Adaptive and Predictive Keyboards

Daniel Buschek

Learning Goals

- Challenges of mobile touch and typing

- Modelling typing behaviour

- Probabilistic methods for keyboard adaptation
  and input decoding

Motivation: Fast typing without errors

Here: mobile devices

- „Inviscid entry rate": Bottleneck is not the text entry UI but coming up with the text

- Estimated as 67 WPM

→ Try to reach this on your phone without errors, e.g. in an online typing speed test.

| Text entry method	Highest reported entry rate (wpm)
Estimate of the inviscid entry rate	*67*
Physical thumb keyboards	60 [3]
Gesture keyboards	45 [9]
Optimized on-screen keyboards	45 [12]
QWERTY on-screen keyboards	40 [12]
KALQ thumb keyboard	37 [14]
Half-QWERTY	35 [13]
Twiddler	35 [11]
WalkType	31 [5]
ContextType	28 [6]
Disambiguating keypads	26 [7]
Unconstrained handwriting recognition	25 [8]
Dasher	20 [21]
Mobile speech	18 [18]
Quikwriting	16 [15]
Unistrokes	16 [1]
TiltText	14 [22]
Multi-tap	12 [23]
Graffiti	11[1]
EdgeWrite	7 [24]
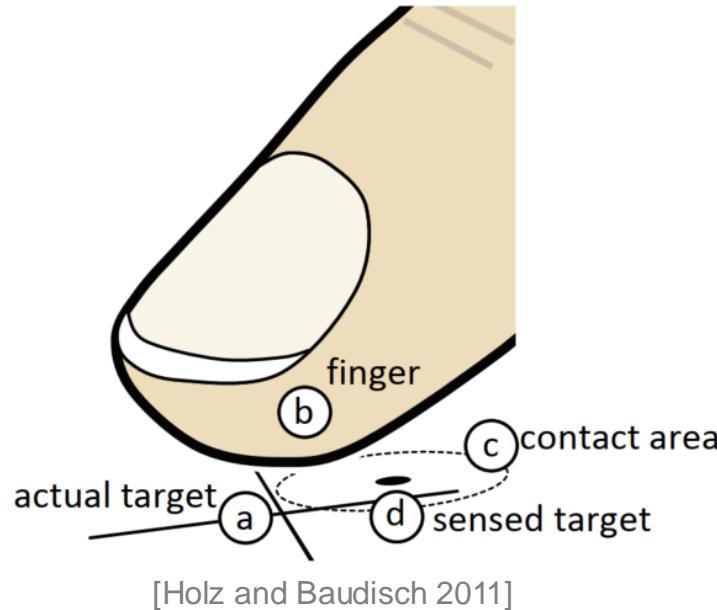
[Kristensson and Vertanen 2014]

Challenges for Mobile Typing

Why is it inaccurate?
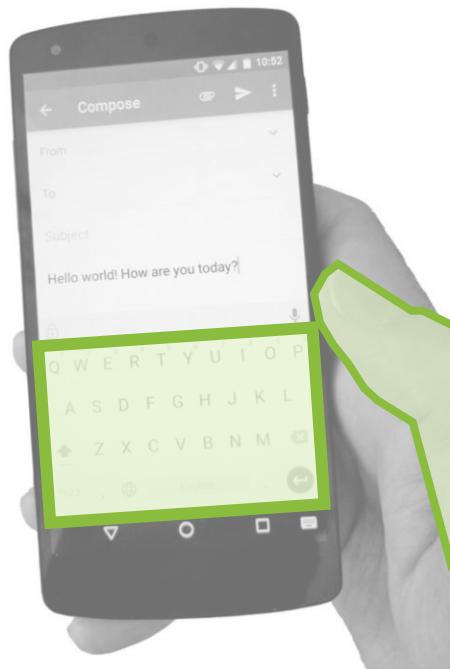
Parallax
eye – finger - screen

[Holz and Baudisch 2011]
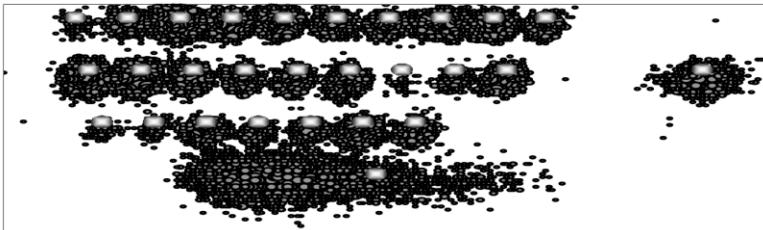
Mobile use
1-2 fingers, small keys, body movement


Hello world! How are you today?

Photo by Ketut Subiyanto

Variance in Touchscreen Keypresses
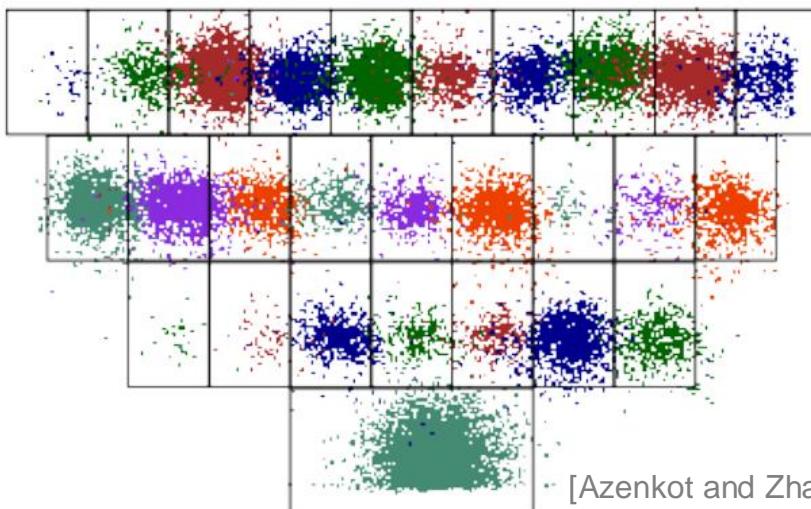
Spread of x,y touch locations around key centres

PDA
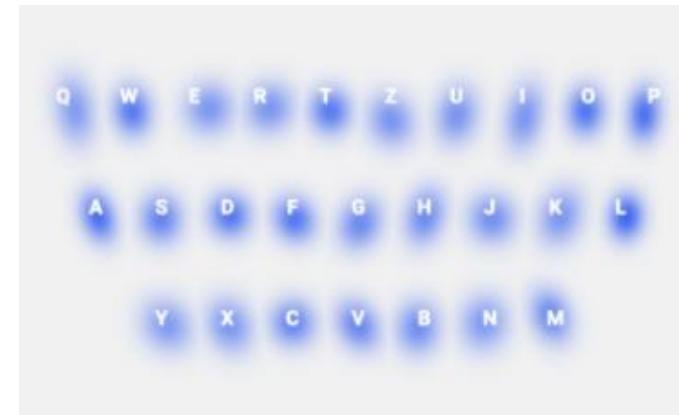
[Goodman et al. 2002]

Smartphone

[Azenkot and Zhai 2012]


https://www.microsoft.com/en-us/swiftkey
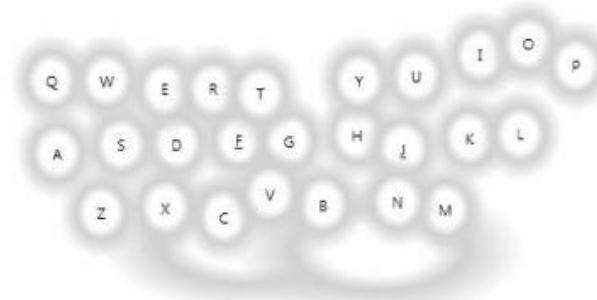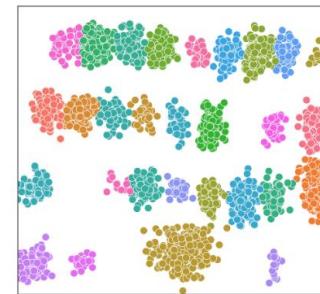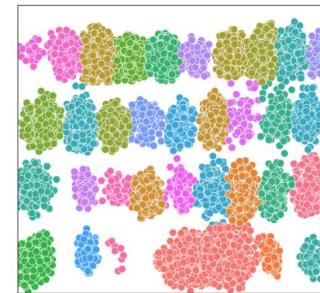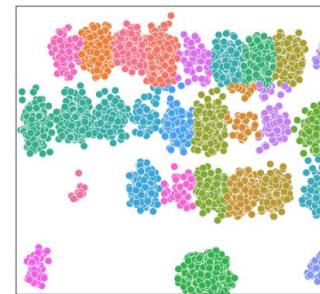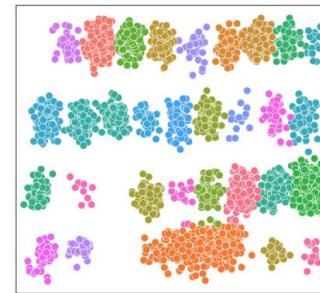
Individual Typing Behaviour

Tabletop

[Findlater and Wobbrock 2012]

Smartphone

[Buschek et al. 2018]
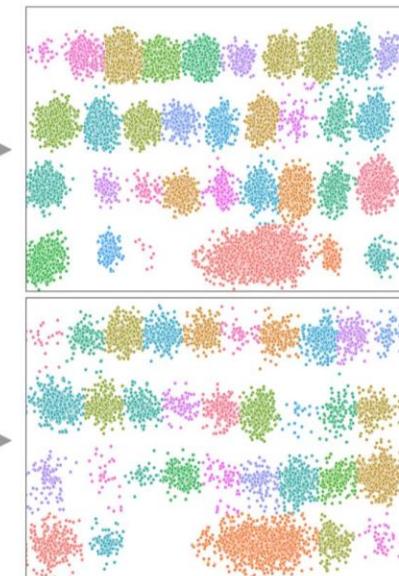
Adapting Keyboards to Typists

