Do what I mean:
Teaching computers to automate repetitive tasks

Tessa Lau
IBM TJ Watson Research
4/20/04
Programming by demonstration

- People should be able to program their computers without writing code
  - If you can use it, you can program it
- Teach computer how to perform task by demonstrating actions in user interface
  - Generalize from demonstration to program
- Automate repetitive tasks
Why is PBD hard?

- Huge space of potential programs
- Very small number of examples
- Prediction must be highly accurate
  - Amounts to understanding user’s intent
- Prior approaches: domain-dependent heuristics
- My work:
  
  *Machine learning approach to PBD*
SMARTedit demo

PROBLEM #1
----------

This is some sample HTML text from which the comments including contents ought to be deleted before publication.

This comment deletion task is one example of the types of repetitive tasks for which SMARTedit saves user effort.
How SMARTedit works

- Action is function: input state → output state
  - Editor state: text buffer, cursor position, etc.
  - Actions: move, select, delete, insert, cut, copy, paste

- Given a state sequence, infer actions
  - Many actions may be consistent with one example
SMARTedit's version space

Action version space is union of different kinds of actions

Action functions map from one text state to another
SMARTedit's version space

Express action functions in terms of locations
Location functions map from text state (buffer, pos) to position

Rectangle indicates atomic (leaf) version space
SMARTedit's version space
How does the system learn?

- Update version space on new example
  - Remove inconsistent hypotheses
  - Prune away parts of the hierarchy
- Execute version space for prediction
  - Give system current state
  - What state would the user produce next?
Updating the version space

- Test consistency of example against entire version space
- Quickly prune subtrees
- Example:
Updating the version space

\[ \text{Location} \cup \text{RowColumn} \]

\[ \text{Row} \quad 1 \quad 2 \quad 3 \]
\[ \text{Col} \quad 0 \]

\[ f(x) = 0 \quad f(x) = x \quad g(x) = 0 \quad g(x) = x \]
\[ f(x) = 1 \quad f(x) = x + 1 \quad g(x) = 1 \quad g(x) = x + 1 \]
\[ f(x) = 2 \quad f(x) = x + 2 \quad g(x) = 3 \quad g(x) = x - 3 \]
\[ \ldots \quad \ldots \quad \ldots \quad \ldots \]

\[ \text{RightSearch} \]

\[ "a" \quad "b" \quad "<" \quad "<!" \quad "<!-" \quad "\ldots" \]
Executing the version space

RowCol

Row

Col

f(x)=2  f(x)=x+1  g(x)=0  g(x)=x-3

(4, 5)

(2,0), (2,2), (5,0), (5,2), (6,11)...

RightSearch

"<"

"<!"

"<!-"

"<!--"

...
Getting your priors straight

- How to choose between possible outputs?
- Associate probability w/ea hypothesis
  - Make better predictions
  - Introduce domain knowledge

- Introduce probabilities at two points in hierarchy:
  - **Probability distribution** over hypotheses at leaf nodes
  - **Weights** for each VS in a union
Experimental results

Very few examples needed!

Results indicate examples that must be demonstrated, out of total number of examples

<table>
<thead>
<tr>
<th>Scenario</th>
<th>train</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>c++comments</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>column-reordering</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>country-codes</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>modify-to-rgb-calls</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>number-fruits</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>subtype-interaction</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>xml-comment-attribute</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>addressbook</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>citation-creation</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>html-comments</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>latex-macro-swap</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>number-citations</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>number-iterations</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>smartedit-results</td>
<td>2</td>
<td>27</td>
</tr>
<tr>
<td>zipselect</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>game-score</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>html-latex</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>indent-voyagers</td>
<td>3</td>
<td>32</td>
</tr>
<tr>
<td>mark-format</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>bold-xyz</td>
<td>4</td>
<td>50</td>
</tr>
<tr>
<td>citation-to-bibtex</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>bindings</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>boldface-word</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>ul-to-dl</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>OKRA</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>outline</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>pickle-array</td>
<td>19</td>
<td>117</td>
</tr>
</tbody>
</table>
User study results

- 6 undergrad CS majors
- 7 repetitive tasks with & later w/out SMARTedit
- Tasks: 4 to 27 iterations, 1-5 min to complete
- Evaluation metrics:
  - Time saved completing task with SMARTedit's help
  - % user actions (keyboard + mouse) saved
  - User feedback
Time saved using SMARTedit

![Graph showing time savings for different tasks using SMARTedit.](graph.png)
Percentage action savings with SMARTedit
Research at an industry lab

- My history:
  - BA/BS in Computer Science & Applied Physics at Cornell University, 5/1995
  - PhD in CS at University of Washington, 5/2001
  - Research Staff Member at IBM Research, 9/2001

- Challenge: find balance between
  - Research that benefits IBM
  - Long-term scientific advances
  - Personal interest and motivation
How can IBM make use of PBD?

- Vision: programming by demonstration can change the world!
- Desk-side technical support: reduce cost of system administration
  - Troubleshooting network connectivity
  - Installing IBM software
- Automated business procedures
  - Requesting travel reimbursement
  - Buying a new workstation
- Personal productivity
Sheepdog demo
Learning from multiple traces

- Align similar steps in procedure:
  - expert #1: 
  - expert #2: 

- Build procedure model:

  WinXP? ↘
  Win2K? ↘
  Static IP? ↘
  Dynamic IP? ↘
  WinXP? ↘
  Static IP? ↘
  Dynamic IP?
What happens next?

- Finding path to impact
  - Web platform more aligned with IBM strategy than Windows platform
  - Learning about web-based server installation and configuration
- Continuing on with research
  - Adding more domain knowledge, as in SMARTedit
  - Learning actions with variable parameters
  - Defining metrics to evaluate usability and system performance