Optimum Algorithm to Minimize Human Interactions in Sequential Computer Assisted Pattern Recognition

Jose Oncina

Dep. Lenguajes y Sistemas Informáticos
Universidad de Alicante

February 4, 2008
Outline

1. Computer Assisted Pattern Recognition
2. Formalizing
3. The Algorithm
4. Examples
5. Experiments
6. Conclusions
Nowadays computers are increasingly applied to solve challenging Pattern Recognition tasks. However, in some cases the results are not as good as desired and human supervision is mandatory. Most often this supervision is off-line: the computer produces a possible solution to the proposed task and then, the user corrects all the errors.
This framework can be described as follows:

1. The user proposes a task to solve,
2. The computer proposes a solution
3. If the user finds the solution good enough, it stops the process
4. If not, the user makes a correction
5. The computer, taking into account the correction, proposes a new solution
6. Return to step 3

As the most valuable resource is the time a user has to spend in the task, an adequate measure of performance is the expected number of corrections the user should make.
Sequential Computer Assisted Pattern Recognition

- We restrict to the case when the solution has a sequential nature.
- We assume the user always corrects the first error he finds.
- The solution prefix before the last correction can be assumed error free for the next iteration.
- Some examples of Sequential Computer Assisted Pattern Recognition tasks:
  - Human language translation
  - Speech to text transcription
  - Audio to score transcription
  - Handwriting text and music transcription
A Machine Translation Example

T: En un lugar de la Mancha, de cuyo nombre no quiero acordarme, . . .
C: In a place of La Mancha, of whose name I do not want to decide to me, . . .
U: In a village
C: In a village of La Mancha, of whose name I do not want to decide to me, . . .
U: In a village of La Mancha, the
C: In a village of La Mancha, the name of which I do not want to decide to me, . . .
U: In a village of La Mancha, the name of which I have
C: In a village of La Mancha, the name of which I have no desire to call to mind, . . .
U: . . .

- It can be observed that the correction of an error may lead to automatic correction of subsequent errors.
- Although in the first attempt there were eleven translation errors, the text was satisfactorily translated making only three corrections.
Usual approach

- At each iteration, given the task and the already known error free solution prefix . . .
- first a probabilistic representation (typically a word-graph) of all the possible solutions that begin with the error free prefix is built
- The most probable solution is proposed as guess for the next step

- We show that it is not optimum. We are going to describe the optimum
- Moreover, to find the most probable solution one has usually to face NP-Hard or undecidable problems (and then, use approximations) or to resort to complex Dynamic Programming techniques in the most simple cases.
- The proposed algorithm is a greedy technique and is free of those inconveniences.
We have

- An input string \( e \) (i.e. a sentence to translate)
- A probabilistic model \( p_e : \Sigma^* \rightarrow [0, 1] \) (i.e. a word-graph describing the probabilities of all the possible translations of the sentence)
- An evidence (i.e. the error free prefix)
- An oracle \( O : \Sigma^* \rightarrow \Sigma^* \cup \{\text{YES}\} \).
  \( O(x) \) returns:
  - \text{YES}: if \( x \) is the correct answer
  - Otherwise, The error free prefix of \( x \) concatenated with the correct next symbol
We want

- A strategy $S : \Sigma^* \rightarrow \Sigma^*$. Given a string returns a possible solution.
- Such that the process:

\[
x = \lambda \quad // \quad x \text{ is the error free prefix}
\]

repeat

\[
y = S(x) \quad // \quad y \text{ is the proposed solution using evidence } x
\]
\[
x = O(y) \quad // \quad x \text{ is the new error free prefix}
\]

until $x == \text{YES}$

- Minimizes the expected number of Oracle queries
Moving symbol by symbol

- The **Strategy** $S_s : \Sigma^* \rightarrow \Sigma$: given the evidence returns a symbol guess.
- The **Oracle**: $O_s : \Sigma^* \rightarrow \Sigma \cup \{\text{YES}\} \cup \{\text{PREF}\}$ $O_s(x)$ returns:
  1. **YES**: if $x$ is equal to the hidden string
  2. **PREF**: if $x$ is a prefix of the hidden string
  3. a symbol $a$: if $x$ is not a prefix of the hidden string, but when the last symbol of $x$ is replaced by $a$ it becomes a prefix of the hidden string (that is what we want to minimize)

- The new scheme

\[
x = \lambda
\]

```
x = \lambda

\text{loop}

\begin{align*}
a &= S_s(x) & \text{// $a$ is the proposed new symbol using evidence}\ x \\
b &= O_s(xa) & \\
\text{if}(b == \text{YES}) & \text{exit loop} & \text{// we have the correct answer} \\
\text{if}(b == \text{PREF}) & x = xa & \text{// we have guessed the next symbol} \\
\text{else} & x = xb & \text{// we made a mistake; $xb$ is the next error free prefix}
\end{align*}
```

end loop

The expected number of interactions

- Suppose the evidence is $x$
- Let $a = S_s(x)$ and $A_S(x)$ be the expected number of error corrections
- Then:

$$A_S(x) = p(a|x)A_S(xa) + \sum_{b \neq a} p(b|x) (1 + A_S(xb))$$  \hspace{1cm} (1)$$

$$= \sum_{b \neq a} p(b|x) + \sum_{b} p(b|x)A_S(xb)$$  \hspace{1cm} (2)$$

$$= (1 - p(a|x)) + \sum_{b} p(b|x)A_S(xb)$$  \hspace{1cm} (3)$$

- The second term of equation 3 does not depend on which symbol the strategy selects when considering evidence $x$
- The first term is minimized if the strategy consists in choosing the symbol $a$ that maximizes the probability $p(a|x)$
The optimum choice is the strategy

$$S(x) = xa_1 \ldots a_n \text{ where } a_i = \arg\max_{a \in \Sigma} p(a|xa_1 \ldots a_{i-1})$$

