Lecture 5: Designing smart systems

Using statistical methods to anticipate user needs and actions with Bayesian strategies
Overview of the course

- Theory driven approaches to HCI
- Design of visual displays
- Goal-oriented interaction
- Designing efficient systems
- **Designing smart systems**
  - Designing meaningful systems (guest lecturer)
  - Evaluating interactive system designs
  - Designing complex systems
Uniform text entry
Information gain per key press

\[ h(x_i) = \log \frac{1}{p(x_i)} \]
The q?
Information gain per key press

“As you are aware, E is the most common letter in the English alphabet, and it predominates to so marked an extent that even in a short sentence one would expect to find it most often”

The Adventure of the Dancing Men,
Sir Arthur Conan Doyle
Hacking Fitt’s Law: “semantic pointing”

Simple application of Fitts Law

What’s wrong with this?
Increasing the depth of the language allows for a further separation...
Building a system based on relative frequencies

Dasher (https://www.youtube.com/watch?v=FLalNywdHxU)
Some lessons from Dasher

• Turning an information theoretic model into a user interface requires a lot of creativity
  => Interaction with Machine learning course

• In many cases simple models (nGrams + smoothing) are as - or more - effective than complex ones (neural nets)

• Supporting even famous software, useful for marginalised groups is hard

(The Financial Times, February 2002)
Artificial languages

new Future.
Artificial languages

Liveness becomes Entelechy, Church et al
A simple scheme for predicting code completions:

```java
void main() {
    Stopwatch sw = new Stopwatch();
    sw. // <--- What goes here?
}
```

Ordering code completion suggestions:

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>elapsed</td>
</tr>
<tr>
<td>elapsedMicroseconds</td>
</tr>
<tr>
<td>elapsedMilliseconds</td>
</tr>
<tr>
<td>elapsedTicks</td>
</tr>
<tr>
<td>Frequency</td>
</tr>
<tr>
<td>hashCode</td>
</tr>
<tr>
<td>isRunning</td>
</tr>
<tr>
<td>noSuchMethod</td>
</tr>
<tr>
<td>Reset</td>
</tr>
<tr>
<td>runtimeType</td>
</tr>
<tr>
<td>Start</td>
</tr>
<tr>
<td>Stop</td>
</tr>
<tr>
<td>toString</td>
</tr>
</tbody>
</table>
Ordering code completion suggestions

We calculate:

\[
P(\text{completion} = \text{“reset”}) \quad | \quad \text{context} = \text{“void main() { Stopwatch sw = new Stopwatch(); sw.”}}
\]

\[
P(\text{completion} = \text{“start”}) \quad | \quad \text{context} = \text{“void main() { Stopwatch sw = new Stopwatch(); sw.”}}
\]

... And the usual:

\[
P(A \mid B) = \frac{P(B \mid A) P(A)}{P(B)},
\]
Ordering code completion suggestions

\[ P(\text{completion} = ? \mid \text{context} = "...") \propto P(\text{context} = "...") \mid \text{completion} = ?) P(\text{completion} = ?) \]

<table>
<thead>
<tr>
<th>Completion c</th>
<th>Count of seen completions</th>
<th>P(completion)</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>10</td>
<td>0.5</td>
</tr>
<tr>
<td>reset</td>
<td>5</td>
<td>0.25</td>
</tr>
<tr>
<td>elapsed</td>
<td>5</td>
<td>0.25</td>
</tr>
</tbody>
</table>
Ordering code completion suggestions

\[ P(\text{completion} = c \mid \text{context} = "...") \propto P(\text{context} = "..." \mid \text{completion} = c) P(\text{completion} = c) \]

<table>
<thead>
<tr>
<th>Completion c</th>
<th>( P(\text{completion}\text{=}c \mid \text{context}) )</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>(0.9 \times 0.5 = 0.45)</td>
<td>0</td>
</tr>
<tr>
<td>reset</td>
<td>(0.4 \times 0.25 = 0.1)</td>
<td>1</td>
</tr>
<tr>
<td>elapsed</td>
<td>(0.2 \times 0.25 = 0.06)</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Completion c</th>
<th>Feature</th>
<th>Feature value</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>“First-Use”</td>
<td>true</td>
<td>9</td>
</tr>
<tr>
<td>reset</td>
<td>“First-Use”</td>
<td>false</td>
<td>1</td>
</tr>
<tr>
<td>elapsed</td>
<td>“First-Use”</td>
<td>true</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>false</td>
<td>4</td>
</tr>
</tbody>
</table>
Some progress in information efficient IDEs
Building user interfaces
(from Per Ola’s material)
Building user interfaces
(from Per Ola’s material)

- Entry and error rate
- Learning curve, familiarity and immediate efficacy
- Form factor, presentation, time and comfort
- User engagement
- Visual attention and cognitive resources
- Privacy
- Single vs Multi-character entry
- Specification vs Navigation
- One/Two handed
- Task integration
- Robustness
- Device independence
- Computational demands
- Manufacturing and support cost
- Localisation
- Market acceptance

The cross-over point

![Diagram showing the cross-over point between unfamiliar and familiar interfaces with performance, time investment, and benefit axes.](image-url)
Building user interfaces: Solution principles  
(from Per Ola’s material)

- From closed to open-loop
  - Avoid the need for a visual feedback loop
- Continuous novice-to-expert transition
  - Avoid explicit learning
- Path dependency
  - Avoid redesign the interaction layer
- Flexibility
  - Enable users to compose and edit in a variety of styles without explicit mode switching
- Efficiency
  - Let users’ creativity by the bottle-neck