Overview

Visible keyboard

Collect touches

Adapt underlying key regions
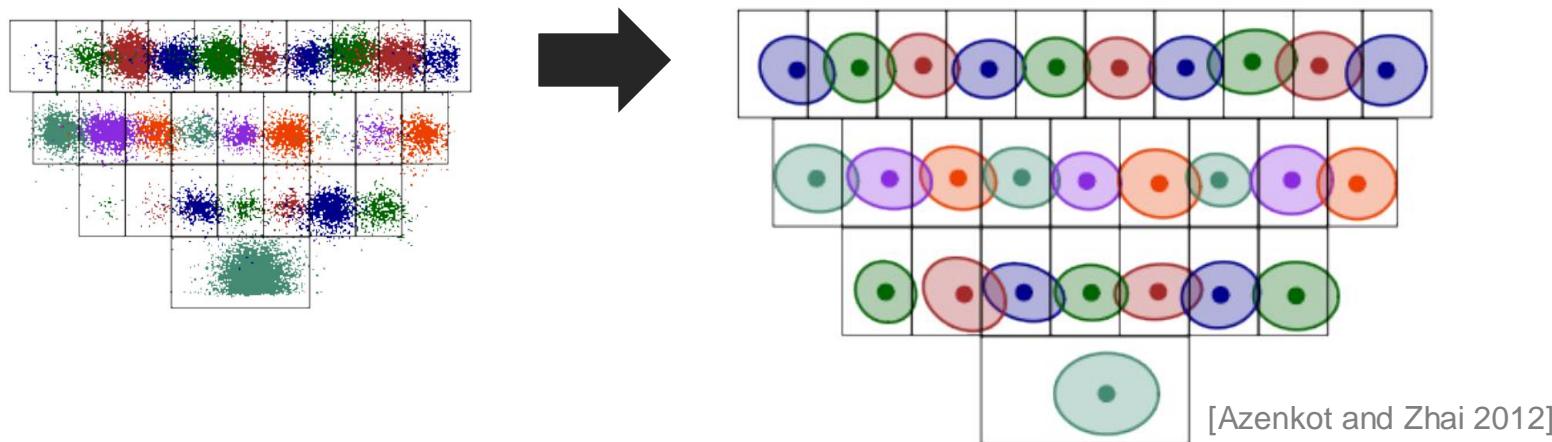
User A

User B

Modelling Touchscreen Keypresses
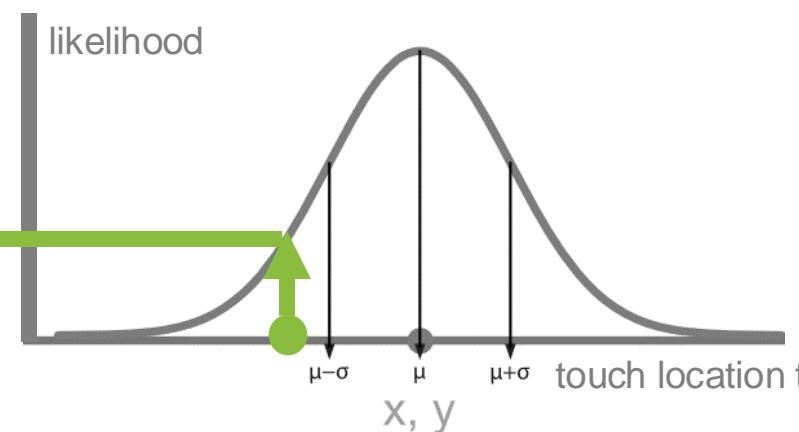
From x,y touch points to one Gaussian per key

[Azenkot and Zhai 2012]

Gaussian key model:

(shown in 1D here)

$$p(t|k) = \mathcal{N}(\mu_k, \sigma_k^2)$$

likelihood

touch location t

$\mu-\sigma$ μ $\mu+\sigma$

x, y
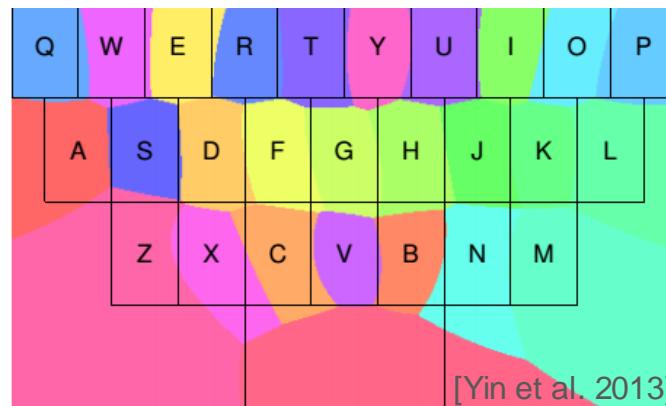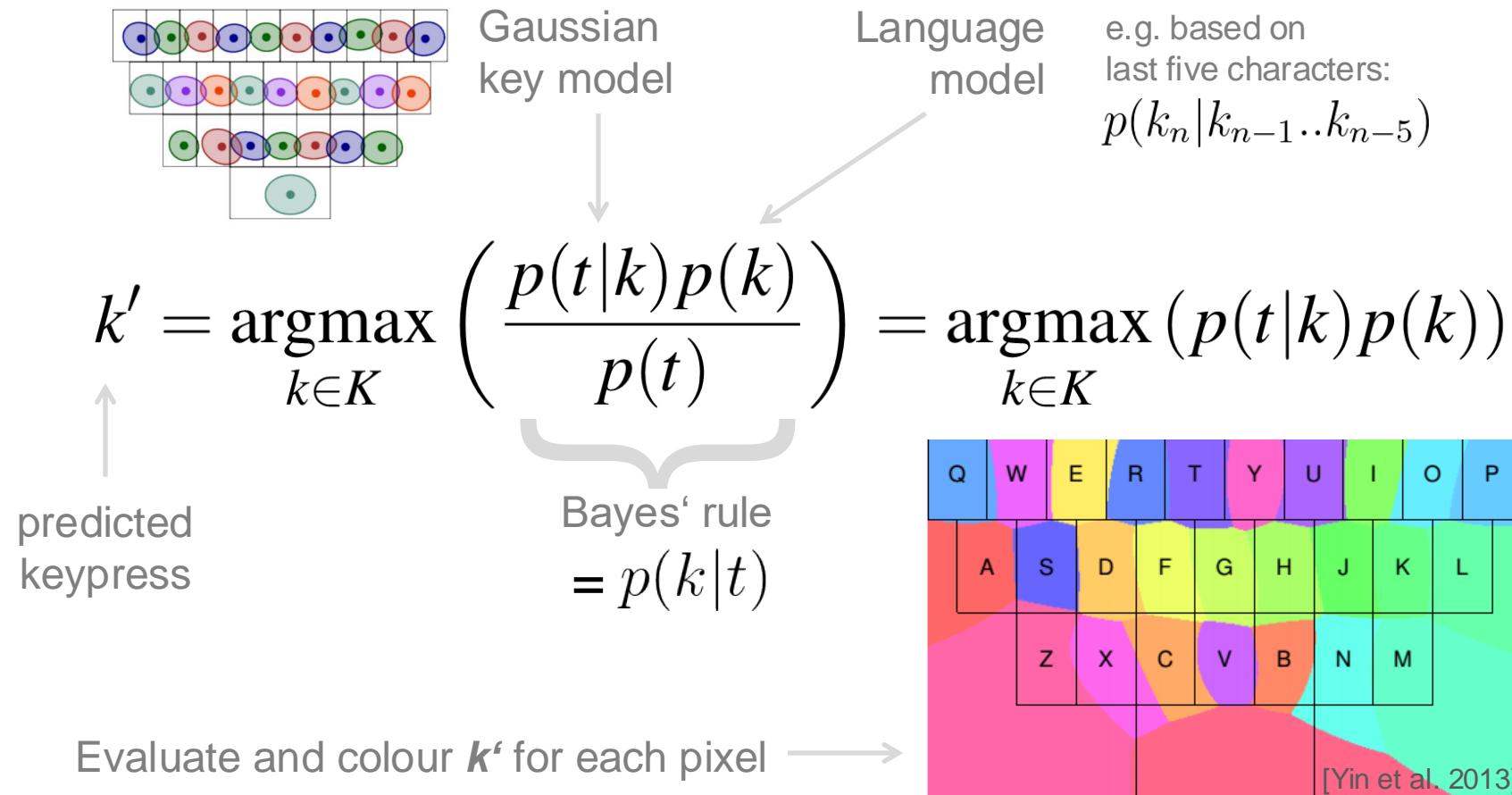
Probabilistic Keyboard Model

Which key does the user intend to press? i.e. „input decoding"

Gaussian key model

Language model

e.g. based on last five characters:

$$p(k_n|k_{n-1}..k_{n-5})$$

$$k' = \underset{k \in K}{\mathrm{argmax}} \left(\frac{p(t|k)p(k)}{p(t)} \right) = \underset{k \in K}{\mathrm{argmax}} \left(p(t|k)p(k) \right)$$

predicted keypress

Bayes' rule
$$= p(k|t)$$

Evaluate and colour **k'** for each pixel ⟶

[Yin et al. 2013]

