Optimizing an Ambiguous Eyes-Free Keyboard

Dylan Gaines

Motivation

- Text entry is a very common task
- Users are not always able to see the keyboard
  - Multitasking, visual impairments, etc.
- Many eyes-free methods are Braille-based
  - Only about 10% of blind Americans know Braille

Ambiguous Keyboards

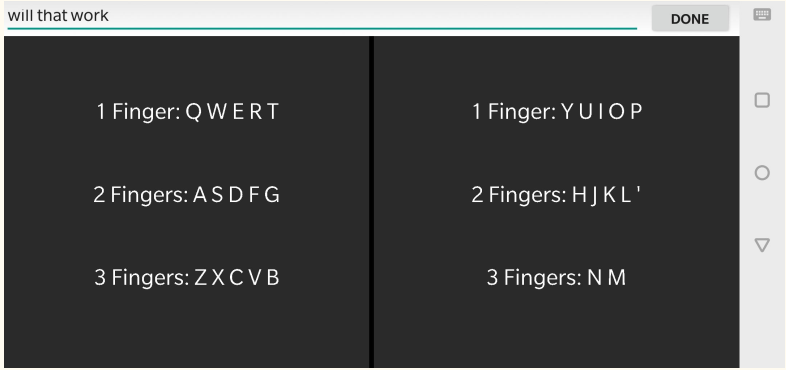
- Place multiple characters on the same key
- Standard telephone keypad
- Two ways to determine letter
  - Multiple Keystrokes
  - Disambiguation algorithms

A telephone keypad ambiguous keyboard[1]

[1] Qin et al. 2018. Optimal-T9: An Optimized T9-like Keyboard for Small Touchscreen Devices. In *ISS'18*.

Past Work: Tap123

- Split the keyboard into 6 groups
  - Based on Qwerty keyboard
- Users tap with 1, 2, or 3 fingers to indicate the row
- Left or right side of screen to indicate side of keyboard

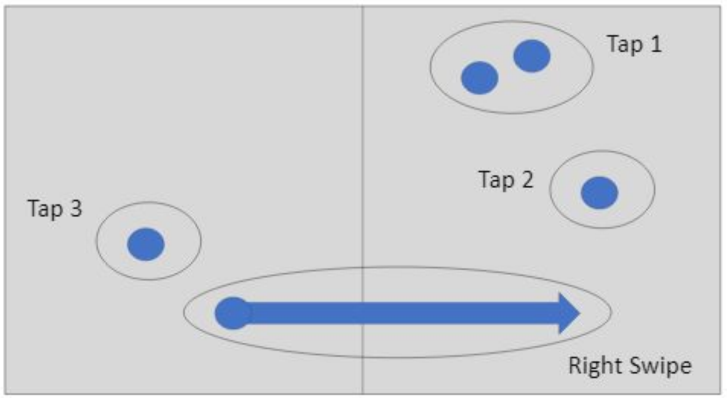


The Tap123 keyboard interface used in past work[1]

[1] Gaines. 2018. Exploring an Ambiguous Technique for Eyes-Free Mobile Text Entry. In *ASSETS'18*.

Past Work: Tap123

- Tap for each character
- Right swipe for space, left swipe for backspace
- Word-level disambiguation algorithm
- Swipe up or down to choose between matching words (N-Best List)



The above tap sequence yields the following N-Best list: *how*, *joe*, *hot*, *hit*, *low*, *lot* [1]

[1] Gaines. 2018. Exploring an Ambiguous Technique for Eyes-Free Mobile Text Entry. In *ASSETS'18*.

The Problem

- Many words have identical tap sequences
- Scanning through the N-Best List takes time
  - Required users to pause and verify after each word
- How can we determine the correct word?
  - Context can give us clues - train a language model
  - Still no guaranteed way

The Solution: Optimization

- We can adjust the groupings to reduce potential conflicts
- Arranging M characters on N keys is NP-Complete[1]
- What is the best way to group the characters?
    - We can look to past work for insight

[1] Lesher et al. 1998. Optimal Character Arrangements for Ambiguous Keyboards. In *IEEE Trans. on Rehab. Eng.*

