

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

$$h(x_i) = \log \frac{1}{p(x_i)}$$

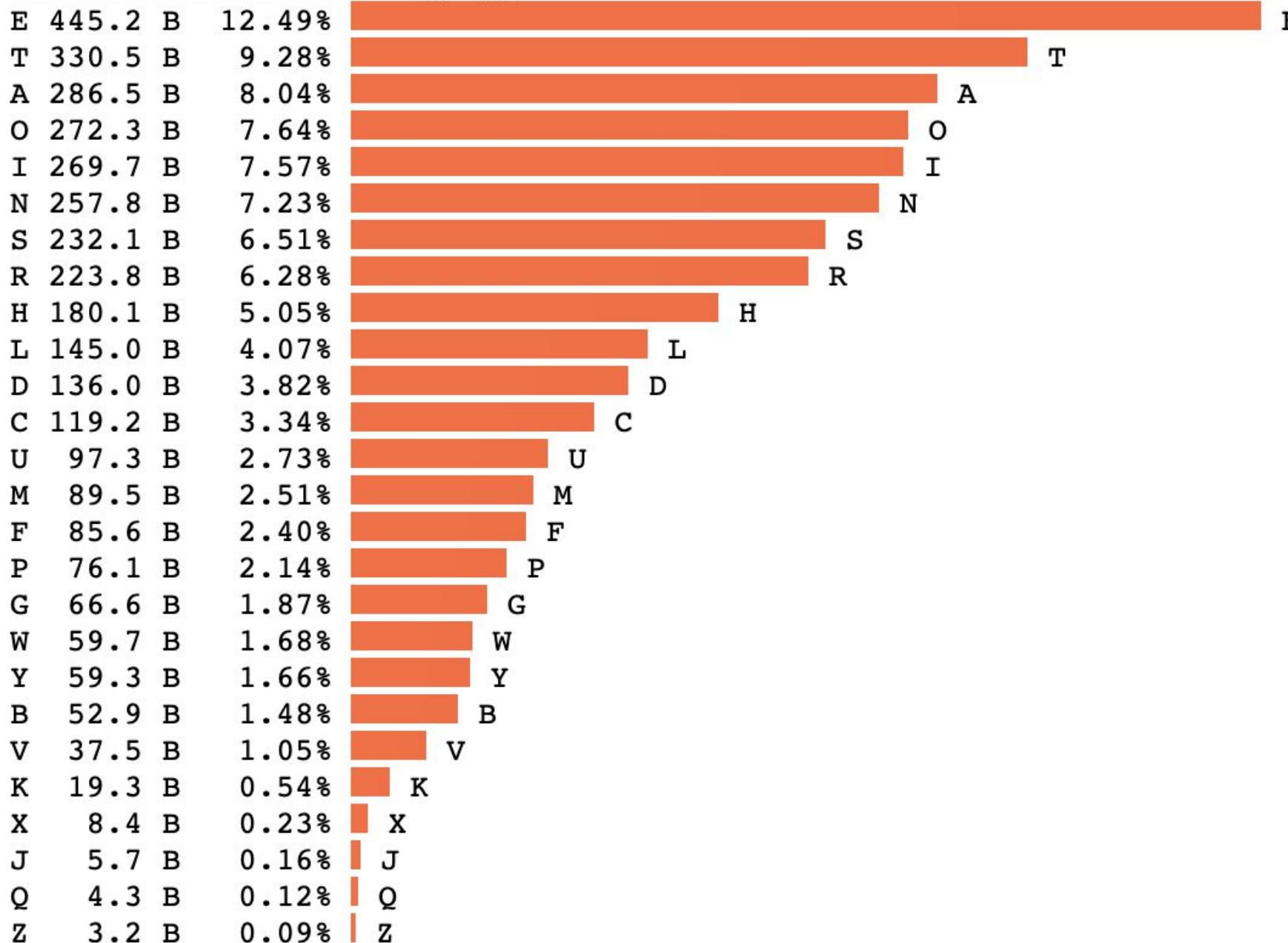
The q?

