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 model allows for a further optimisation, accounting not only for “pointing” target width, but distance of travel ...
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

=> Part II / Part III Human-Centred AI module

• 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)
Deploying smart interfaces
(from Per Ola Kristensson, Cambridge Professor of Interactive Systems Engineering)
Deploying smart interfaces
(from Per Ola Kristensson, Cambridge Professor of Interactive Systems Engineering)

- 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
Deploying smart interfaces
(from Per Ola Kristensson, Cambridge Professor of Interactive Systems Engineering)

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

new Future.
Ordering code completion suggestions

A simple scheme for predicting code completions:

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

```
elapsed
elapsedMicroseconds
elapsedMilliseconds
elapsedTicks
Frequency
hashCode
isRunning
noSuchMethod
Reset
runtimeType
Start
Stop
toString
```
Ordering code completion suggestions

We calculate:

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

... 

And the usual:

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

\[ P(\text{completion} = ? \mid \text{context} = \ldots) \propto P(\text{context} = \ldots \mid \text{completion} = ?) \quad P(\text{completion} = ?) \]

Feature vector

<table>
<thead>
<tr>
<th>Completion c</th>
<th>Count of seen completions</th>
<th>(P(\text{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>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>true</td>
<td>2</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
Where are we going with GPT/CoPilot?

From *Moral Codes* + joint PPIG / Lund AI meeting in 2023:

- Good: allow connection to contextual knowledge
- Poor: doesn’t support visual design notations
- Good: more efficient predictive text, ideally software reuse, cross-PL
- Poor: rife with plagiarism and IP infringement
- Good: accessible to non-programmers and learners
- Poor: hallucinations even worse than in natural language (subtle errors that look right), non-repeatable behaviour

Is CoPilot the next compiler for an even higher-level language? (Remember FORTRAN the Formula Translator).

There is lots of detailed work in our group - e.g. Michael Lee’s project for Part II HCAI module.
Real Co-pilot

https://stocksnap.io/photo/pilots-travel-OBR3AXHVMQ, CC0
Airbus autopilot

https://docs.flybywiresim.com/pilots-corner/a32nx-briefing/flight-deck/glareshield/fcu/
Airbus autopilot

Approach Speed

Page 1 of OPS DATA explains how the speed correction is applied. The approach speed increment should be added to the \( V_{\text{REF}} \) (shown as \( V_{\text{LS}} \) on the PERF APPR page) for Flap FULL. In addition, provided that the resultant \( V_{\text{APP}} \) does not exceed \( V_{\text{REF}} +20 \text{kt} \), one third of the headwind component should be added to this figure.

The resultant speed should be inserted, if possible, in the \( V_{\text{APP}} \) field on the PERF APPR page and bugged on the standby airspeed indicator. If the situation requires the speed to be Selected, rather than Managed, then the speed calculated above can be set on the FCU.

Insertion of the calculated \( V_{\text{APP}} \) on the PERF APPROACH page will ensure that if Managed speed is available, the correct approach speed will be flown. Also the benefits of GS mini will be available, even though the aircraft is landing in an abnormal configuration.

For example, a DUAL ADR FAULT requires a direct law landing flown in Config 3, using a \( V_{\text{APP}} \) of \( V_{\text{LS}} \) Flap FULL plus 10kt, plus one third of the headwind component, subject to the 20kt limit described above.
Tesla autopilot

Design considerations

- Situational awareness
- Interaction style for configuring future behaviour
- Degree of knowledge of the behaviour of the system

Some local research for hybrid human/AI futures:
Multiverse Explorer
The programming analogy challenge 2024:
Example #4: eX-Twitter (we look at Emmanuel Adesola’s GoBase)
Is “social” actually a function? In eX-Twitter, any actual work is only a side-effect of networking interactions, meaning bullshit is maximised and message value is reduced to near-zero.

Elon Musk told our PM Rishi Sunak that after these tasks have been automated by AI, nobody will need to work any more.

In Emmanuel’s GoBase, the social network delivers proof of work. How does the functional programming paradigm account for side-effects in the world outside the computer, and can the same approach be used to fix the bullshit to value ratio in the non-digital world?