Fast Two–Level HMM Decoding Algorithm for Large Vocabulary Handwriting Recognition

Alessandro L. Koerich, Robert Sabourin & Ching Y. Suen

Pontifical Catholic University of Paraná (PUCPR), Brazil
École de Technologie Supérieure, Université du Québec, Canada
CENPARMI, Concordia University, Canada

9th International Workshop on Frontiers in Handwriting Recognition, Tokyo, Japan
October 2004
Outline

• Motivation & Challenge
• Background on LVHR
• Goal
• Methodology
• Handwriting Recognition System
• Fast Two-Level HMM Decoding Algorithm
• Experimental Results
• Summary, Conclusion & Future Work
Motivation

• A baseline off-line handwritten recognition system developed by A. El-Yacoubi in 1998 at the SRTP had the following performance:

  100-word vocabulary
  - Recognition rate: 95.89% (4,481 out of 4,674 words)
  - Speed: 2 sec/word

  30,000-word vocabulary
  - Recognition rate: 73.70% (3,445 out of 4,674 words)
  - Speed: 8.2 min/word
  26 days for the whole test set !!!!
Large Vocabulary Handwriting Recognition (LVHR)

- Most of the research in handwriting recognition has focused on relatively simple problems.
  - Less than 100 classes
    - digits (10 classes)
    - characters (26 to 52 classes)
    - words (up to 100 words)

- To pass from few classes to a large number of classes (> 1,000) is a real challenge.
Large Vocabulary Handwriting Recognition (LVHR)

• Most of the classification algorithms currently used in handwriting recognition are not suitable for large number of classes.

• Few large datasets to allow training and performance evaluation.

• Few results have been reported in literature.
Large Vocabulary Handwriting Recognition (LVHR)

Accuracy

Complexity

handprinted writer-dependent small vocabulary

cursive writer-dependent small vocabulary

unconstrained writer-dependent small vocabulary

handprinted writer-independent small vocabulary

cursive writer-independent small vocabulary

unconstrained writer-independent small vocabulary

unconstrained omnewriter open vocabulary

unconstrained omnewriter large vocabulary
Current Methods for LVHR [speed]

• Lexicon pruning (prior to the recognition)
  – Application environment
  – Word length and shape

• Organization of the search space
  – Lexical tree x Flat lexicon

• Search strategy
  – Viterbi beam search
  – A*
  – Multi-pass

Most of these methods are not very efficient or/and they introduce errors which affect the recognition accuracy.
Current Methods for LVHR [accuracy]

- Improvements in accuracy are associated with:
  - Feature set
  - Modeling of reference patterns
  - More than one model for each character class
  - Combination of different feature sets / classifiers

The complexity of the recognition process has been steadily increasing with the recognition accuracy.
Challenge

• We have to account for two aspects that are in **mutual conflict**: recognition speed and recognition accuracy!

• Is it possible to overcome the accuracy and speed problems to make large vocabulary off-line handwriting recognition feasible?
Challenge

• It is relatively easy to improve the recognition speed while trading away some accuracy.

• But it is much harder to improve the recognition speed while preserving (or even improving) the original accuracy.
Goal

• To address the problems related to accuracy and speed
• Build an off-line handwritten word recognition system which has the following characteristics:
  – Omniwriter (writer independent)
  – Very-large vocabulary (80,000 words)
  – Unconstrained handwriting (cursive, handprinted, mixed)
  – Acceptable recognition accuracy
  – Acceptable recognition speed
Methodology

- Build a lexicon-driven LV handwritten word recognition system based on HMMs to generate a list of $N$-best word hypotheses as well as the segmentation of such word hypotheses into characters.

- **Problem**: Current decoding algorithms are not efficient to deal with large vocabularies.

- **Solution**: Speedup the recognition process using a novel decoding strategy that reduces the repeated computation and preserves the recognition accuracy.
Methodology

- The idea is to take into account particular aspects of the handwriting recognition system:
  - Architecture of the hidden Markov models (characters).
  - Feature extraction and segmentation (perceptual features).
  - Lexicon-driven approach.
Handwriting Recognition System

• Segmentation-recognition approach

• Lexicon-driven approach where character HMMs are concatenated to build up words according to the lexicon

• Global recognition approach to account for unconstrained handwriting
Handwriting Recognition System