DIY: Probabilistic Keyboard Model

```
touchX = ... // touch X coordinate
touchY = ... // touch Y coordinate
num_keys = ... // number of keys on keyboard
means = [...] // list of all key means (2D key locations)
variances = [...] // list of key variances (real values) or covariances (2x2 matrices)

probs = [] // list to store the likelihoods of each key being pressed
sum = 0 // variable to store sum of likelihoods for normalisation (see below)

for k = 0 to num_keys: // iterate over all keys
    // evaluate touch location under distribution of the key*:
    prob_t_given_k = multinormal_pdf(touchX, touchY, means[k], variances[k])
    // likelihood of key without touch info; uniform (here), or based on language*:
    prob_k = 1/num_keys
    // store product and add it to the sum of all likelihoods*:
    probs[k] = prob_t_given_k * prob_k
    sum = sum + probs[k]

// normalise, so that the likelihoods add up to 1*:
probs = probs / sum //note: "/" is element-wise division

// find most likely key:
pressed_key_index = argmax(probs)
// TODO for adaptation: update means and variances with new touchX and touchY
```
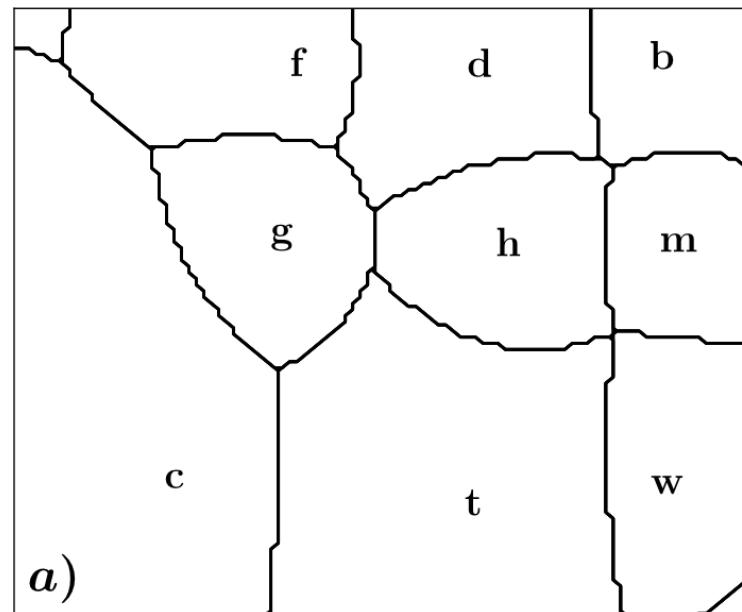
* in real implementation use logarithm and corresponding operations for numerical stability
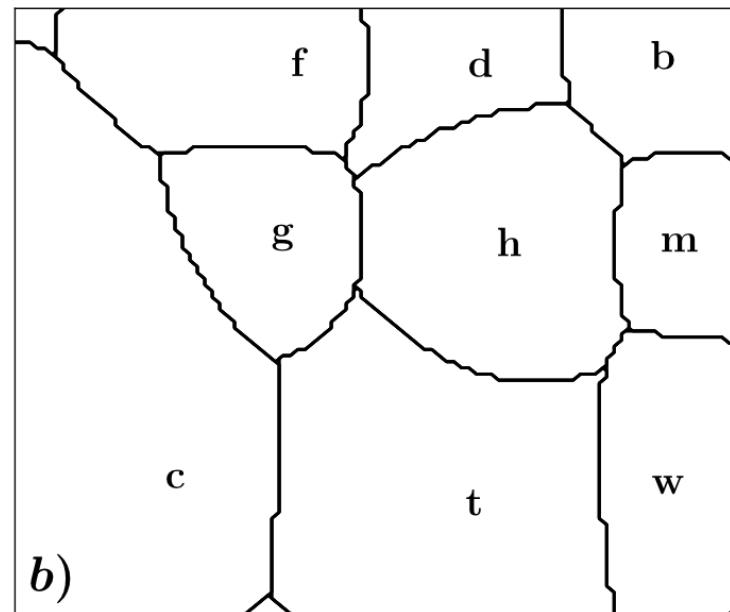
Language Model Influence
Example: bigram model for English

After „**n**":

After „**t**":
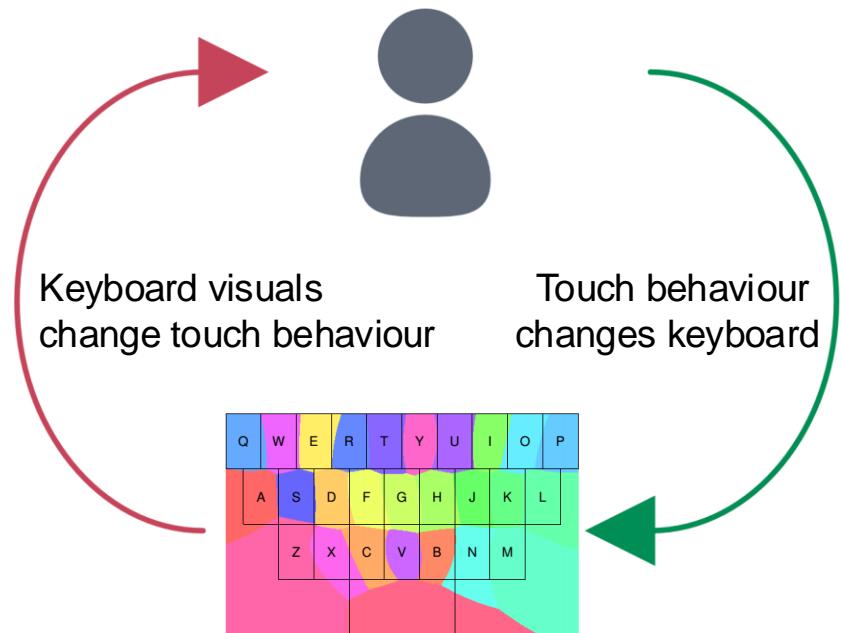
Adaptation in the Background

Why do our keyboards not look like this?
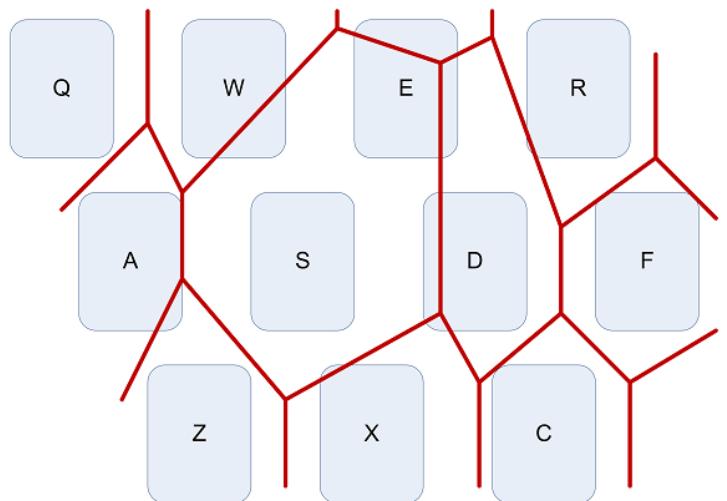
[Yin et al. 2013]

→ Avoid co-adaptation of user and system

Keyboard visuals change touch behaviour

Touch behaviour changes keyboard
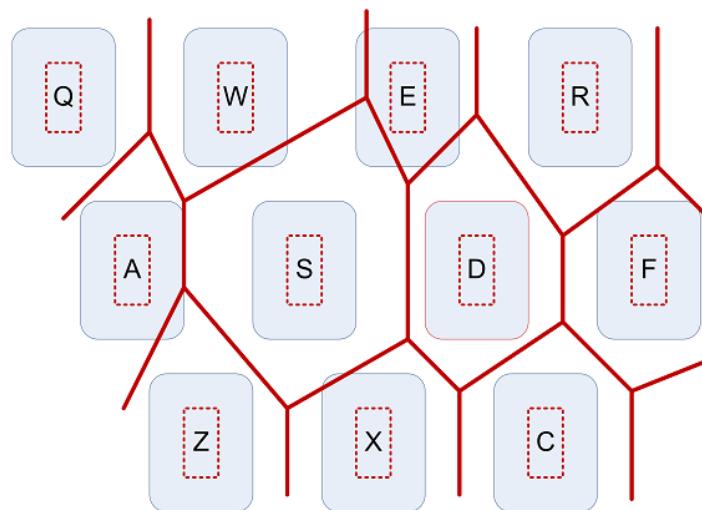
Adaptation vs Distortion

Unlimited adaptation

With protected key region
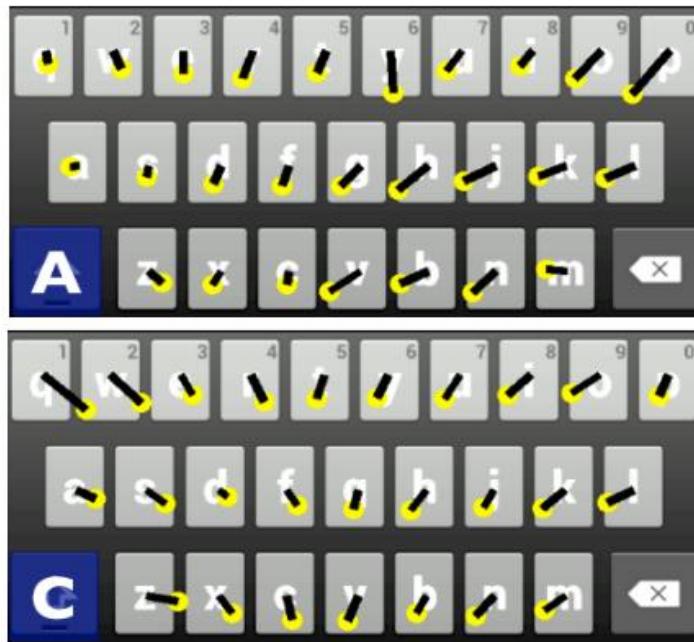
Here: (almost) impossible to type „e"!

