Towards collaborative agents for automatic on-line handwriting recognition

Lambert Schomaker (NICI)
Eduard Hoenkamp (NICI)
Marshall Mayberry (Univ. of Texas, Austin)

NICI, Nijmegen Institute for Cognition and Information
University of Nijmegen, P.O.Box 9104
6500 HE Nijmegen, The Netherlands
Tel: +31 24 3616029 / Fax: +31 24 3616066
schomaker@nici.kun.nl
projects in the Cognitive Engineering group at NICI:

- On-line recognition approaches
- Comparison of forensic handwriting systems
- UNIPEN
- Multimodal speech and handwriting input
- Information Retrieval/Information Filtering
- Content-based image retrieval
- Hybrid (NN/AI) modeling
• Multi-level information integration

• Agents: old for new?

• A triple-agent system
multi-level information integration
"context"

• Use of context: a panacea for limited bottom-up classification performance?

• It is difficult to realize efficient use of context:
  – in case of complex input
    (cf. OCR of newspaper page vs. OCR of mail envelope)
  – under dynamic and free input conditions
    (writing a letter on a pen computer)
multi-level information integration

"context"

- What? (... are the relevant context bits: the "frame" problem, Pylyshyn)

- How?

- No elegant solutions for multi-level information integration exist, as yet
how to exploit ”context”
syntax

• Earlier experiments with NLP & on-line recognition: disappointing

• Parser for Dutch, using sentences from office context

• Batch architecture
  (strokes → characters → words → sentence)
  – use of context postponed until last word of sentence .
  – was slow!
  – written input may be syntactically incorrect

  – writers don’t write job applications or love letters in this way
how to exploit "context"
syntax, continued

Needed: interactive approach
(e.g., incremental parser)

- probabilistic language models
  (works: but large corpus needed, many parameters)

- grammars
  (concise & explicit: but may lack information)

How to make a system which is modular and dynamically configurable?
the multiple-agent approach

old wine in new bottles?

• **O.G. Selfridge (1958)**

• **Daemons**

• **Critics gallery**

• **Multiple experts**

• **Society of mind**
the multiple-agent approach

what’s new?

• Good definitions (Wooldridge & Jennings)

• Game theory, negotiation algorithms

• Multi-sensor fusion algorithms

• Learning
  – genetic algorithms
  – case-based reasoning

• Formalisms: Knowledge Interchange Format (KIF), Knowledge Query and Manipulation Language (KQML)

• Try: http://ontolingua.nici.kun.nl
Potential for pattern recognition:

- Realisation of complex decision boundaries again: the double spiral argument

- Solve geometrically, e.g., with a MLP?  
  → Overfit!

- Solve algorithmically, by search?  
  → More powerful!
a triple-agent system

experiment:

design a system

• simple

• interactive (user is present & time is real)

• using bottom-up and top-down information

• using agent architecture

→ in order to see what the use of syntactic information may yield under simplified conditions
a triple-agent system

design issues

• no natural language input but Scheme program input on a pen computer

• interactive:
  – no machine font substitution, leave ink ’as is’
  – use color for state feedback
  – give user full control, using virtual buttons, menus etc.

• bottom-up: Kohonen LVQ classifier of unistrokes

• top-down: Scheme parser (LR, incremental)
a triple-agent system

Scheme code example (towers of Hanoi)

hanoi.scm

(define ringlist
  (lambda (l n)
    (define mring
      (lambda (size)
        (cons 'ring size)))
    (if (= n 0)
      l
      (ringlist (cons (mring n) l) (- n 1)))))

(define mpole
  (lambda (ndisks)
    (cons 'pole (ringlist nil ndisks))))

(define disks
  (lambda (pole)
    (cdr pole)))
  .
  .
  .
  .
Agents:
1. shape classifier
2. expression classifier
3. user+user interface
a triple-agent system

Shape Classifier agent

- Input: tokens of the Scheme language, written as unistrokes
- unistrokes, resampled to 60 samples
- Kohonen LVQ, nearest centroid match
- translate to $\bar{\mu} = (0, 0)$
- normalize rms radius to $\sigma_r = 1$
- feature vector:
  - $(x_k, y_k)$ 60 normalized coordinates
  - $(\cos(\phi_k), \sin(\phi_k))$ 59 pairs
  - total $119 \times 2 = 238$
- training, 5-10 samples of a token
- learning rule $f_j = \eta x_j + (1 - \eta)f_{j-1}$
- token recognition rate $\approx 85\%$
a triple-agent system