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

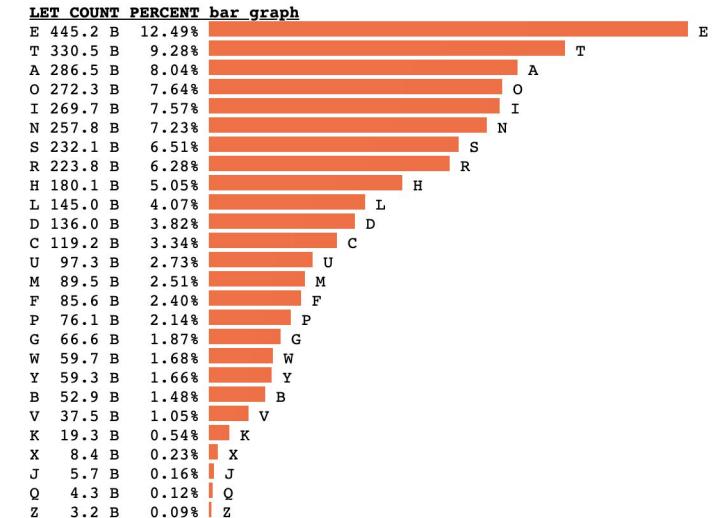
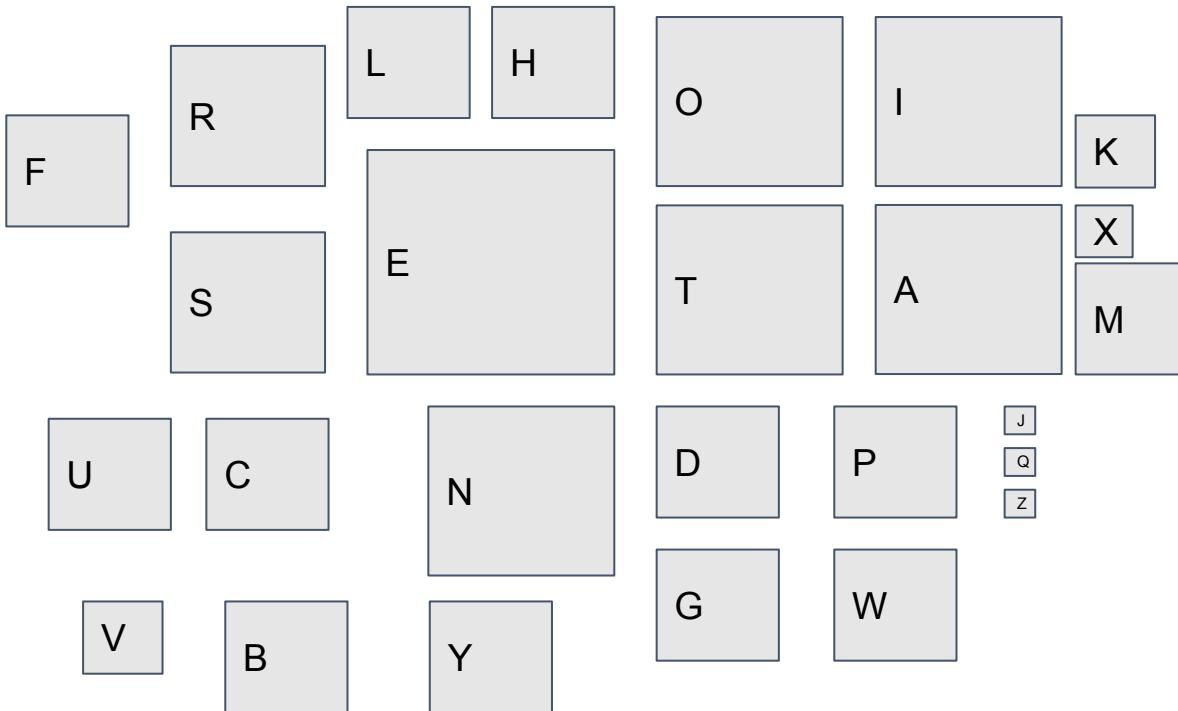
LET COUNT PERCENT bar graph