[Gunawardana et al. 2010]
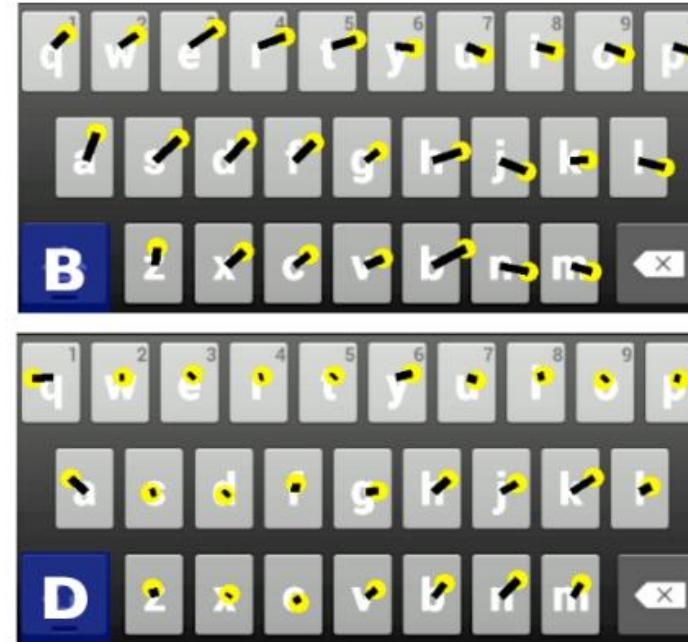
Context Adaptations

e.g. hand posture – „ContextType", Goel et al. 2013

Left thumb

Right thumb
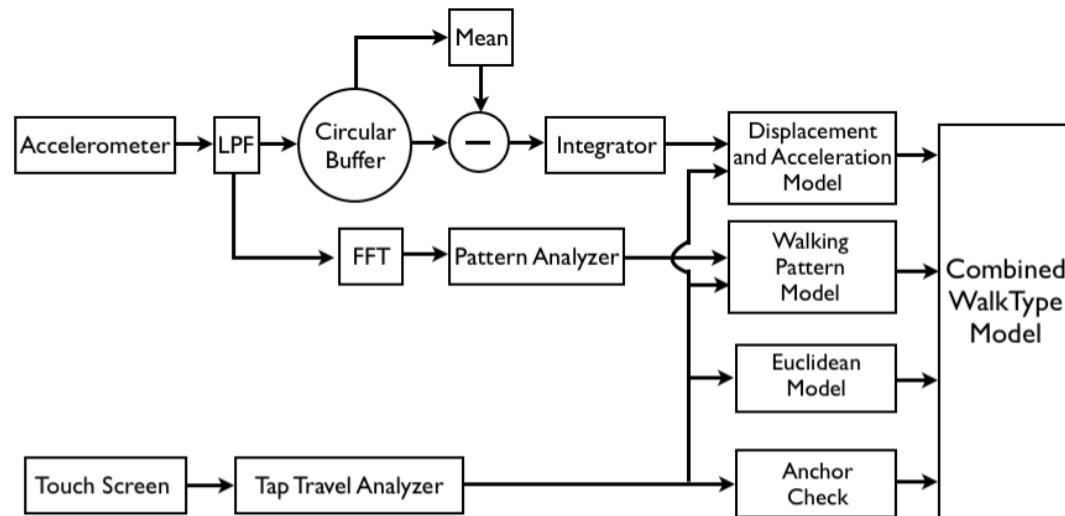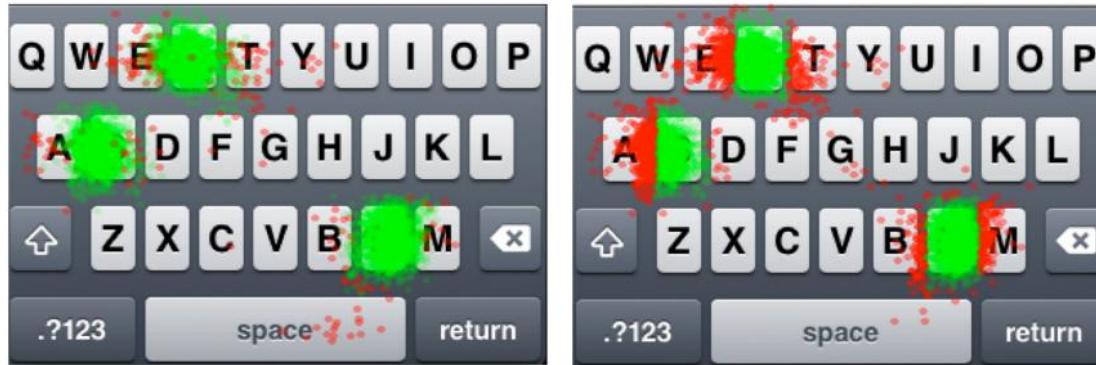
Index finger

Two thumbs

[Goel et al. 2013]

# Context Adaptations
e.g. walking – „WalkType", Goel et al. 2012

[Goel et al. 2012]

Decoding Typing Sequences

- Infer intended input after entering whole word or sentence
  - ▢ More evidence for inference
  - ▢ No need for user to pay attention to intermediate output
  - ▢ No intermediate feedback

- Example (sentence-based decoding):

„pleaseforwarxmetheatachement"

Hit enter ⬅
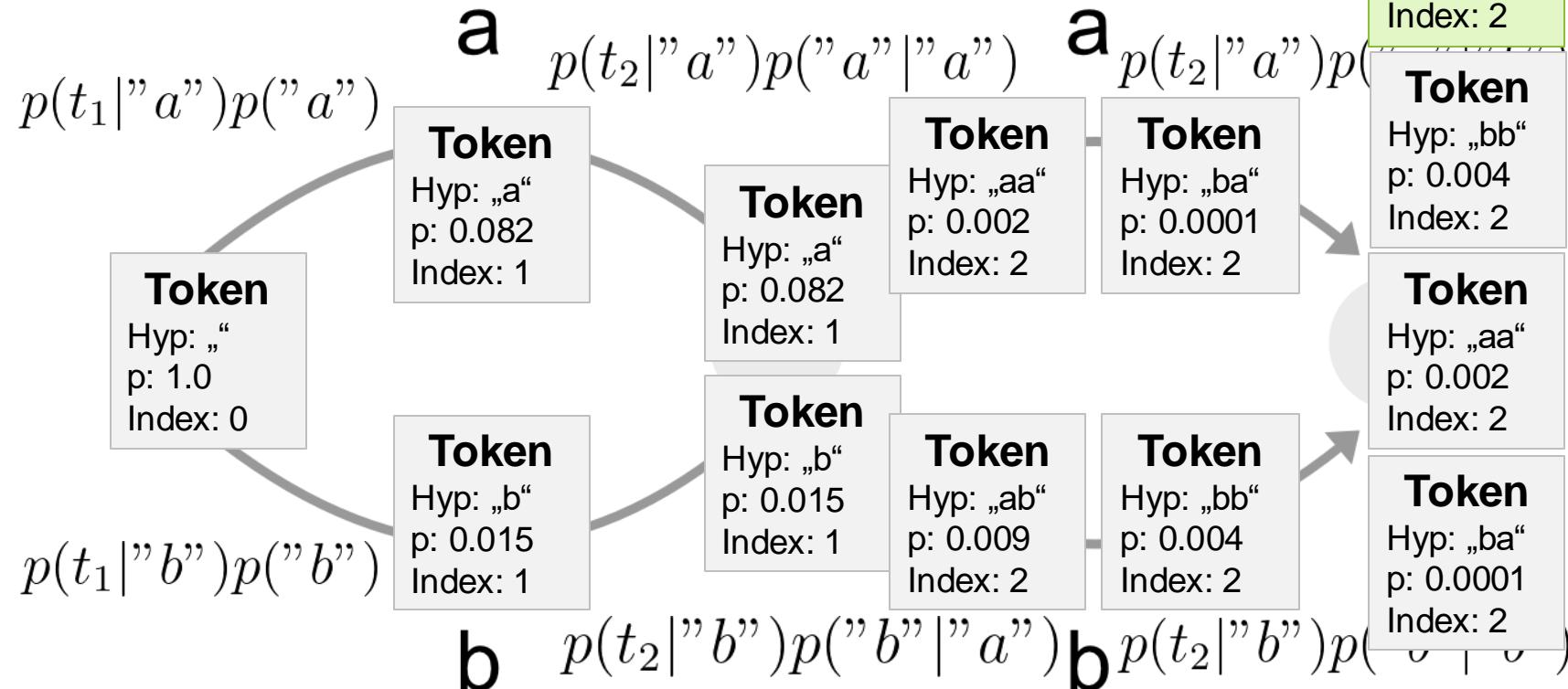
„Please forward me the attachement."

[Vertanen et al. 2015]

Decoding Typing Sequences

Token passing algorithm
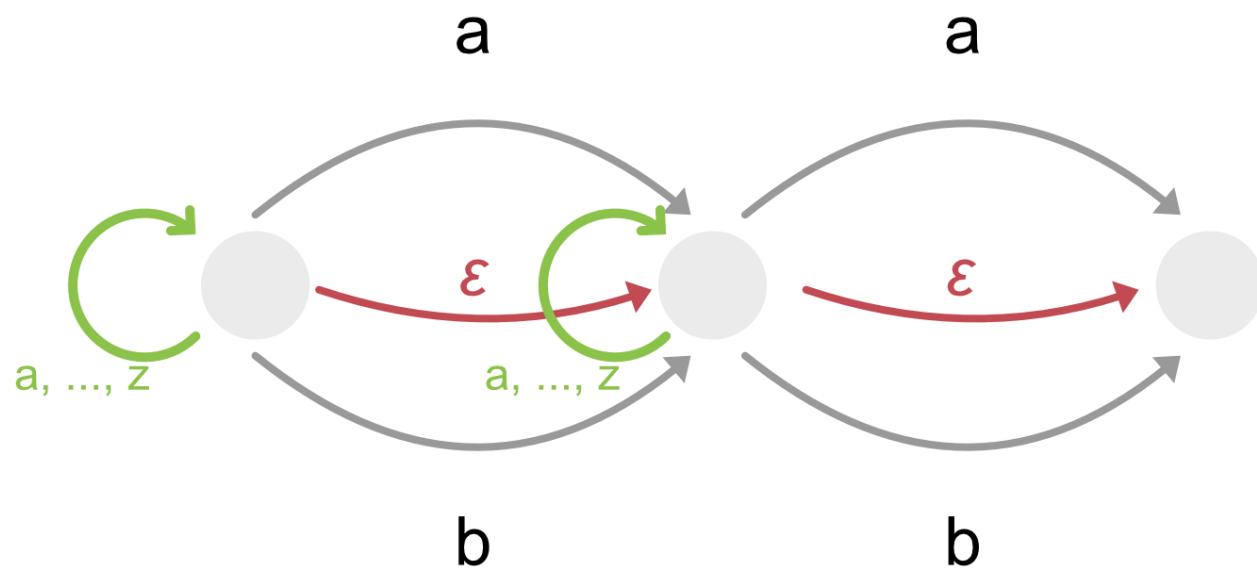
Decoding Typing Sequences

