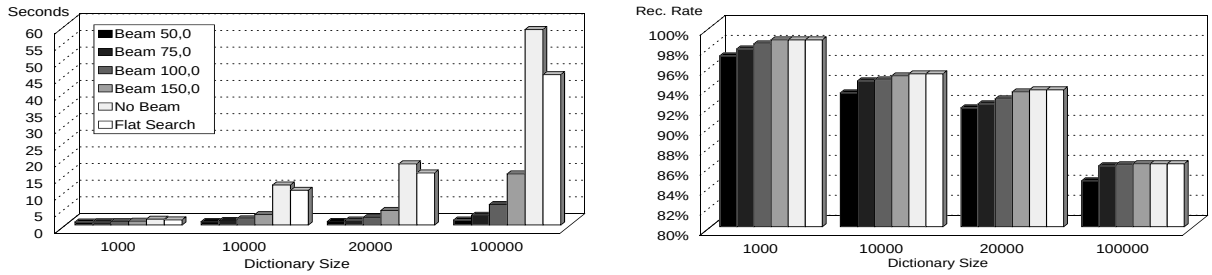

Figure 3: Recognition times (left) and recognition results (right) for different beam sizes and dictionary sizes from 1,000 words up to 100,000 words.

3. **Expansion**: for each node being active in the current frame test whether a transition from the last state of the model i to the first state of any child HMM j leads to a lower accumulated score s_{j_0} in the first state of that model. If that holds and the new score is above the pruning threshold the HMM j is marked to be active in the next frame.

4. **Word Transition**: For each active word end node we test the transition from the node to any of the tree roots as we did in the expansion phase above. This is done only for the continuous handwriting recognition task which is out of the scope of this paper.

4. EXPERIMENTS AND RESULTS

We have chosen an isolated word recognition task to demonstrate the behaviour of our tree search approach for different beams and dictionary sizes and to show its superiority compared to a flat search approach. The **NPen**$^{++}$ system has been trained on 5,700 patterns from 80 different writers. Testing of the system was performed on 2,500 patterns from an independent set of 40 writers. The dictionaries used in our experiments were selected randomly from the ARPA Wall Street Journal task, which was originally defined for continuous speech recognition evaluations.

For each dictionary we have measured the total run-time needed for testing all 2,500 patterns using different beam sizes on a standard 120Mhz Pentium PC running Linux. In figure 4. the average recognition time in seconds for each pattern and the total recognition performance (word accuracy) is shown. Recognition times in figure 4. include an average of 0.84 seconds for preprocessing and the forward pass through the front-end TDNN for each pattern.

As can be seen from figure 4. the recognition results for our tree search without using any beam (no pruning) are the same as the results for the flat search, but the recognition time is slightly higher, because of the additional costs of the tree search. Using a beam size of 150.0 for the tree search results in nearly the same recognition accuracy as without a beam but is already much faster than the flat search. Decreasing the beam size further to 100.0, 75.0, and 50.0 gives even faster response times, but also very small reduction

in recognition accuracy. For this task a beam size around 75.0 is a reasonable good compromise between speed and accuracy.

5. CONCLUSIONS AND FUTURE WORK

In this paper we have shown that an efficient search technique is essential for large dictionary handwriting recognition systems to be practical useful. The proposed tree search and pruning technique reduces the search space dramatically and is therefore significantly faster than search techniques, which have to process the whole dictionary to come up with the correct answer. The benefit in run-time is much higher than the small decrease in recognition accuracy.

Work is in progress to further reduce the search space by using duration modeling to limit the lifetime of nodes in the search tree. First experiments have shown that this duration modeling gives an additional 10%-20% reduction of search time without loosing any recognition performance.

References

[1] S. Manke and M. Finke, and A. Waibel, "NPen++: A Writer Independent, Large Vocabulary On-Line Cursive Handwriting Recognition System", *Proceedings of the International Conference on Document Analysis and Recognition*, Montreal, 1995.

[2] M. Schenkel, I. Guyon, and D. Henderson, "On-Line Cursive Script Recognition Using Time Delay Neural Networks and Hidden Markow Models", *Proceedings of the ICASSP-94*, Adelaide, April 1994.

[3] S. Manke, M. Finke, and A. Waibel, "Combining Bitmaps with Dynamic Writing Information for On-Line Handwriting Recognition", *Proceedings of the ICPR-94*, Jerusalem, October 1994.

[4] I. Guyon, P. Albrecht, Y. Le Cun, W. Denker, and W. Hubbard, "Design of a Neural Network Character Recognizer for a Touch Terminal", *Pattern Recognition*, 24(2), 1991.

[5] P. Haffner and A. Waibel, "Multi-State Time Delay Neural Networks for Continuous Speech Recognition", *Advances in Neural Network Information Processing Systems (NIPS-4)*, Morgan Kaufman, 1992.

[6] A. Waibel, T. Hanazawa, G. Hinton, K. Shiano, and K. Lang, "Phoneme Recognition using Time-Delay Neural Networks", *IEEE Transactions on Acoustics, Speech and Signal Processing*, March 1989.