Hacking Fitt's Law: “semantic pointing”



Simple application of Fitts Law



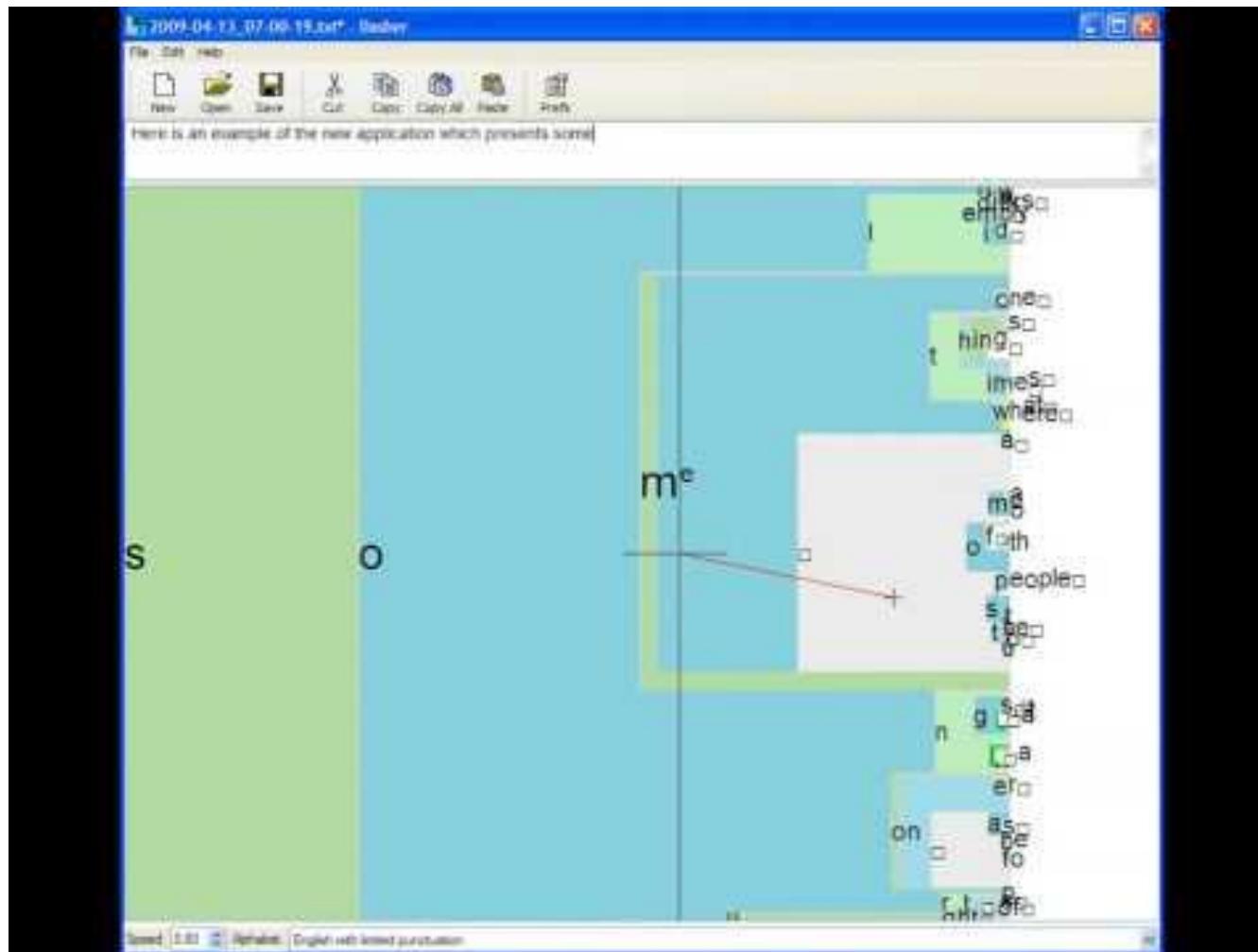
What's wrong with this?