With insertion and deletion
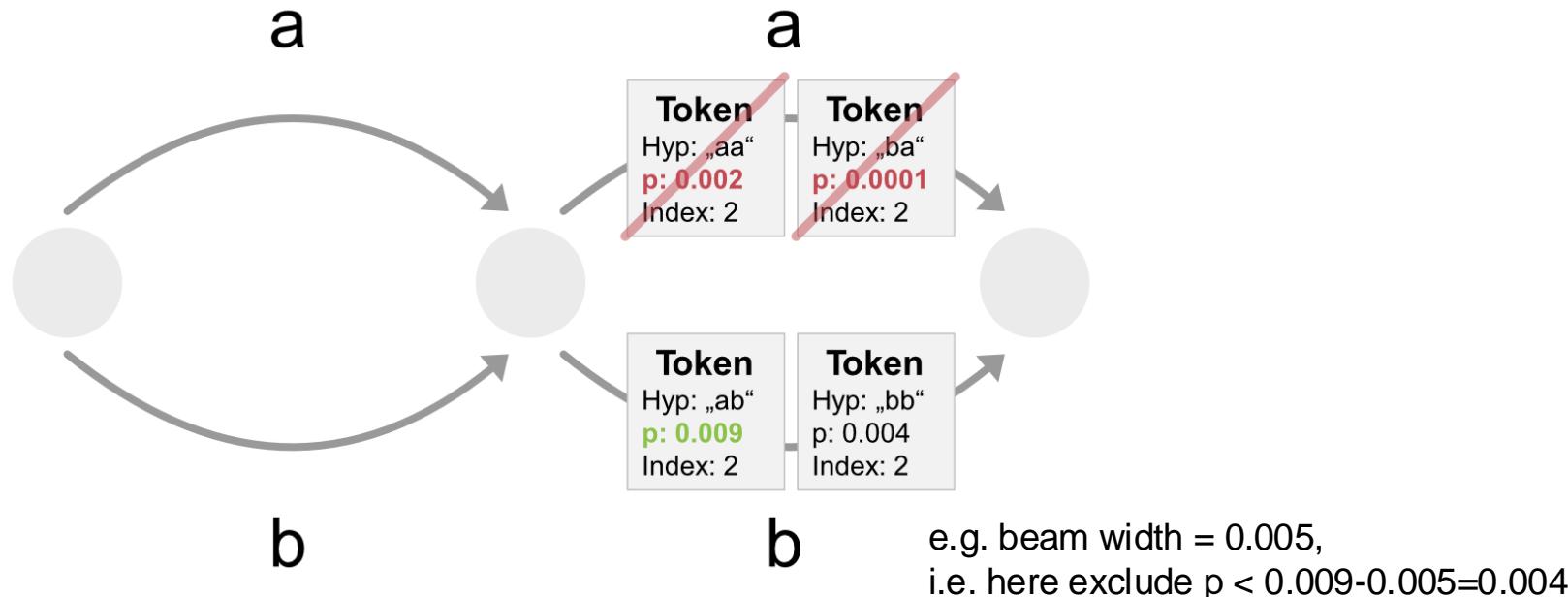
- Previous slide: Substitution-only decoder

- Extensions: **insertion** and **deletion** transitions, with „penalty"

Decoding Typing Sequences
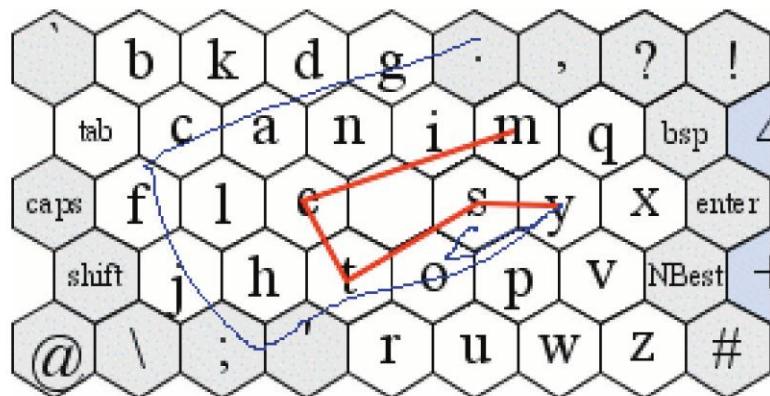
With beam search / pruning

- Problem: Large search space
  Substitution-only → exponential, Insertion → infinite

- Solution: **Beam search / pruning**
  Per index, only propagate tokens that are within a certain range
  (=„beam width") of the probability of the most likely token.

e.g. beam width = 0.005,
i.e. here exclude p < 0.009-0.005=0.004
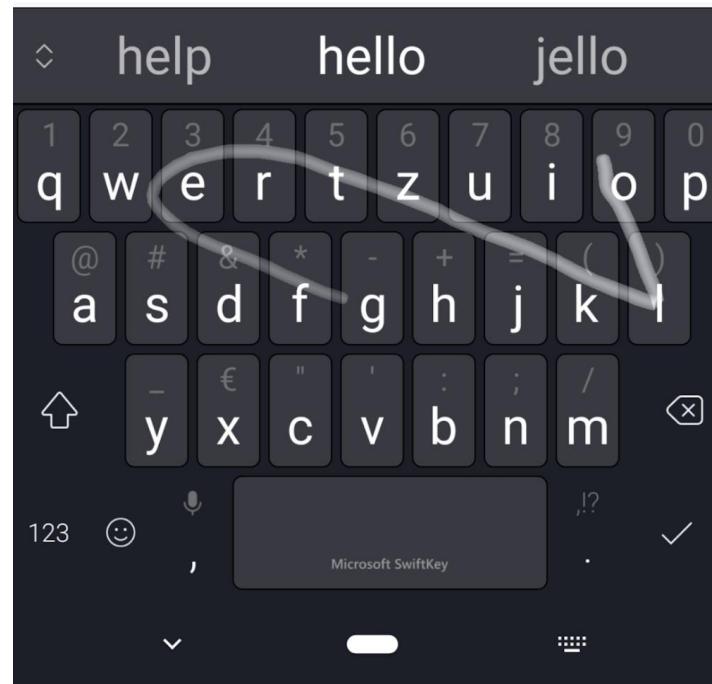
Gesture-based Decoding

Infer intended word from shape of finger trace on the keyboard

„SHARK²" [Kristensson and Zhai 2004]

Microsoft SwiftKey (screenshot Nov 2020)

Gesture-based Decoding

Shape model ↘   ↙ Language model

$$w' = \operatorname{argmax}_{w \in W} (p(trace|w)p(w))$$

Stored template (ideal) shapes
for all words in dictionary W

User's touch trace

Distance metric

e.g. see
Kristensson and Zhai,
2004

…
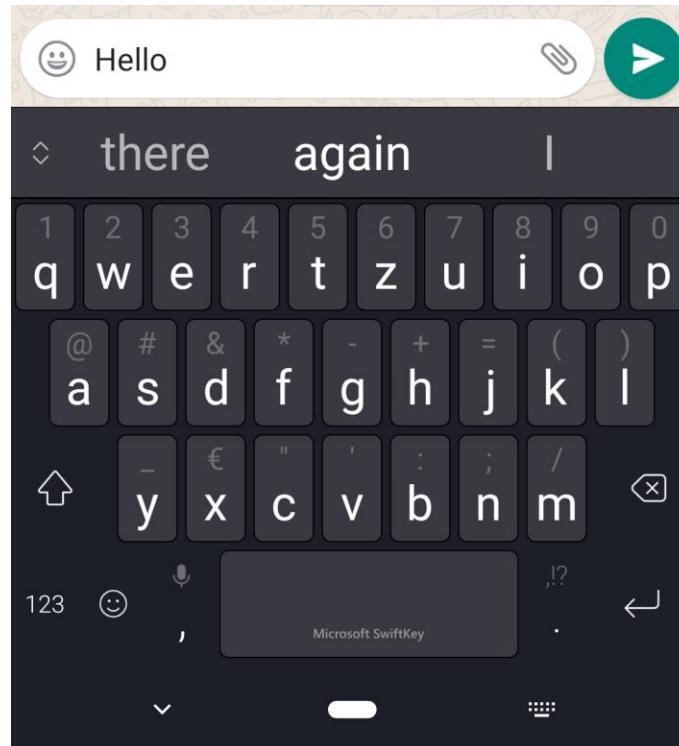
Word Prediction

- So far: Inference used touch input

- Now: Predict *next* word that user has not yet started to type, only using language context

$$p(w_t \mid w_{t-n} \ldots w_{t-1})$$

- E.g. n-gram word models, i.e. context of last n-1 words

- More recently: Deep Learning to include larger context

Summary

- Improving keyboards by probabilistically combining input information with language information

- **Adaptation:**
  - Individual input behaviour → adaptation to typist
  - Further sensors → adaptation to context

- **Prediction/Decoding:**
  - Single touch + language context → current key
  - Touch sequences + language context → current word/sentence
  - Language only → next word(s)

Questions & Discussion

- Which problems do adaptive and predictive keyboards address?

- Explain how touch information and language information can be combined for keyboard adaptation. What effect does this achieve at the pixel level?

- Explain decoding of touch sequences with token passing and beam pruning.

- Are adaptive and predictive keyboards „deceptive"?

- Which (further) factors could be considered for adaptation and word prediction in keyboards?

- Which other UIs beyond keyboards could benefit from similar approaches? What might have to be changed?