Instead of the classical one

$$S(x) = xy \text{ where } y = \arg\max_{y \in \Sigma^*} p(y|x)$$

Note that the optimum strategy can be implemented as a greedy algorithm and is much simpler to implement that the classical one.
Three strings example

Data:

<table>
<thead>
<tr>
<th>string</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa</td>
<td>0.36</td>
</tr>
<tr>
<td>ab</td>
<td>0.24</td>
</tr>
<tr>
<td>b</td>
<td>0.4</td>
</tr>
</tbody>
</table>

\[
p(a|\lambda) = \frac{p(aa) + p(ab)}{p(aa) + p(ab) + p(b)} = 0.6
\]

\[
p(b|\lambda) = \frac{p(b)}{p(aa) + p(ab) + p(b)} = 0.4
\]

\[
p(a|a) = \frac{p(aa)}{p(aa) + p(ab)} = 0.6
\]

\[
p(b|a) = \frac{p(ab)}{p(aa) + p(ab)} = 0.4
\]

Then, the expected number of mistakes is:

Classical Strategy

\[
A = 0 \cdot p(b) + 1 \cdot p(aa) + 2 \cdot p(ab)
= 0 \cdot 0.4 + 1 \cdot 0.36 + 2 \cdot 0.24
= 0.84
\]

Optimum Strategy

\[
A = 0 \cdot p(aa) + 1 \cdot p(ab) + 1 \cdot p(b)
= 0 \cdot 0.36 + 1 \cdot 0.24 + 1 \cdot 0.4
= 0.64
\]
Stochastic regular Grammar

\[
\begin{align*}
\text{Prob.} & \quad p(\lambda) = 0.1 \\
& \quad p(a) = 0.09 \\
& \quad p(aa) = 0.081 \\
& \quad \ldots
\end{align*}
\]

\[
\begin{align*}
\text{Cond. Prob.} & \quad p(\lambda) = 0.1 \\
& \quad p(a|\lambda) = 0.9 \\
& \quad p(a|a) = 0.9 \\
& \quad \ldots
\end{align*}
\]

Then, the expected number of mistakes is:

\[
A = \sum_{i=0}^{\infty} p(a^i) i = 9
\]

Classical Strategy

<table>
<thead>
<tr>
<th>Stochastic Regular Grammar</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9  [ S \rightarrow aS ]</td>
</tr>
<tr>
<td>0.1  [ S \rightarrow \lambda ]</td>
</tr>
</tbody>
</table>

Optimum Strategy

Exactly 1
<table>
<thead>
<tr>
<th>Str. length</th>
<th>Voc. size: 2</th>
<th>Voc. Size: 10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Classical</td>
<td>Optimum</td>
</tr>
<tr>
<td>10</td>
<td>4.498</td>
<td>4.363</td>
</tr>
<tr>
<td>20</td>
<td>4.443</td>
<td>4.289</td>
</tr>
<tr>
<td>30</td>
<td>4.450</td>
<td>4.211</td>
</tr>
<tr>
<td>40</td>
<td>4.452</td>
<td>4.233</td>
</tr>
<tr>
<td>50</td>
<td>4.599</td>
<td>4.245</td>
</tr>
<tr>
<td>60</td>
<td>4.546</td>
<td>4.191</td>
</tr>
<tr>
<td>70</td>
<td>4.545</td>
<td>4.170</td>
</tr>
<tr>
<td>80</td>
<td>4.432</td>
<td>4.290</td>
</tr>
<tr>
<td>90</td>
<td>4.598</td>
<td>4.255</td>
</tr>
</tbody>
</table>
Extenden Feldman Task

- Descriptions of simple **two-dimensional visual scenes** involving a few geometric objects.
- The **Context Free Grammar** that describes the language was transformed into a probabilistic one by distributing the probability mass uniformly over all the rules sharing the same left-hand non-terminal.
- Examples of sentences:
  - A large square and a large circle are far to the right of a medium dark triangle and a circle.
  - The dark circle which is below the circle and the triangle is removed.
  - A small dark circle is added far above the medium dark circle and the medium circle.

- Results:

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Corrections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical</td>
<td>8.03</td>
</tr>
<tr>
<td>Optimum</td>
<td>6.39</td>
</tr>
</tbody>
</table>
Conclusions

The problem of minimizing the number of interactions in Sequential Computer Assisted Pattern Recognition tasks has been studied.

We found that:

1. Proposing the most probable solution is not the optimum strategy in this context.
2. Building the solution by concatenating the locally most probable next symbol yields the optimum strategy.
3. The proposed optimum algorithm can be implemented as a simple greedy algorithm.

Two different sets of experiments were performed to compare both strategies and illustrate the validity of the theoretical work.