Front-end Parameterization
- Pre-Processing
- Segmentation
- Feature Extraction

Recognition
- Lexicon
- Decoding

Training
- Character Models
- Word Labels

N-Best Word Hypotheses
- CAUBEYRES: -22.350
- COUBEYRAC: -23.063
- CAMPAGNAC: **-23.787**
- COURPIGNAC: -24.028
- CAMBAYRAC: -24.093
- COMPREGNAC: -24.097
- CAMPAGNAN: -24.553
Conventional Approach

- **Given:**
  - An input word
  - A lexicon with \( V \) words
  - Character HMMs (a-z, A-Z, 0-9, symbols)

1. Extract features from the input word.
2. Build up word HMM for a word in the lexicon.
3. Align the sequence of features (observation sequence) with the word HMM.
4. Decode the word HMM (estimate a confidence score).
5. Repeat Step 2 until all words in the lexicon are decoded.
6. Select those words which provide the highest confidence scores.
Conventional Approach

Character HMMs

Lexicon
Conventional Approach

\[ P(O|w) \text{ or } P("Es-sCu" | "BYE") \]
Conventional Approach (Shortcomings)

• We have observed that there is a great number of repeated computation during the decoding of words in the lexicon.

• The current algorithms decode an observation sequence in a time-synchronous fashion.

• The probability scores of a character within a word depends on the probability scores of the immediate preceding character.
Character HMMs
Fast Two-Level HMM Decoding Algorithm

• Main ideas:
  - Avoid repeated computation of state sequences
  - Reusability of character likelihoods
  - Context independent (lexicon)
During the recognition is it possible to decode the character “a” only once since it is always represented by the same character model?
Fast Two–Level HMM Decoding Algorithm

• To solve this problem of repeated computation a novel algorithm that breaks up the decoding of words into two levels is proposed:

– **First Level**: Character HMMs are decoded considering each possible entry and exit point in the trellis and the results are stored into arrays.

– **Second Level**: Words from the lexicon are decoded but reusing the results of first level. Only character boundaries are decoded.
FTLDA: First Level

• The idea is to avoid repeated computation
• We evaluate the matching between $O$ and each $\lambda$
• Assume that each $\lambda$ has a single initial state (entry) and final state (exit).
• Compute best state sequences between initial state and final state considering a single beginning frame ($b$) at time and all possible ending frames ($e$)
• Store in an array best state sequences and probabilities of all pairs of beginning and ending frames $P_A(b,e)$
FTL HMM Decoding Algorithm: First Level

- We end up with arrays of best state sequences and probabilities for each character HMM
- They are independent of the context (position within the word)

\[ b_1, b_2, b_3, b_4, b_5, b_6, b_7 \]

\[ \text{e-decoded characters to decode any word} \]

\[ \text{P(1,3) P(1,4) P(1,5) P(1,6) P(1,7)} \]
\[ \text{P(2,4) P(2,5) P(2,6) P(2,7) P(2,8)} \]
\[ \text{P(3,5) P(3,6) P(3,7) P(3,8) P(3,9)} \]
\[ \text{P(4,6) P(4,7) P(4,8) P(4,9) P(4,10)} \]
\[ \text{P(5,7) P(5,8) P(5,9) P(5,10) P(5,11)} \]

\[ e_1, e_2, e_3, e_4, e_5, e_6, e_7 \]
FTLDA: Second Level

• The idea is to use the “pre-decoded” characters.
• The problem now is to find the word in the lexicon that best matches with the $O$.
• Words are formed by the concatenation of single character HMMs.
• Words are decoded from left to right.
• Words have well-defined initial ($b=1$) and terminations ($e=T$).
FTL HMM Decoding Algorithm: Second Level

We do not need to decode HMM states which is the most time consuming computation during the decoding ($N^2T$)

\[
\hat{\delta}_t(l) = \max_{1 \leq b \leq T} \left( \hat{\delta}_b(l-1) \chi(b,t) \right)
\]
Recognition System Based on HMMs

- The output of the Word Recognition System Based on HMMs is a list with the $N$–best word hypotheses, the segmentation of such word hypotheses into characters and likelihoods.