References

- Azenkot, S., & Zhai, S. (2012). Touch behavior with different postures on soft smartphone keyboards. *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, 251–260. https://doi.org/10.1145/2371574.2371612

- Buschek, D., Bisinger, B., & Alt, F. (2018). ResearchIME: A Mobile Keyboard Application for Studying Free Typing Behaviour in the Wild. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–14. https://doi.org/10.1145/3173574.3173829

- Findlater, L., & Wobbrock, J. (2012). Personalized input: Improving ten-finger touchscreen typing through automatic adaptation. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 815–824. https://doi.org/10.1145/2207676.2208520

- Goel, M., Findlater, L., & Wobbrock, J. (2012). WalkType: Using accelerometer data to accomodate situational impairments in mobile touch screen text entry. *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems - CHI '12*, 2687. https://doi.org/10.1145/2207676.2208662

- Goel, M., Jansen, A., Mandel, T., Patel, S. N., & Wobbrock, J. O. (2013). ContextType: Using Hand Posture Information to Improve Mobile Touch Screen Text Entry. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2795–2798. https://doi.org/10.1145/2470654.2481386

- Goodman, J., Venolia, G., Steury, K., & Parker, C. (2002). Language modeling for soft keyboards. *Proceedings of the 7th international conference on Intelligent user interfaces*, 194–195. https://doi.org/10.1145/502716.502753

- Gunawardana, A., Paek, T., & Meek, C. (2010). Usability guided key-target resizing for soft keyboards. *Proceedings of the 15th international conference on Intelligent user interfaces - IUI '10*, 111. https://doi.org/10.1145/1719970.1719986

- Holz, C., & Baudisch, P. (2011). Understanding touch. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2501–2510. https://doi.org/10.1145/1978942.1979308

References

- Kristensson, P. O., & Vertanen, K. (2014). The inviscid text entry rate and its application as a grand goal for mobile text entry. *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*, 335–338. https://doi.org/10.1145/2628363.2628405

- Kristensson, P.-O., & Zhai, S. (2004). SHARK2: A large vocabulary shorthand writing system for pen-based computers. *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology - UIST '04*, 43. https://doi.org/10.1145/1029632.1029640

- Vertanen, K., Memmi, H., Emge, J., Reyal, S., & Kristensson, P. O. (2015). VelociTap: Investigating Fast Mobile Text Entry using Sentence-Based Decoding of Touchscreen Keyboard Input. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 659–668. https://doi.org/10.1145/2702123.2702135

- Yin, Y., Ouyang, T. Y., Partridge, K., & Zhai, S. (2013). Making touchscreen keyboards adaptive to keys, hand postures, and individuals: A hierarchical spatial backoff model approach. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2775–2784. https://doi.org/10.1145/2470654.2481384

Further Reading:

- Bi, X., Li, Y., & Zhai, S. (2013). FFitts law: Modeling finger touch with fitts' law. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1363–1372. https://doi.org/10.1145/2470654.2466180

- Oulasvirta, A., Kristensson, P. O., Bi, X., & Howes, A. (Hrsg.). (2018). *Computational Interaction*. Oxford University Press.

For more content see: https://iui-lecture.org

Daniel Buschek
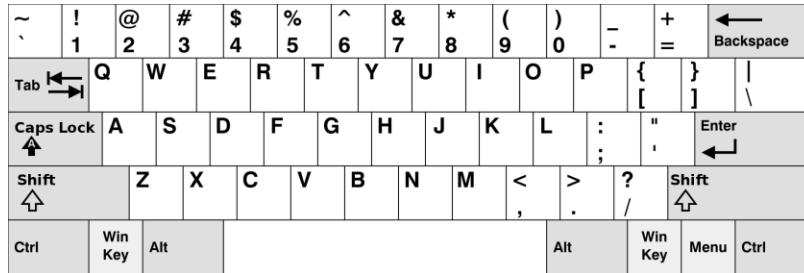
Optimization-based keyboard design

Daniel Buschek

Learning Goals
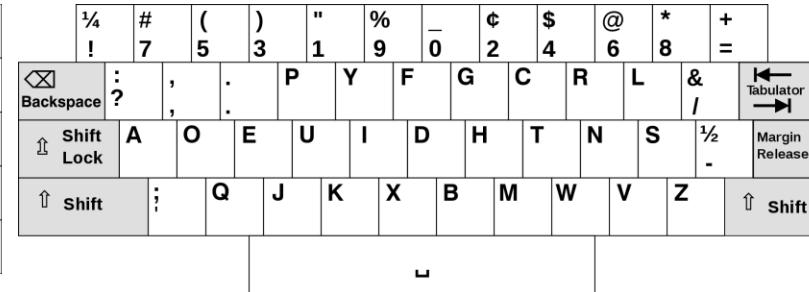
- Combinatorial optimization as a UI design approach

- Components of optimization

- Example optimizer and application results

Motivation: Fast typing without errors

- Are some layouts better than others?

- If so, how do we find the best one?

QWERTY, by Christopher Sholes, 1873

Dvorak, by August Dvorak, 1936
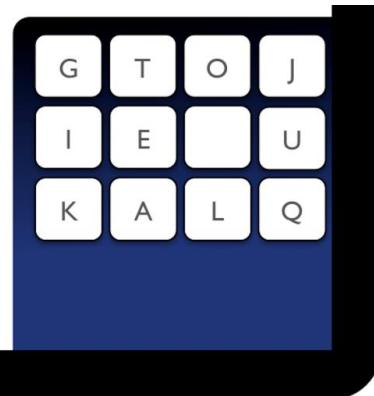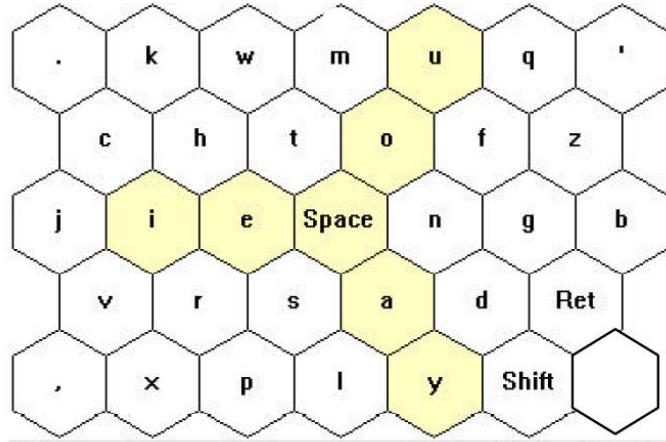
# Key Assignment Problem

How many layouts are there?   **26! = 4 * 10²⁶**

For comparison - stars in the universe: https://www.esa.int/Science_Exploration/Space_Science/Herschel/How_many_stars_are_there_in_the_Universe

Examples of Optimised Designs

Zhai et al. 2000, Dunlop and Levine 2012, Oulasvirta et al. 2013, Gong et al. 2018

What is „optimal"?

- **Design space:**
  Best among which options?

- **Design objective:**
  Best for what?

- **Optimizer:**
  How to find the best design?
 Best with which guarantees?

Design Space, formalised
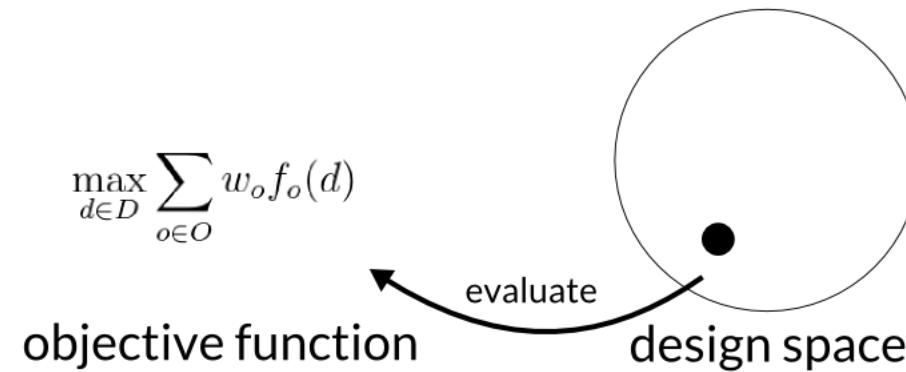
$$d = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}$$

Design space **D**
with **n** design variables

Design Space: Set of possible layouts

$$26! = 4 * 10^{26}$$

design space

Objective Function: How to judge a layout?

$$\max_{d \in D} \sum_{o \in O} w_o f_o(d)$$

objective function   evaluate   design space
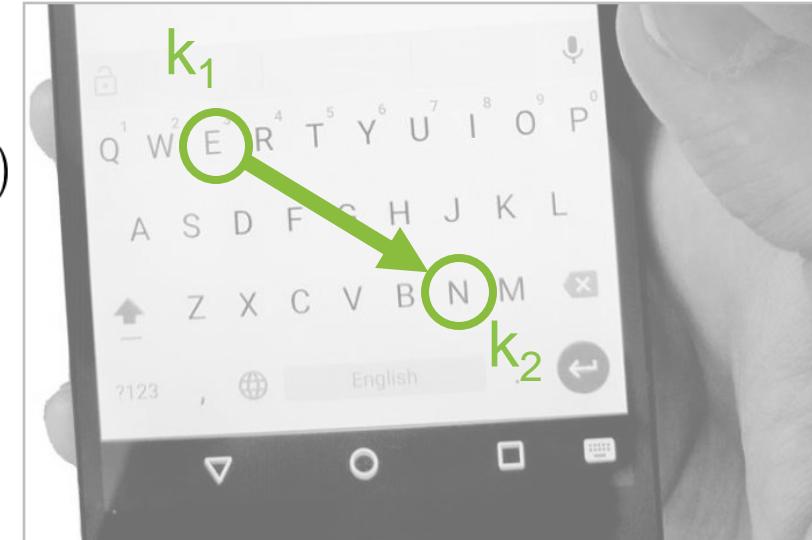
# Objective Function: How to judge a layout?