Shape Classifier agent (pseudo code)

Init:

init-communication
read-table-with-token-templates
ask-parser-for-type-of-each-token

while(true) {
    switch (read-request()) {
        case unistroke
            classify unistroke
            query-parser
            combine-parser-expectancy-and-shape-classification
            notify-user-agent
        
        case train
            update-token-shape-and-label
            notify-user-agent
    
    }
}
a triple-agent system
Expression Classifier for Scheme

- context-free grammar
- LR parser: incremental, no look ahead
- use lex/yacc (shift/reduce)
- tokens:

  .               /=
  (               =
  )               and
  *               begin
  +               BOOLEAN
  -               case
  .               CHAR
  cond            let
  define          let*
  delay           NUMBER
  do              or
  else            set!
  if              STRING
  lambda          VAR
a triple-agent system

Expression Classifier for Scheme

• Example of rule:

state 29

\begin{align*}
\text{Def} & : \text{LPAR} \text{ DEFINE\_VAR} \text{ Expr} \text{ RPAR} \\
\text{Def} & : \text{LPAR} \text{ DEFINE\_LPAR} \text{ VAR} \text{ RPAR} \text{ Body} \text{ RPAR} \\
\text{Def} & : \text{LPAR} \text{ DEFINE\_LPAR} \text{ VAR} \text{ DefFormals} \text{ RPAR} \text{ Body} \text{ RPAR}
\end{align*}

\begin{align*}
\text{VAR} & \quad \text{shift} \ 55 \\
\text{LPAR} & \quad \text{shift} \ 56 \\
. & \quad \text{error}
\end{align*}

• After each token: generate list of expected tokens and update state

• Requests to parser agent:
  \begin{align*}
  & \text{Accept\_token} \\
  & \text{Reset\_state} \\
  & \text{Delete\_token} \\
  & \text{Forward\_token}
  \end{align*}
Expression Classifier agent (pseudo code)

Init:
  init-communication
  read-grammar

while(true) {
  switch (read-request()) {
    case token
      process-token
      update-parser-state
      return-expected-tokens
    
    case reset
      reset-parser-state
  .
  .
  }

User Interface agent (pseudo code)

init-communication
start-parser
start-classifier

create-windows
create-event->task-bindings
    (ink events, parser events, classifier events)

wait-for-events(forever)
a triple-agent system

User Interface
• VAR expected
• token *dog* written
• token *dog* rejected → must be new token!
a triple-agent system

results

good news:

- 100% ’recognition’

- users (Scheme programmers) like it!

- agent architecture is very convenient
a triple-agent system

results

bad news:

- individual information contributions by the agents must be analysed and quantified
- VAR becomes a problem in case of unconstrained scope
- NUMBER and STRING are open categories
results

Information content of Scheme source code

<table>
<thead>
<tr>
<th>Symbols</th>
<th>$N_{alphabet}$</th>
<th>$2\log(N_{alphabet})$</th>
<th>Entropy</th>
<th>Redundance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw token stream</td>
<td>2003</td>
<td>11.0</td>
<td>6.3</td>
<td>4.7</td>
</tr>
<tr>
<td>Lumped token stream</td>
<td>28</td>
<td>4.8</td>
<td>2.4</td>
<td>2.4</td>
</tr>
</tbody>
</table>

(Based on corpus of N=27310 tokens.
Lumped means: use placeholders instead of actual instances of VAR.)

Entropy: $-\sum_{i=1}^{N_{alphabet}} p_i \cdot 2\log p_i$
a triple-agent system
results

**Parser expectancy**

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Avg. $N_{alternatives}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw token stream (VAR scope=whole corpus)</td>
<td>1891.5</td>
</tr>
<tr>
<td>Raw token stream (VAR scope=single function)</td>
<td>97.4</td>
</tr>
<tr>
<td>Lumped token stream</td>
<td>16.0</td>
</tr>
</tbody>
</table>

(Scheme source-code corpus of 27310 tokens.
Lumped means: use placeholders instead of actual instances of VAR.)

→ If scope is not limited to a single function, the parser adds very little information. Reasons: users’ naming creativity and the presence of constants (string, number).
• User actions are definitely needed!

• But their work can be made easier by using syntactical context

• The virtues of a grammar:
  ”Look Ma’ - No probabilities!”

• Beware of placeholders (name slots) in the grammar

• Just a first step towards the use of a multiple-agent architecture