2021

Towards Location-Independent Eyes-Free Text Entry

Dylan C. Gaines
Michigan Technological University, dcgaines@mtu.edu

TOWARDS LOCATION-INDEPENDENT EYES-FREE TEXT ENTRY

By

Dylan C. Gaines

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Computer Science

MICHIGAN TECHNOLOGICAL UNIVERSITY

2021

This thesis has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Computer Science.

Department of Computer Science

Thesis Advisor:	*Dr. Keith Vertanen*
Committee Member:	*Dr. Scott Kuhl*
Committee Member:	*Dr. Timothy Havens*
Committee Member:	*Dr. Ricardo Eiris*
Department Chair:	*Dr. Linda Ott*

Dedication

To Dr. Keith Vertanen,

for his guidance, encouragement, and never-ending patience,

To Jenna,

for always believing in me, and

To My Parents,

without whom I would not be where I am today.

Contents

List of Figures

List of Tables

Abstract

We propose an interface for eyes-free text entry using an ambiguous technique and conduct a preliminary user study. We find that user are able to enter text at 19.09 words per minute (WPM) with a 2.08% character error rate (CER) after eight hours of practice. We explore ways to optimize the ambiguous groupings to reduce the number of disambiguation errors, both with and without familiarity constraints. We find that it is feasible to reduce the number of ambiguous groups from six to four. Finally, we explore a technique for presenting word suggestions to users using simultaneous audio feedback. We find that accuracy is quite poor when the words are played fully simultaneously, but improves when a slight delay is added before each voice.

Chapter 1

Introduction

In today's society, text entry is a task that most people complete several times a day on a variety of devices. Smartphones, laptop computers, and smartwatches are all increasingly common. However, so are situations in which a user may not be able to visually see a keyboard. Perhaps their visual attention is needed elsewhere if they are multitasking, or perhaps the user has a visual impairment that prevents them from seeing the keyboard. On a physical keyboard many people are able to touch type, using only tactile feedback and knowledge of where the keys are located. However, this becomes much more difficult when the keyboard is virtual and the tactile separation between keys is removed.

1.1 Existing Solutions

1.1.1 Braille-Based Interfaces

Many research solutions focus on using Braille-based gestures to map to characters. Perkinput [1] encoded characters as six-bit binary strings using the characters' Braille representation. These binary strings were entered using either three fingers on each hand or using two three-fingered touches with a single hand. The authors found that users were able to type an average of 17.56 words per minute with an uncorrected error rate of 0.14% using a single hand, or 38.0 words per minute with an error rate of 0.26% using the two-handed approach.

BrailleType places six targets on a touchscreen–one in each corner and one along each long edge–that correspond to the dot locations in a Braille character [15]. Users mark these dots by dragging their finger to each dot in the encoding, waiting for an audio confirmation at each location. Double tapping on the screen inputs the character represented by the currently marked dots, and swiping to the left clears any marked dots or deletes the last character if no dots are marked. In user studies, participants entered text at 1.45 words per minute using BrailleType compared to 2.11 words per minute using a technique identical to VoiceOver. Error rates were on average 14.12%

for VoiceOver and 8.91% for BrailleType.

Using BrailleTouch, a user would hold the device with the screen facing away from them and use the first three fingers on each hand to tap the Braille encoding for each character [17]. In user testing, the authors found that expert Braille typists obtained an average entry rate of 23.2 words per minute with a 14.5% error rate in their final of five sessions using BrailleTouch on a smartphone. For comparison, they averaged 42.6 words per minute in the final session on a standard physical Braille keyboard with an error rate of 5.3%.

In contrast to Perkinput [1] and BrailleTouch [17] which split the 3x2 Braille matrix into left and right sides, TypeInBraille [13] allows users to enter characters one row at a time (i.e. using three actions instead of two). Users tap on the left or right side (or both) to indicate that that dot is raised, or with three fingers to indicate no dots in that row. Swiping to the right indicates the end of a character. In user studies, the authors found that participants were able to achieve an average of about 7 words per minute with just under 5% error rate using TypeInBraille.