Bigrams

AA	BA	CA	DA	EA	FA	GA	HA	IA	JA	KA	LA	MA	NA	OA	PA	QA	RA	SA	TA	UA	VA	WA	XA	YA	ZA
AB	BB	CB	DB	EB	FB	GB	HB	IB	JB	KB	LB	MB	NB	OB	PB	QB	RB	SB	TB	UB	VB	WB	XB	YB	ZB
AC	BC	CC	DC	EC	FC	GC	HC	IC	JC	KC	LC	MC	NC	OC	PC	QC	RC	SC	TC	UC	VC	WC	XC	YC	ZC
AD	BD	CD	DD	ED	FD	GD	HD	ID	JD	KD	LD	MD	ND	OD	PD	QD	RD	SD	TD	UD	VD	WD	XD	YD	ZD
AE	BE	CE	DE	EE	FE	GE	HE	IE	JE	KE	LE	ME	NE	OE	PE	QE	RE	SE	TE	UE	VE	WE	XE	YE	ZE
AF	BF	CF	DF	EF	FF	GF	HF	IF	JF	KF	LF	MF	NF	OF	PF	QF	RF	SF	TF	UF	VF	WF	XF	YF	ZF
AG	BG	CG	DG	EG	FG	GG	HG	IG	JG	KG	LG	MG	NG	OG	PG	QG	RG	SG	TG	UG	VG	WG	XG	YG	ZG
AH	BH	CH	DH	EH	FH	GH	HH	IH	JH	KH	LH	MH	NH	OH	PH	QH	RH	SH	TH	UH	VH	WH	XH	YH	ZH
AI	BI	CI	DI	EI	FI	GI	HI	II	JI	KI	LI	MI	NI	OI	PI	QI	RI	SI	TI	UI	VI	WI	XI	YI	ZI
AJ	BJ	CJ	DJ	EJ	FJ	GJ	HJ	IJ	JJ	KJ	LJ	MJ	NJ	OJ	PJ	QJ	RJ	SJ	TJ	UJ	VJ	WJ	XJ	YJ	ZJ
AK	BK	CK	DK	EK	FK	GK	HK	IK	JK	KK	LK	MK	NK	OK	PK	QK	RK	SK	TK	UK	VK	WK	XK	YK	ZK
AL	BL	CL	DL	EL	FL	GL	HL	IL	JL	KL	LL	ML	NL	OL	PL	QL	RL	SL	TL	UL	VL	WL	XL	YL	ZL
AM	BM	CM	DM	EM	FM	GM	HM	IM	JM	KM	LM	MM	NM	OM	PM	QM	RM	SM	TM	UM	VM	WM	XM	YM	ZM
AN	BN	CN	DN	EN	FN	GN	HN	IN	JN	KN	LN	MN	NN	ON	PN	QN	RN	SN	TN	UN	VN	WN	XN	YN	ZN
AO	BO	CO	DO	EO	FO	GO	HO	IO	JO	KO	LO	MO	NO	OO	PO	QO	RO	SO	TO	UO	VO	WO	XO	YO	ZO
AP	BP	CP	DP	EP	FP	GP	HP	IP	JP	KP	LP	MP	NP	OP	PP	QP	RP	SP	TP	UP	VP	WP	XP	YP	ZP
AQ	BQ	CQ	DQ	EQ	FQ	GQ	HQ	IQ	JQ	KQ	LQ	MQ	NQ	OQ	PQ	QQ	RQ	SQ	TQ	UQ	VQ	WQ	XQ	YQ	ZQ
AR	BR	CR	DR	ER	FR	GR	HR	IR	JR	KR	LR	MR	NR	OR	PR	QR	RR	SR	TR	UR	VR	WR	XR	YR	ZR
AS	BS	CS	DS	ES	FS	GS	HS	IS	JS	KS	LS	MS	NS	OS	PS	QS	RS	SS	TS	US	VS	WS	XS	YS	ZS
AT	BT	CT	DT	ET	FT	GT	HT	IT	JT	KT	LT	MT	NT	OT	PT	QT	RT	ST	TT	UT	VT	WT	XT	YT	ZT
AU	BU	CU	DU	EU	FU	GU	HU	IU	JU	KU	LU	MU	NU	OU	PU	QU	RU	SU	TU	UU	VU	WU	XU	YU	ZU
AV	BV	CV	DV	EV	FV	GV	HV	IV	JV	KV	LV	MV	NV	OV	PV	QV	RV	SV	TV	UV	VV	WV	XV	YV	ZV
AW	BW	CW	DW	EW	FW	GW	HW	IW	JW	KW	LW	MW	NW	OW	PW	QW	RW	SW	TW	UW	VW	WW	XW	YW	ZW
AX	BX	CX	DX	EX	FX	GX	HX	IX	JX	KX	LX	MX	NX	OX	PX	QX	RX	SX	TX	UX	VX	WX	XX	YX	ZX
AY	BY	CY	DY	EY	FY	GY	HY	IY	JY	KY	LY	MY	NY	OY	PY	QY	RY	SY	TY	UY	VY	WY	XY	YY	ZY
AZ	BZ	CZ	DZ	EZ	FZ	GZ	HZ	IZ	JZ	KZ	LZ	MZ	NZ	OZ	PZ	QZ	RZ	SZ	TZ	UZ	VZ	WZ	XZ	YZ	ZZ

