Using Segmentation Constraints in an Implicit Segmentation Scheme for Online Word Recognition

Émilie CAILLAULT – LASL, ULCO/University of Littoral (Calais)
Christian VIARD-GAUDIN – IRCCyN, University of Nantes
Aim

• Problem:
  – Training of a neuro-markovian system in the context of cursive handwritten word recognition.

• Difficulty:
  – Bootstrapping of the system allowing a training from scratch.

• Proposed solution:
  Implicit segmentation procedure
  • Definition of a segmentation control function
  • Definition of a quality segmentation measurement
How train a neuro-markovian system?

Character level training

- Teaching signal: character
- Classifier
- HMM

Signal

Word level training

- Teaching signal: word
- Classifier
- HMM

Différents hybrides

- MMC-MMC
- RN-MMC
- MMC - RN
- SVM-MMC
- Kppv-MMC

RN-MMC
Parole/Hors-ligne
Global system overview: MS-TDNN

- Scanning of the input word by a TDNN (temporal convolution).
- TDNN produces the sequence of observation probabilities.
- No explicit segmentation is carried out but just an overlapping window is regularly shifted.
- Advantages of the TDNN architecture: weight sharing ⇒ reduction in computation times and in memory requirements.
- Reduction of the number of computation with an adapted topology of the TDNN due to the sliding observation window.
Dynamic Programming Likelihood computation

- Hence, the likelihood for each word in the lexicon is computed by multiplying the observation probabilities over the best path through the graph using the Viterbi algorithm.

- The word model with the highest likelihood is the top one recognition candidate.

For each entry in the lexicon, a pseudo-HMM-word model is constructed dynamically by concatenating letter HMM...
Training:

- Word level training
- Classical backpropagation algorithm
- Cost function mixing MLE, MMI and character level discrimination [ICDAR 2005]

« one » recognized as « over »

True Hmm: (+)    ooonnnneee

Best Hmm: (−)    oooovveeer

Best character: (−)  aouvneel
Illustration of the segmentation problem

- Random initialisation of the TDNN
  ⇒ all outputs are equally probable

Example: Possible paths for the English word ‘of’ with 10 observations (TDNN windows)
How to obtain desired Likelihood distribution?

**Existing solutions:** Training with a character database
- duration model for HMM sub-unit [Penacee]
- manual segmentation constraints [Remus, Npen++]

**Proposed solution:**
- no modification of the HMM structure (transitions=1)
- no training with a character database
- integration of a duration model in the emission probabilities

Segmentation paths for the word ‘of’
Weighting function $f_T$: 

$$\text{Outputs}(t,l) = \min(\text{Outputs}_{NN}(t,l) \times f_T(H);1)$$

B = First observation position for the concerned letter  
E = Last observation position for the concerned letter  
H = Height of the function $f_T$  
S = E-B = letter size
Weighting function applied: case of the word ‘of’

H = 100
How to evaluate the effectiveness of this weighting function? ASR

\[
ASR = \frac{1}{N} \sum_{e=1}^{N} \left( \sum_{i=1}^{NL_s(e)} \frac{nb\_obs(i)}{NLc(i) \times T / NL(e)} \right)
\]

ASR: Averaged homogenous Segmentation Rate

e: index of the considered sample
T: number of observations scanned by the TDNN
N: (fixed or limited) number of training samples
NL(e): number of letters in the word label
NLs(e): number of sequences of different letters
i: index of a sequence of letters from 1 to NLs
NLc(i): number of the same consecutive letter
and nb\_obs(i): number of observations of the letter i in the optimal Viterbi path.

<table>
<thead>
<tr>
<th>Case: ‘of’</th>
<th>NL(o)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL(f)</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ASR</td>
<td>0.36</td>
<td>0.64</td>
<td>0.84</td>
<td>0.96</td>
<td>1</td>
<td>0.96</td>
<td>0.84</td>
<td>0.64</td>
<td>0.36</td>
<td></td>
</tr>
</tbody>
</table>
Experiments:

Training in 2 stages:

1. $f_T$ segmentation constraint function for 20 epochs
2. Training with no segmentation constraint
<table>
<thead>
<tr>
<th>System</th>
<th>1S-TDNN Segmentation constraint</th>
<th>3S-TDNN Segmentation constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognition rate (%)</td>
<td>87.09</td>
<td>91.31</td>
</tr>
<tr>
<td>Relative improvement (%)</td>
<td><strong>32.6</strong></td>
<td><strong>28.1</strong></td>
</tr>
<tr>
<td>Average Segmentation Rate (ASR)</td>
<td>0.423</td>
<td>0.498</td>
</tr>
<tr>
<td>Relative improvement (%)</td>
<td>17.7</td>
<td>11.8</td>
</tr>
</tbody>
</table>

**Results on online word IRONOFF database**

IRONOFF word database
- English / French
- Lexicon: 197 words
- 2/3 Training: 20,898 words
- 1/3 TEST: 10,448 words
Conclusions

• A simple and effective process to bootstrap a hybrid system without any labeled character database

• Important increasing of the recognition rate by an initialisation stage with a duration model apply directly on the TDNN outputs
  ➢ Weighting function $f_T$

• Better quality of segmentation $\Rightarrow$ better recognition rates
  ➢ ASR – Averaged Segmentation Rate
Future works: controlled segmentation stage

- Measure of the ASR for each training iteration
- Adaptation of the parameter H in the weighting function to reduce the segmentation constraints according to the ASR value.
  - $H = f(\text{ASR})$.
  - Initialisation with $H = 100 \Rightarrow$ high segmentation constraint, only balanced segmentation paths are possible
  - Then relax constraint on H when ASR has reached a reasonable value.
Le moteur de reconnaissance : le TDNN

MLP

Global Vision

TDNN

Local vision to global vision

Weight sharing