1.1.2 Non-Braille Interfaces

Although these methods have been able to produce results that are quite good, statistics released by the National Federation of the Blind in 2009 show that less than 10%

of legally blind Americans know Braille [14]. Additionally, the vast majority of people with vision are unlikely to know Braille, creating the need for eyes-free text entry methods that are not Braille-based. Some commercial solutions to this problem, such as Apple's VoiceOver, allow users to explore the keyboard with audio feedback before confirming their selection for each character [3]. An additional non-Braille approach relies upon character recognition. Tinwala and MacKenzie developed a *Graffiti*-based approach [18] in which users would perform gestures similar to drawing each character on the screen. They found that users entered text with at an average of 10.0 words per minute with an error rate of 4.3%.

No-Look Notes arranges the characters into eight groups in a circle on the screen [3]. The groups are identical to the groups of letters found on a standard phone keypad and are explored in a manner similar to Apple's VoiceOver. The user explores the segments by dragging their finger on the screen until the audio feedback indicates the segment they want. Tapping a second finger on the screen selects the segment and brings up the characters in that segment to be explored and selected in a similar fashion. In a direct comparison, the authors found that participants entered text using No-Look Notes at 1.32 words per minute and only 0.66 words per minute using VoiceOver. Additionally, they found that users entered 0.11 incorrect characters per correct character in the target word using No-Look Notes, and 0.60 incorrect characters using VoiceOver.

1.2 Project Motivation

While these text entry methods all work, VoiceOver and No-Look Notes both require multiple user actions to enter a single character [3]. The *Graffiti*-based approach only required a single action, but that action was a gesture that the authors mentioned may not be properly recognized all the time [18]. These factors result in entry rates that are much lower than those reported for single-action Braille-based methods such as two-handed Perkinput [1] and BrailleTouch [17]. This creates a need for a non-Braille eyes-free text entry method that uses a short, single-action gesture, such as a tap, to input each character. A single tap would take a fraction of the time needed to enter a character using VoiceOver or No-Look Notes and would make eyes-free text entry much more efficient for blind and sighted users alike. Removing as much location dependency as possible removes the need for exploration and confirmation techniques such as that used by VoiceOver.

A non-Braille approach would also make this technique more accessible to a user with a situational impairment. A user that is walking down the street, for example, may wish to send a message without looking at their phone. A location-independent approach also has applications beyond a touchscreen. The main benefit of using a touchscreen is that precise touch coordinates can be captured. If we no longer need locations, we can map each group of characters to some gesture that can be detected

another way. This could have applications in augmented and virtual reality head-mounted displays, or in a wearable device that detects finger movements.

Chapter 2

Interface Design

In 2016, Vertanen proposed a text entry technique based on the number of fingers a user tapped with [19]. Taps ranging from one to five fingers signified ambiguous groups of characters that could be based on a variety of mappings. After each word, the sequence of taps was fed to the VelociTap decoder [23] to determine the most likely intended word based on the taps and the context of what had already been typed. While some mappings were Braille-based, others were simply alphabetical and removed the need to know Braille encodings. We can extend this work to create an input method using this technique that generates mappings from a Qwerty keyboard layout. In this chapter we will discuss the design of this interface as well as a small user study. The goal of this study was to determine if users are able to learn this new text entry technique and obtain similar or better performance than other eyes-free

interfaces.

2.1 System Description

The Tap123 keyboard accepts taps of between one and three fingers. The number of fingers corresponds to the row of a Qwerty keyboard that the intended letter is in. In the original technique proposed by Vertanen [19], only the number of fingers was used to determine the group of characters. However, our interface uses the side of the screen on which the taps occur to further split each keyboard row into a left and right side. The exact groupings used can be seen on the interface depicted in Figure 2.1.