Increasing the depth of the language allows for a further separation...

Building a system based on relative frequencies

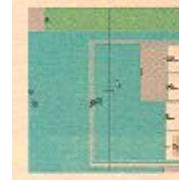


Some lessons from Dasher

- 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

It guesses your thoughts, then types

DAVID MACKAY SET OUT to invent a better way of entering text on devices such as digital assistants and mobile phones. His creation, which he calls "Dasher", is a little like an arcade game: *Attack of the Killer Alphabets*, perhaps. A reader in physics at Cambridge, he used his knowledge of probability to devise a system where the letters appear to flow – on the screen – towards the writer's pen or cursor. As the letters flood by, the shape of your word appears as if by magic, stretching out into the alphabet soup like a character in a colour blindness test. It's smart maths rather than magic: the system guesses the word you are trying to write and flows the next character towards the cursor. It also learns the kinds of words you use. Only minute movements of pen or cursor are needed, making Dasher a prime candidate for use by both able-bodied and disabled. It could, for example, be driven by a device which tracks eye movement. MacKay is a co-founder of Transversal, a commercial venture which, again, exploits probability theory to make the interrogation of computer databases simpler. MacKay believes in sharing software and Dasher is free to download from the web, much to his Transversal colleagues' horror. Get your copy before they shrink wrap it. www.inference.phy.cam.ac.uk/djw30/dasher/download.html