Metrics: Travel Distance

- Many optimization papers minimize finger travel distance
  - Less distance traveled = faster entry
- Place frequent bigrams close to each other
  - Bigrams are sequences of two letters
- Not as relevant for location-independent approach

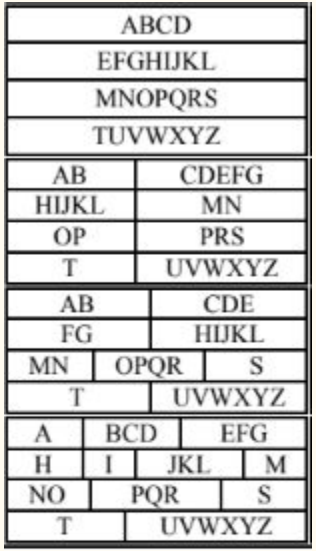
Metrics: Clarity

- Can be optimized in both ambiguous and unambiguous keyboards
- Some letters can be frequently substituted for each other to result in a valid word
  - Bad bigrams, "badgrams" [1]
- Unambiguous keyboards want badgrams not adjacent
- Ambiguous keyboards want frequent badgrams in separate groups

[1] Dunlop and Levine. 2012. Multidimensional Pareto Optimization of Touchscreen Keyboards for Speed Familiarity and Improved Spell Checking. In *CHI'12*.

Metrics: Familiarity

- Difficult for people to learn new keyboards
- Some researchers have strict familiarity constraints
  - Alphabetically Constrained (right top)
  - Qwerty Constraints (right bottom)



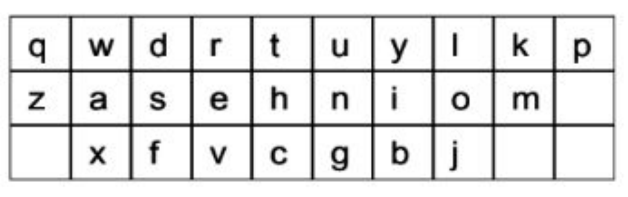
Alphabetically constrained ambiguous keyboards for 4, 8, 9, and 12 keys[1]

A Qwerty-constrained ambiguous keyboard[2]

[1] Gong and Tarasewich. 2005. Alphabetically Constrained Keypad Designs for Text Entry on Mobile Devices. In *CHI'05*.
[2] Qin et al. 2018. Optimal-T9: An Optimized T9-like Keyboard for Small Touchscreen Devices. In *ISS'18*.

Metrics: Familiarity

- Others have soft constraints
  - Letters can move one key in each direction from Qwerty position
- Still others include a Qwerty-similarity metric in optimization
  - Allows some keys to move far if most are close



Quasi-Qwerty soft constrained keyboard[1]

A keyboard optimized with a Qwerty-similarity metric (among others)[2]

[1] Bi et al. 2010. Quasi-Qwerty Soft Keyboard Optimization. In *CHI'10*.
[2] Dunlop and Levine. 2012. Multidimensional Pareto Optimization of Touchscreen Keyboards for Speed Familiarity and Improved Spell Checking. In *CHI'12*.

Algorithm: n-opt

- Character-level confusability matrix
  - Number of times one character is more probable than the true character
- Start with valid groupings
- Check every n-tuple to see if a swap improves optimization metric
  - If any swaps are made, repeat pass

[1] Lesher et al. 1998. Optimal Character Arrangements for Ambiguous Keyboards. In *IEEE Trans. on Rehab. Eng.*

Algorithm: n-opt

- Computationally expensive for large n
  - 5-opt largest tested in paper
- Not guaranteed to find global optimum
- Tested with many initial keyboards
  - 2-opt at first, then 5-opt on best

[1] Lesher et al. 1998. Optimal Character Arrangements for Ambiguous Keyboards. In *IEEE Trans. on Rehab. Eng.*

Algorithm: Pareto Optimization

- Used to optimize for multiple parameters
- Dunlop and Levine optimized for Travel Distance, Clarity, and Familiarity[1]
- Set of initial layouts taken through iterations of change
- Track Pareto optimal layouts (Pareto Front)
  - No other layout is better on all metrics
- Chose layout nearest the 45° line
  - Qin et al. chose layout with maximum average of metrics[2]

[1] Dunlop and Levine. 2012. Multidimensional Pareto Optimization of Touchscreen Keyboards for Speed Familiarity and Improved Spell Checking. In *CHI'12*.
[2] Qin et al. 2018. Optimal-T9: An Optimized T9-like Keyboard for Small Touchscreen Devices. In *ISS'18*.