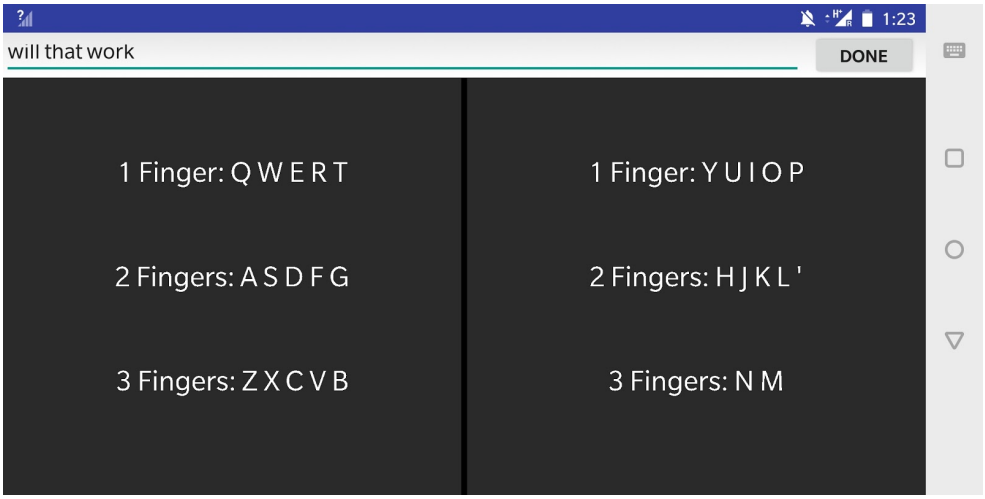


Figure 2.1: The Tap123 Interface

In addition to these multi-finger touch events to indicate groups of characters, a left

swipe can be used to backspace a tap, while a right swipe sends the tap sequence to the VelociTap decoder [23] for recognition and inserts a space. If a left swipe is performed immediately after decoder recognition, it will instead delete the entire word.

As in [19], the VelociTap decoder is configured to treat each combination of finger count and side as a key with multiple possible letters. The decoder is configured to return a list of the 6 most likely words, which we will refer to as the N-Best list, based on the current sequence of taps and the previously entered text (left context). If the first word returned is not the word the user intended, swiping up or down on the keyboard will iterate forwards or backwards through N-best list, respectively.

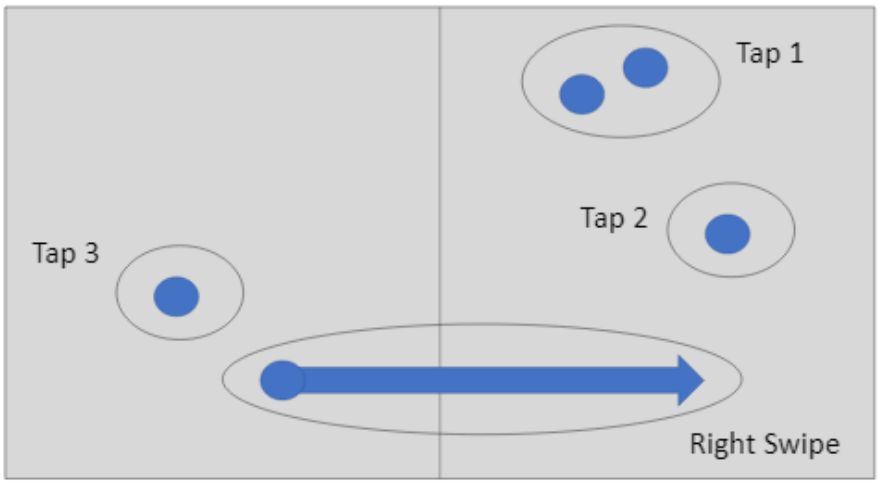


Figure 2.2: An example of a word entered using Tap123

An example event sequence of a word being entered is shown in Figure 2.2. The first tap consists of two fingers on the right side of the screen, while the second tap is one

finger on the right side. The third and final tap is one finger on the left side of the screen. The tap sequence is sent to the decoder when the interface receives the right swipe gesture. The first word recognized in this case with no left context is ‘how’, with the remainder of the N-Best list consisting of ‘joe’, ‘hot’, ‘hit’, ‘low, and ‘lot’.