- Finger movement time (e.g. Fitts' law)

$$t(k_1, k_2) = a + b \log_2 \left(\frac{D}{W} + 1\right)$$

- Language properties (e.g. bigram frequencies)

e.g. $p("n"|"e") = 0.001$
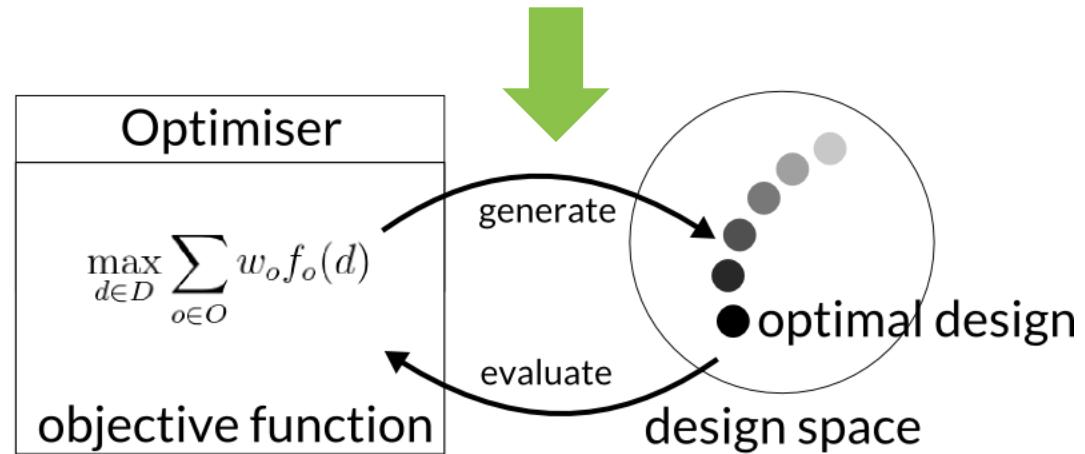
- Combined: mean time between two key presses

$$f(d) = \sum_{k_1 \in K} \sum_{k_2 \in K} p(d(k_2)|d(k_1))t(k_1, k_2)$$
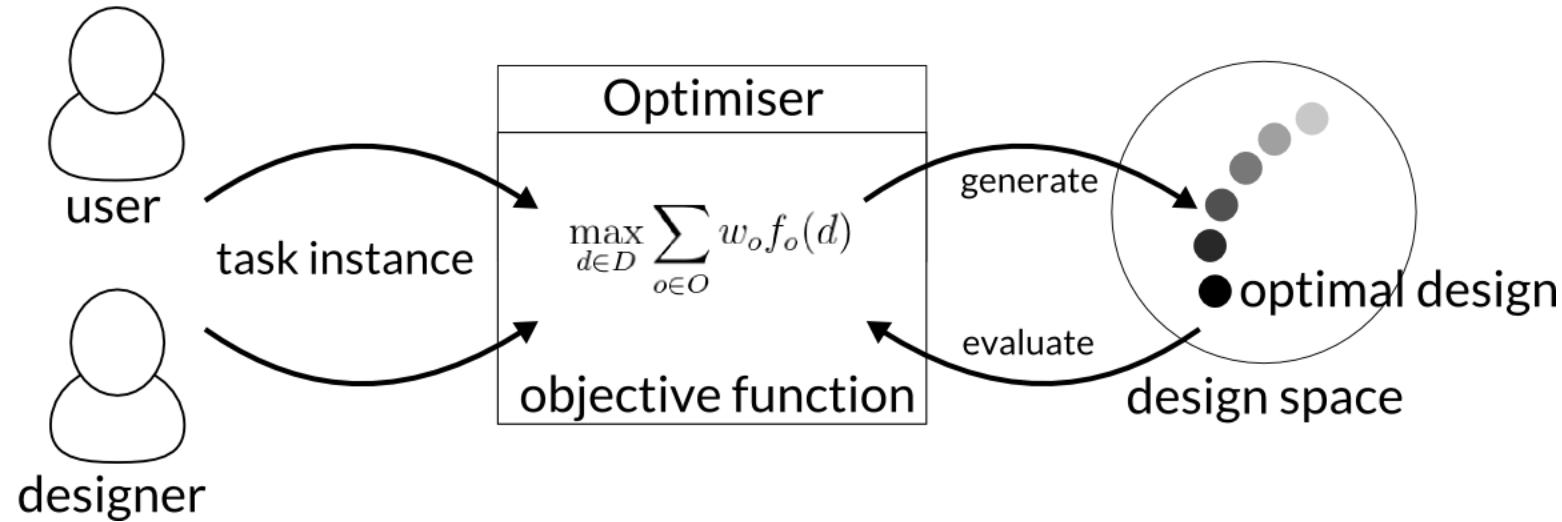
where the design d maps from keys to characters

Optimizer: How to pick layouts?

Design Task
e.g. keyboard layout optimization
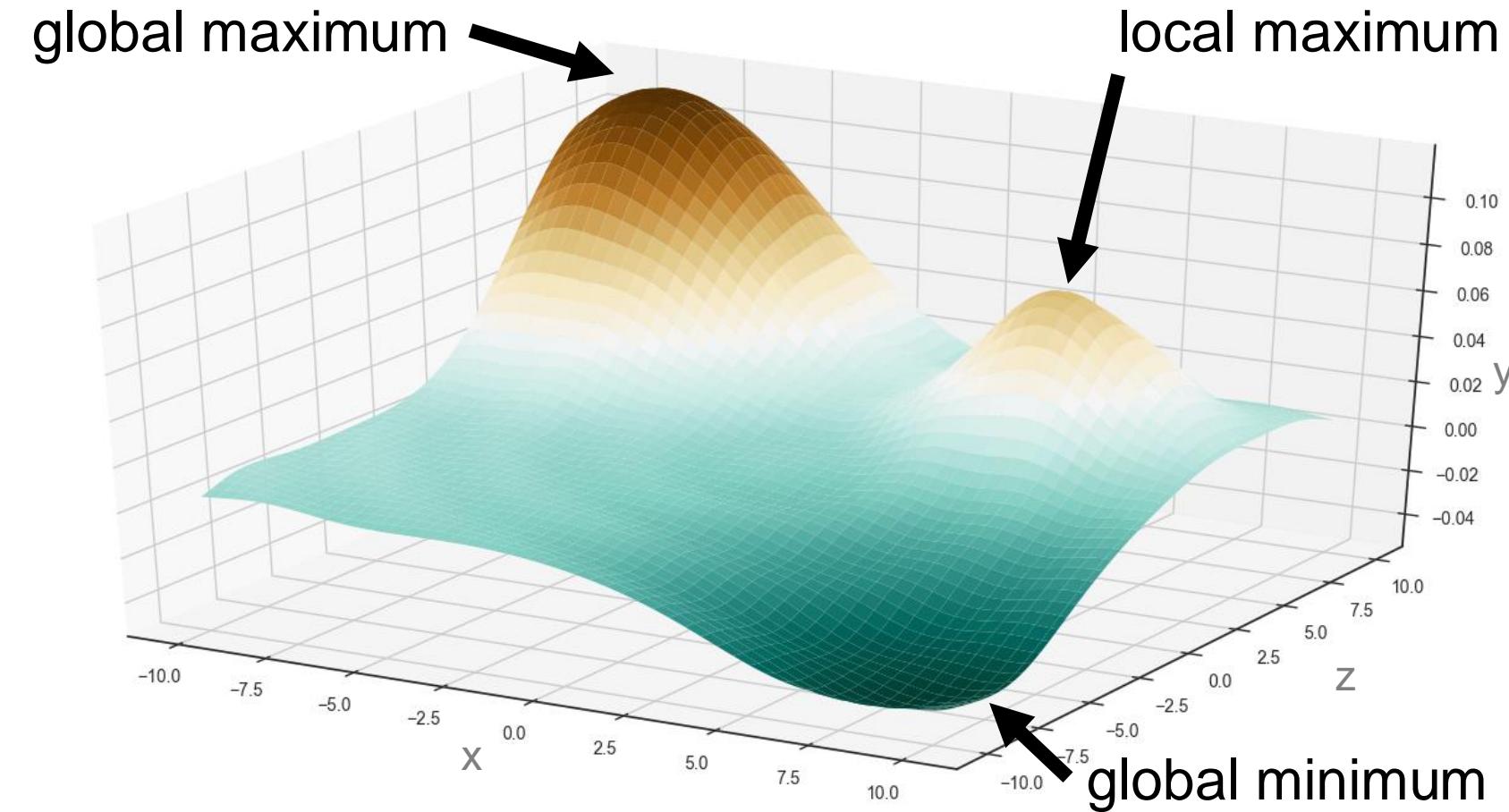
A Simple Optimizer
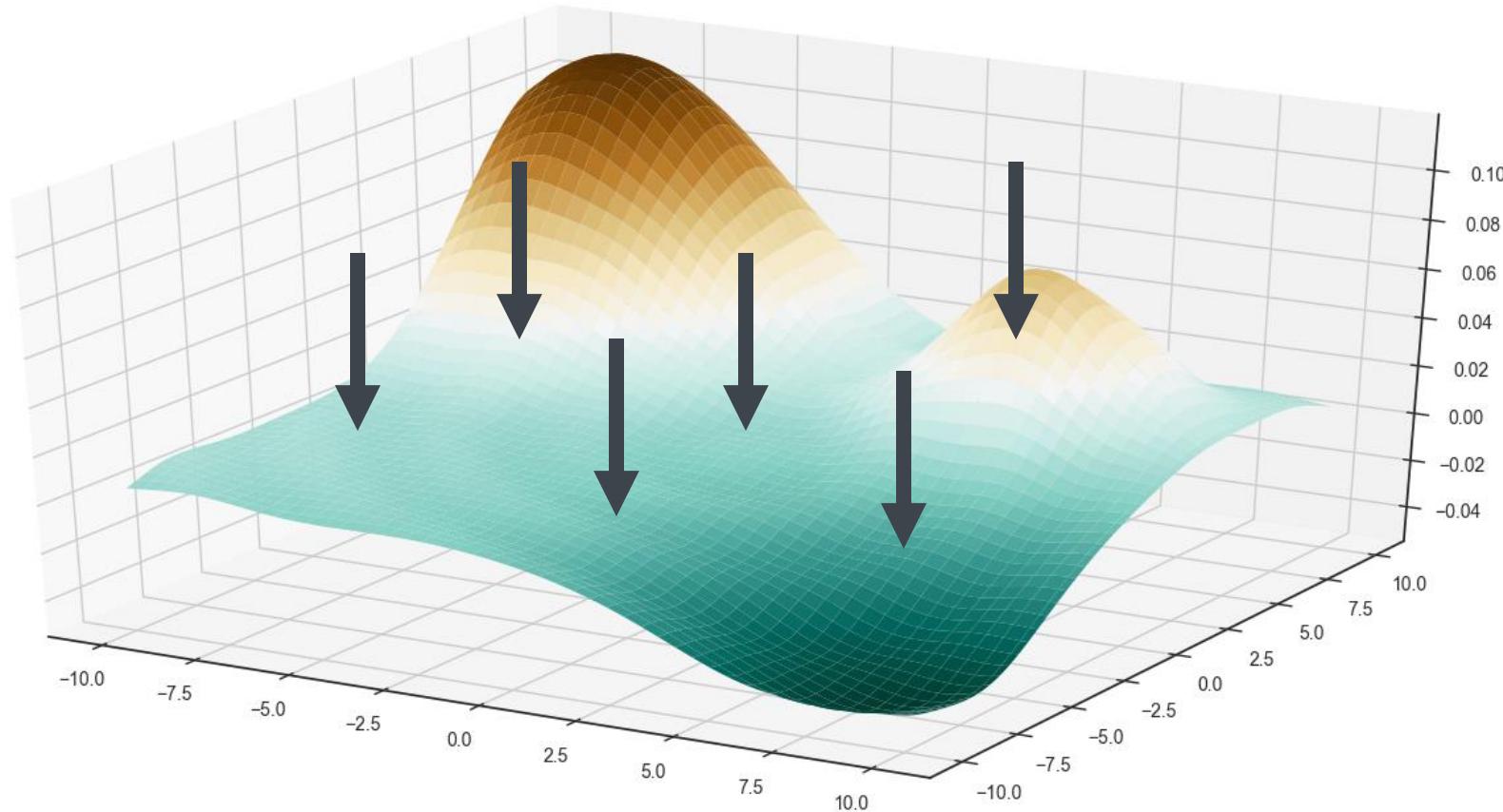
- Can you think of a trivial optimizer?