Algorithm: Genetic Algorithms

- Gong and Tarasewich use Genetic Algorithms to optimize unconstrained layout[1]
- Candidates reproduce, crossover, and mutate
    - Many generations happen, keeping the best candidates

[1] Gong and Tarasewich. 2005. Alphabetically Constrained Keypad Designs for Text Entry on Mobile Devices. In *CHI'05*.

Proposed Methodology: Corpus Analysis

- Use Vertanen mobile phrase training set[1]
- Use VelociTap decoder to predict words using context[2]
- Generate table of badgrams for mispredicted words
    - Convert to probabilities by dividing by the sum

[1] Vertanen and Kristensson. 2021. Mining, Analyzing, and Modeling Text Written on Mobile Devices. In *Natural Lang. Eng.*
[2] Vertanen et al. 2015. VelociTap: Investigating Fast Mobile Text Entry using Sentence-based Decoding of Touchscreen Keyboard Input. In *CHI'15*.

Proposed Methodology: Metrics

- Primary metric Tap Clarity
- Iterate through test set, calculate word error rate
    - Computationally expensive
- Use sum of badgram probabilities for grouped characters
    - $Clarity_{badgram} = 1 - \sum_{\forall i,j \in \alpha} p_{ij}$,

Proposed Methodology: Algorithm

- Optimize both constrained and unconstrained
- Constraint based on alphabetical ordering
  - Some swaps allowed: minimum metric score
- Use n-opt and GA for unconstrained
- Use Pareto Optimization for constrained

Proposed Methodology: Number of Keys

- Will create both 4-key and 6-key layouts
- 6-key likely to have better disambiguation accuracy
  - Fewer characters per key, fewer collisions
- 4-key easier to use with one hand
  - Can be fully location-independent

Proposed Methodology: Final Testing

- Determine word error rate on test phrase set
- Compare all candidate layouts
    - Find best 4-key and best 6-key constrained and unconstrained
- Compare final four in longitudinal user study
    - Compare entry and error rates over time

Conclusions

- Optimizing a location-independent ambiguous keyboard
- Considering both familiarity and disambiguation performance
- Generate candidate 4-key and 6-key layouts for user testing

References

Ryan Qin, Suwen Zhu, Yu-Hao Lin, Yu-Jung Ko, and Xiaojun Bi. 2018. Optimal-T9: An Optimized T9-like Keyboard for Small Touchscreen Devices. In *ISS '18: Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*. 137-146.

Dylan Gaines. 2018. Exploring an Ambiguous Technique for Eyes-Free Mobile Text Entry. In *ASSETS '18: Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*. 471-473.

Gregory W. Lesher, Bryan J. Moulton, and D. Jeffery Higginbotham. 1998. Optimal Character Arrangements for Ambiguous Keyboards. In *IEEE Transactions on Rehabilitation Engineering*. 415-423.

Mark D. Dunlop and John Levine. 2012. Multidimensional Pareto Optimization of Touchscreen Keyboards for Speed, Familiarity and Improved Spell Checking. In *CHI '12: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2669-2678.

Jun Gong and Peter Tarasewich. 2005. Alphabetically Constrained Keypad Designs for Text Entry on Mobile Devices. In *CHI '05: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 211-220.

Xiaojun Bi, Barton A. Smith, and Shumin Zhai. 2010. Quasi-qwerty Soft Keyboard Optimization. In *CHI '10: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 283-286.

Keith Vertanen and Per Ola Kristensson. 2021. Mining, Analyzing, and Modeling Text Written on Mobile Devices. Natural Language Engineering. 1-33.

Keith Vertanen, Haythem Memmi, Justin Emge, Shyam Reyal, and Per Ola Kristensson. 2015. VelociTap: Investigating Fast Mobile Text Entry using Sentence-based Decoding of Touchscreen Keyboard Input. In CHI '15: Proceedings of the ACM Conference on Human Factors in Computing Systems. 659-668.