With the absence of visual feedback, Tap123 instead provides audio feedback to convey information to the user. When any tap is performed, the keyboard speaks the letters in the group that corresponds to that tap. Upon recognition or an up or down swipe, the keyboard speaks the word that has been selected. If a word or tap is deleted, the keyboard will inform the user what they deleted. If the user taps and holds their finger on the screen for 600 ms or longer (long press), this will prompt the keyboard to read the contents of the entry text field. During the user study, a long press caused the application to reread the prompt text and then the contents of the entry field.

Users also need a way to close the keyboard interface when they are finished entering text. We chose the gesture of tapping with six fingers (the three on each hand they were already using to enter text). In practice the interface was configured to close if it recognized five or more simultaneous touch events since this would not occur in normal text entry and we wanted the gesture to work even if the device failed to recognize one of the fingers.

2.2 User Study

Our user study consisted of a series of sessions designed to introduce participants to the entry method slowly and allow them to learn the interactions. Though all four participants were sighted, the device on which entry occurred was obscured from the participant's view during all text entry tasks. In each session, participants were asked to type given phrases using Tap123. These phrases were read to the participants using Android's native Text-to-Speech. Participants completed 8 sessions, each totaling about 40 minutes of text entry broken into about 10-minute segments. Since each session was time-based, the exact number of prompts completed varied, but across all participants there were an average of 86 prompts per session. Participants were asked to enter text bimanually with the device flat on the table.

The first few sessions were designed to build slowly and introduce the participant to the entry method. During the first session, the prompts given to participants consisted of single words ordered in small sets designed to obtain complete coverage of all available characters. The second session progressed to simple phrases with at most four words and a maximum word length of six characters. The phrases given to participants in the second session were also pruned to remove any that would require use of the N-Best list. In the third session, participants still entered simple phrases, but they were introduced to the N-Best list feature. For example, in the

context of the Session 3 sample prompt shown in Table 2.1, the tap sequence for 'fine' initially recognizes the word 'done'. There were no restrictions on the prompts that participants entered in all subsequent sessions.

Session	Sample Reference Text
1	leaves
2	are you there
3	should be fine
4	he says he has some ideas
6	so you're ignoring me
8	this is the crew

Table 2.1
Sample prompts given to users by session

2.3 Results

We measure text entry quality using 3 metrics. First we use Character Error Rate (CER). We measure this as the minimum number of insertions, deletions, and substitutions required to obtain the reference text, divided by the number of characters in the reference text, multiplied by 100%. Participants had an average 4.05% CER in Session 1, which improved to 2.08% CER averaged across the final 3 sessions. Figure 2.3 shows that most participants obtained much better accuracy after initially struggling in Session 1.