<table>
<thead>
<tr>
<th>Label</th>
<th>Segmentation</th>
<th>Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>NERON</td>
<td>21112</td>
<td>-8.522033</td>
</tr>
<tr>
<td>VERON</td>
<td>21112</td>
<td>-8.530042</td>
</tr>
<tr>
<td>HERON</td>
<td>21112</td>
<td>-9.009810</td>
</tr>
<tr>
<td>MERON</td>
<td>21112</td>
<td>-9.354612</td>
</tr>
<tr>
<td>GERTON</td>
<td>211012</td>
<td>-9.356061</td>
</tr>
<tr>
<td>SERON</td>
<td>21112</td>
<td>-9.358621</td>
</tr>
<tr>
<td>CERON</td>
<td>21112</td>
<td>-9.432840</td>
</tr>
<tr>
<td>VERTON</td>
<td>211012</td>
<td>-9.570351</td>
</tr>
<tr>
<td>FERON</td>
<td>21112</td>
<td>-9.573650</td>
</tr>
<tr>
<td>USSON</td>
<td>21112</td>
<td>-9.583147</td>
</tr>
</tbody>
</table>
Experiments

- 70 HMMs (a-z, A-Z, 0-9, symbols)
- Global Lexicon: 85,092 city names
- Test dataset: 4,674 unconstrained words (city names)
- Platform: AMD Athlon 1.1GHz running Linux

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of Samples</th>
<th>Word Average Length</th>
<th>Number of Different Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>12,023</td>
<td>10.69</td>
<td>4,814</td>
</tr>
<tr>
<td>Validation</td>
<td>3,475</td>
<td>11.64</td>
<td>1,392</td>
</tr>
<tr>
<td>Test</td>
<td>4,674</td>
<td>11.14</td>
<td>2,540</td>
</tr>
</tbody>
</table>
Performance on the Test Dataset

<table>
<thead>
<tr>
<th>Lexicon Size (#)</th>
<th>Recognition Rate (%)</th>
<th>Recognition Time (sec/word)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FTLD LexTree</td>
</tr>
<tr>
<td>10</td>
<td>98.84</td>
<td>0.010</td>
</tr>
<tr>
<td>1,000</td>
<td>91.01</td>
<td>0.273</td>
</tr>
<tr>
<td>10,000</td>
<td>81.06</td>
<td>1.992</td>
</tr>
<tr>
<td>40,000</td>
<td>73.23</td>
<td>7.516</td>
</tr>
<tr>
<td>80,000</td>
<td>68.65</td>
<td><strong>14.46</strong></td>
</tr>
</tbody>
</table>

15 times faster . . .
Summary of Speed Improvements

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Computational Complexity</th>
<th>Storage Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTLDA</td>
<td>$T^2N^2M + T^2LV$</td>
<td>$2T^2N + 2MT^2$</td>
</tr>
<tr>
<td>Viterbi</td>
<td>$TN^2LV$</td>
<td>$2TN^2LV$</td>
</tr>
</tbody>
</table>

$T$: length of the observation sequence (30), $N$: number of HMM states (10), $M$: number character HMMs (70) $L$: average word length (11), $V$: lexicon size (80,000)

- $V$ is the dominant variable
- $T$ is quadratic but its magnitude is low

The FTLDA is advantageous while $T < N^2$
Conclusion

• We have built an omniwriter off-line handwritten word recognition system that deals efficiently with large and very-large vocabularies, unconstrained handwriting styles, and runs on personal computers with an acceptable performance.

BEFORE

80,000–word lexicon
• Accuracy: 68.65%
• Speed: 3.6 min/word

NOW

80,000–word lexicon
• Accuracy: 68.65%
• Speed: 14.46 sec/word
## Summary

<table>
<thead>
<tr>
<th>Problem</th>
<th>• How to improve the recognition speed while preserving the recognition rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idea</td>
<td>• To avoid the repeated computation of the probabilities of the same character HMMs given an observation sequence</td>
</tr>
<tr>
<td>Solution</td>
<td>• To breakup the computation of characters and words ➔ <strong>Fast Two-Level HMM Decoding Algorithm</strong></td>
</tr>
<tr>
<td>Results</td>
<td>• Speedup of the recognition process (up to 15x) while maintaining exactly the same recognition rate.</td>
</tr>
</tbody>
</table>
Conclusions and Implications

• The Fast Two–Level HMM Decoding Algorithm has speeded up significantly the recognition process without affecting the accuracy.

• Space and Time Tradeoff

• Modularity and Reusability

• The results obtained are still far from meeting the throughput requirements of many applications.

• The proposed algorithm can be naturally mapped to parallel processing (SMP and clusters).

• **Shortcoming:** It is not a general approach
Future Work

• Use of heuristics into the fast two-level HMM decoding algorithm to further speedup the recognition process.