- Random Search:
  1. Generate random design
  2. Keep if better than current best design
  3. Repeat

Optimization Landscape

Here: objective function (y) across two design parameters (x, z)

global maximum

local maximum

global minimum

Random Guessing

Optimizers

- **Heuristic methods** (e.g. Simulated Annealing)
  - Flexible
  - Not guaranteed to find global optimum

- **Exact methods** (e.g. Integer Programming)
  - Guarantees
  - Less flexible objectives

Example: Simulated Annealing

- Metaphor: shaping hot metal

- Flexible at beginning (exploration)

- Gradually more rigid as it "cools down" (exploitation)

Photo by Kateryna Babaieva
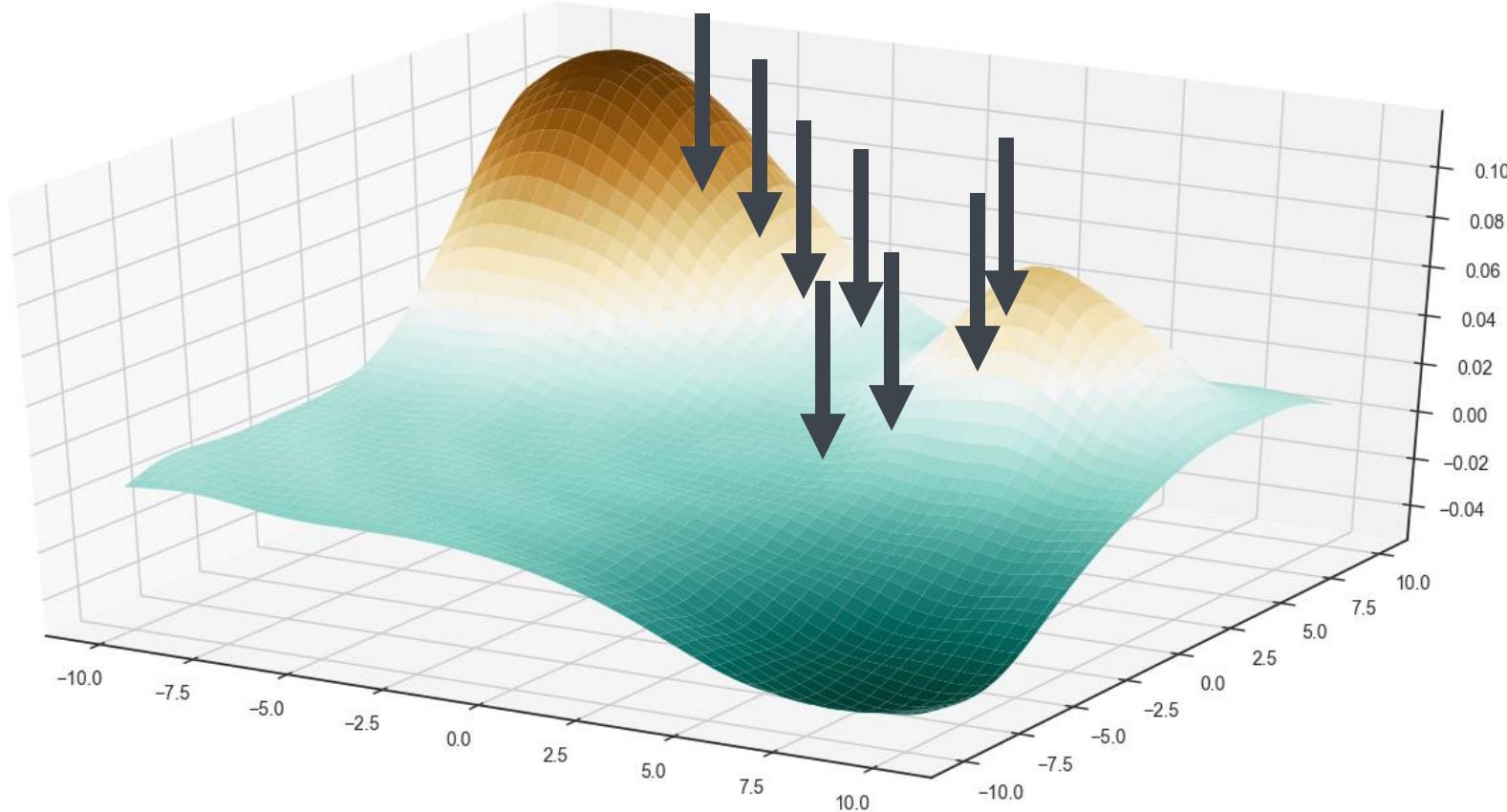
```
For i=0 to N:
  reduce temperature T
  generate neighbor design
  if better: go to neighbor
  else: still go with chance relative to T
```

Simulated Annealing

Example Results

"boxpum"
found with random search

WPM: 31.97

j k z f b o x p u m

d n i t c y r l v

h a e s g q w

"aero"
found with Simulated Annealing

WPM: 36.61

x y c h t i n d k z

b l a e r o f g j

v p m s u w q

Challenge – can you find a better layout than "aero"?
Use the provided python notebook as a starting point.

Example Results

With a modified objective function

"chat"
found with Simulated Annealing

WPM: 34.56

z   x   y   v   p   l   d   u   s   m

q   j   b   f   r   e   o   n   i

k   w   g   c   h   a   t

What was this layout optimized for?

→ Typing with right thumb, reduce thumb stretching

Potential of Optimization-based Design

- Obtaining information on the design problem and a formal specification

- Exploring a large design space comprehensively

- Improving quality and robustness of designs

- Estimating possible improvements

- Supporting human designers

- Optimization during use, personalised UIs

- Requires: Models of user behaviour, formal problem definition / objective function, computational capacity, …

Questions & Discussion

- Name and explain the key components of optimization-based UI design.

- How can designers influence obtained designs in this approach?

- Explain Simulated Annealing. Can you consider a design resulting from this method as „optimal"? Why (not)?

- If it is possible to find better designs than QWERTY, why are we not using them widely?

- Beyond keyboard layouts, which other UI design problems could be addressed with this approach? And which are hard to address in this way?

References

- Dunlop, M., & Levine, J. (2012). Multidimensional pareto optimization of touchscreen keyboards for speed, familiarity and improved spell checking. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2669–2678. https://doi.org/10.1145/2207676.2208659

- Gong, J., Xu, Z., Guo, Q., Seyed, T., Chen, X. „Anthony", Bi, X., & Yang, X.-D. (2018). WrisText: One-handed Text Entry on Smartwatch using Wrist Gestures. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*, 1–14. https://doi.org/10.1145/3173574.3173755

- Oulasvirta, A., Reichel, A., Li, W., Zhang, Y., Bachynskyi, M., Vertanen, K., & Kristensson, P. O. (2013). Improving two-thumb text entry on touchscreen devices. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2765–2774. https://doi.org/10.1145/2470654.2481383

- Zhai, S., Hunter, M., & Smith, B. A. (2000). The metropolis keyboard—An exploration of quantitative techniques for virtual keyboard design. *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology - UIST '00*, 119–128. https://doi.org/10.1145/354401.354424

Further Reading:

- Oulasvirta, A., Dayama, N. R., Shiripour, M., John, M., & Karrenbauer, A. (2020). Combinatorial Optimization of Graphical User Interface Designs. *Proceedings of the IEEE*, *108*(3), 434–464. https://doi.org/10.1109/JPROC.2020.2969687

- Simon, H. A. (1969). *The sciences of the artificial.* The MIT Press.

For more content see: https://iui-lecture.org

Daniel Buschek