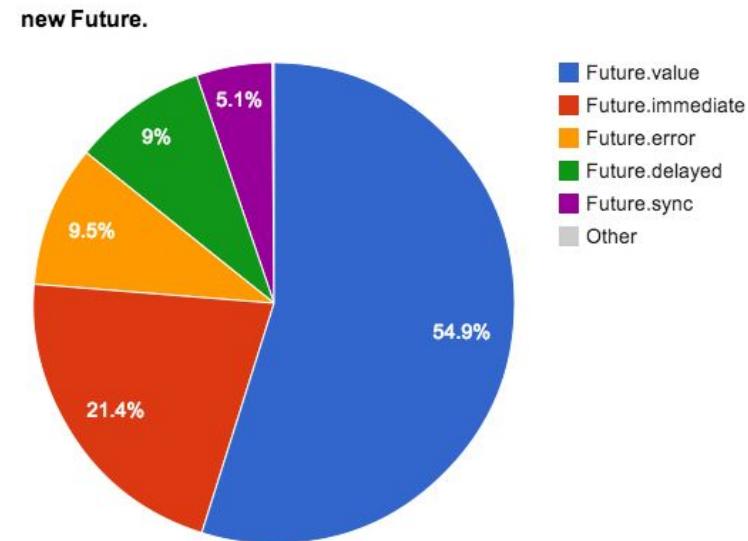
Artificial languages

new Future.?

```
New Future.aaaaa()  
New Future.aaaab()  
New Future.aaaac()  
New Future.error()  
...  
New Future.aaaad()  
New Future.aaaae()  
New Future.value()
```

Artificial languages

new Future.?



Ordering code completion suggestions

A simple scheme for predicting code completions:

```
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:

$$\begin{aligned} P(completion = "reset" \mid context = "void main() \{ Stopwatch sw = new Stopwatch(); sw.") \\ P(completion = "start" \mid context = "void main() \{ Stopwatch sw = new Stopwatch(); sw.") \end{aligned}$$

...

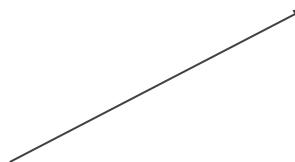
And the usual:

$$P(A \mid B) = \frac{P(B \mid A) P(A)}{P(B)},$$

Ordering code completion suggestions

$$P(completion = ? \mid context = "...") \propto P(context = ... \mid completion = ?) P(completion = ?)$$

Feature vector



Completion c	Count of seen completions	P(completion)
start	10	0.5
reset	5	0.25
elapsed	5	0.25



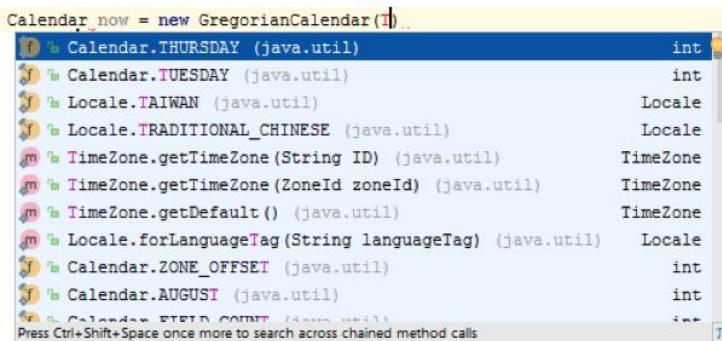
Ordering code completion suggestions

$$P(completion = c \mid context = "...") \propto P(context = "..." \mid completion = c) P(completion = c)$$

Completion c	P(completion==c context)	Order
start	$0.9 * 0.5 = 0.45$	0
reset	$0.4 * 0.25 = 0.1$	1
elapsed	$0.2 * 0.25 = 0.06$	2

Completion c	Feature	Feature value	Count
start	"First-Use"	true	9
		false	1
reset	"First-Use"	true	2
		false	3
elapsed	"First-Use"	true	1
		false	4

Some progress in information efficient IDEs



A screenshot of the JSNICE web interface. The interface has a header with "JS NICE 2020 Version - Google Drive", "TYPING, TYPE INFERENCE AND DEOBFUSCATION", and "ABOUT". The main area is divided into two sections: "ENTER JAVASCRIPT" on the left and "RESULT. ARE YOU SATISFIED?" on the right. In the "ENTER JAVASCRIPT" section, the following code is shown:

```
1 // Put your JavaScript here that you want to rename, deobfuscate,
2 // or infer types for:
3 function chunkData(e, t) {
4     var n = [];
5     var r = e.length;
6     var i = 0;
7     for (; i < r; i += t) {
8         if (i + t < r) {
9             n.push(e.substring(i, i + t));
10        } else {
11            n.push(e.substring(i, r));
12        }
13    }
14    return n;
15}
16 // You can also use some ES6 features.
17 const get = (a,b) => a.getElementById(b);
18
19
```

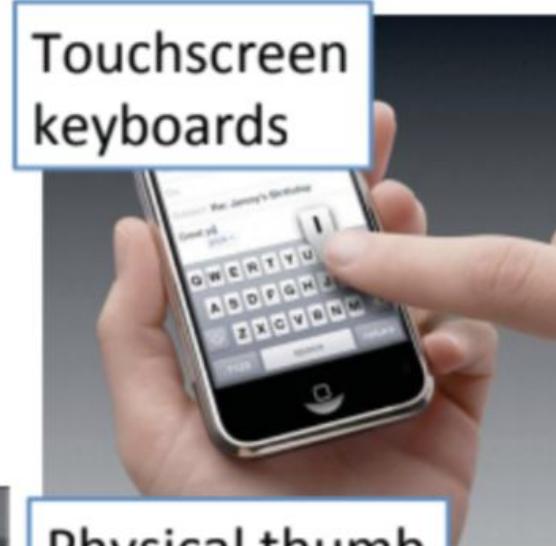
Below this, a "NICIFY JAVASCRIPT" button is visible. In the "RESULT. ARE YOU SATISFIED?" section, the deobfuscated and type-inferred code is displayed:

```
1 'use strict';
2 /**
3  * @param {string} bin
4  * @param {number} size
5  * @return {}
6 */
7 function chunkData(bin, size) {
8     /* @type {!Array} */
9     var results = [];
10    var length = bin.length;
11    var doc = (number) =>
12    var i = 0;
13    for (; i < length; i = i + size) {
14        if (i + size < length) {
15            results.push(bin.substring(i, i + size));
16        } else {
17            results.push(bin.substring(i, length));
18        }
19    }
20    return results;
21}
22 const get = (doc, key) => {
23    return doc.getElementById(key);
24};
25
```

At the bottom right, there are "YES" and "NO" buttons. At the very bottom of the JSNICE interface, there are "Tweet" and "Like" buttons with the number "287".

Building user interfaces

(from Per Ola's material)



Physical thumb keyboards

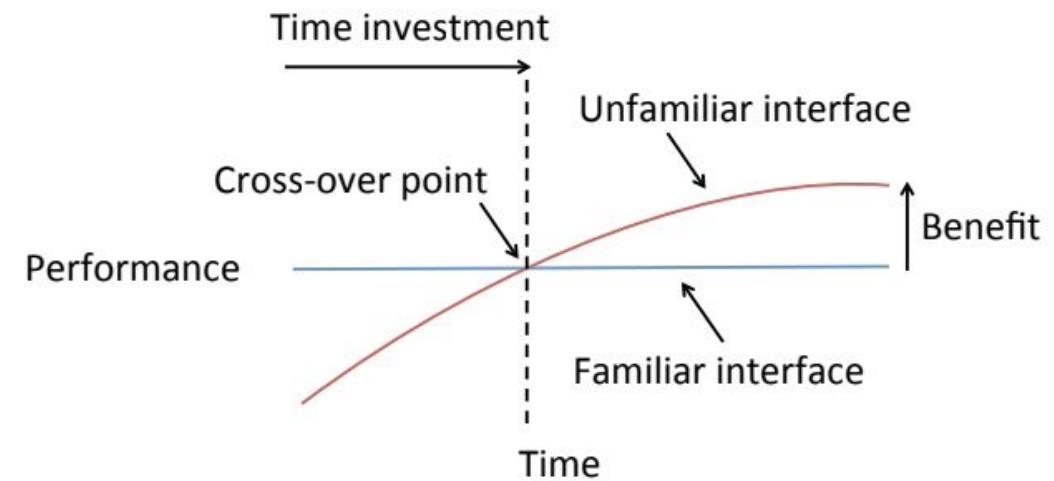


Building user interfaces

(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



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