Next, we measure entry rate in words per minute (WPM), taking each word to be

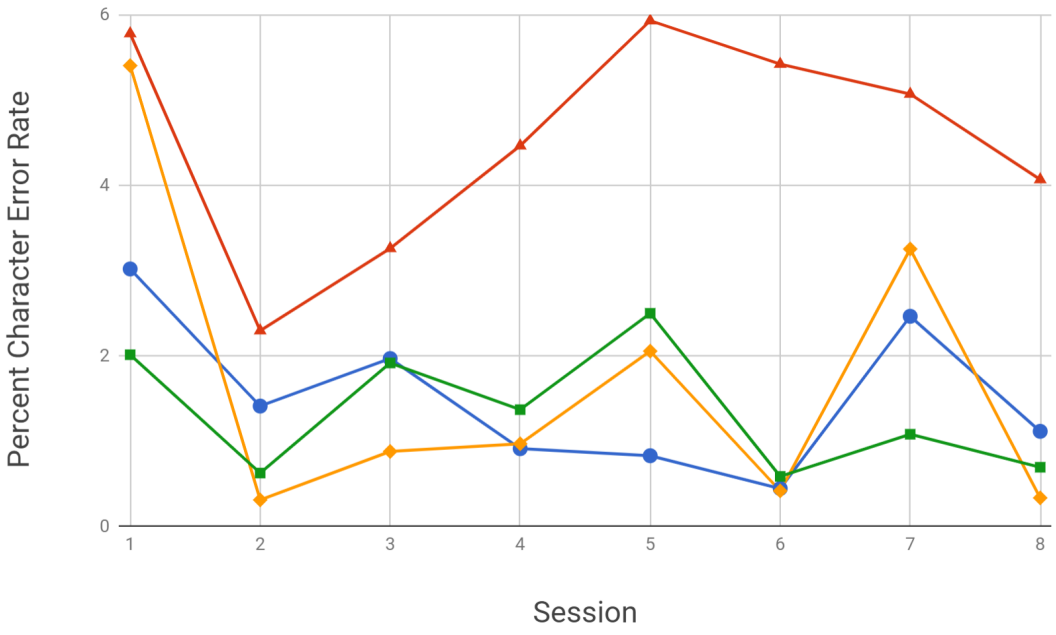


Figure 2.3: Character Error Rate by session using Tap123

5 characters, including a space. Time was measured from the start of the first tap until participants closed the keyboard, indicating they were finished. As shown in Figure 2.4, participants averaged 4.32 WPM in Session 1, and achieved an average 19.09 WPM across the final 3 sessions.

Finally, we measure Backspaces per Tap (BPT) by dividing the total number of taps deleted by the total number of taps entered. BPT allows us to measure the accuracy of participants' initial taps instead of the accuracy of the final entered text. Participants averaged 0.199 BPT in Session 1, but as shown in Figure 2.5, remained relatively stable at 0.068 average BPT for the remainder of the sessions.

We excluded from all of our analysis a total of 25 phrases that participants expressed

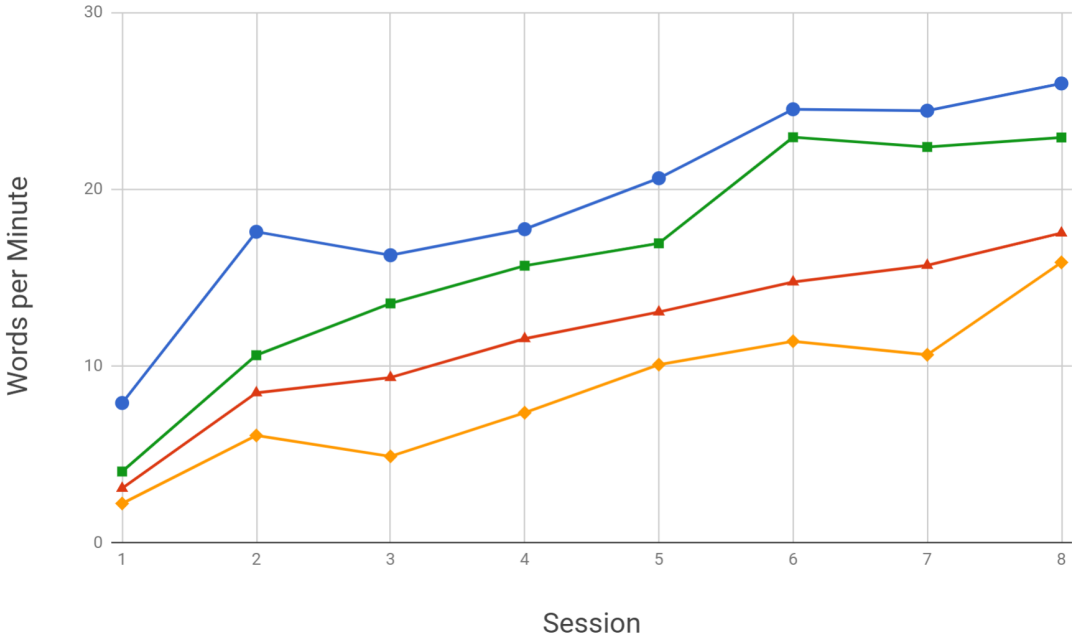


Figure 2.4: Entry rate by session using Tap123

difficulty hearing or spelling during entry. We did this to more accurately represent the performance of the interface, since in these cases the issue was rooted in the transcription task itself and not in the entry method; in a real-world application the user will know what they are trying to type. Excluded prompts frequently included proper nouns, such as in 'did you mean oxley'.

When asked about unnatural interactions after the conclusion of their final session, participants noted that "Closing the keyboard was always an issue for me", and "Finding the correct word because my suggested words are above keyboard on phone."

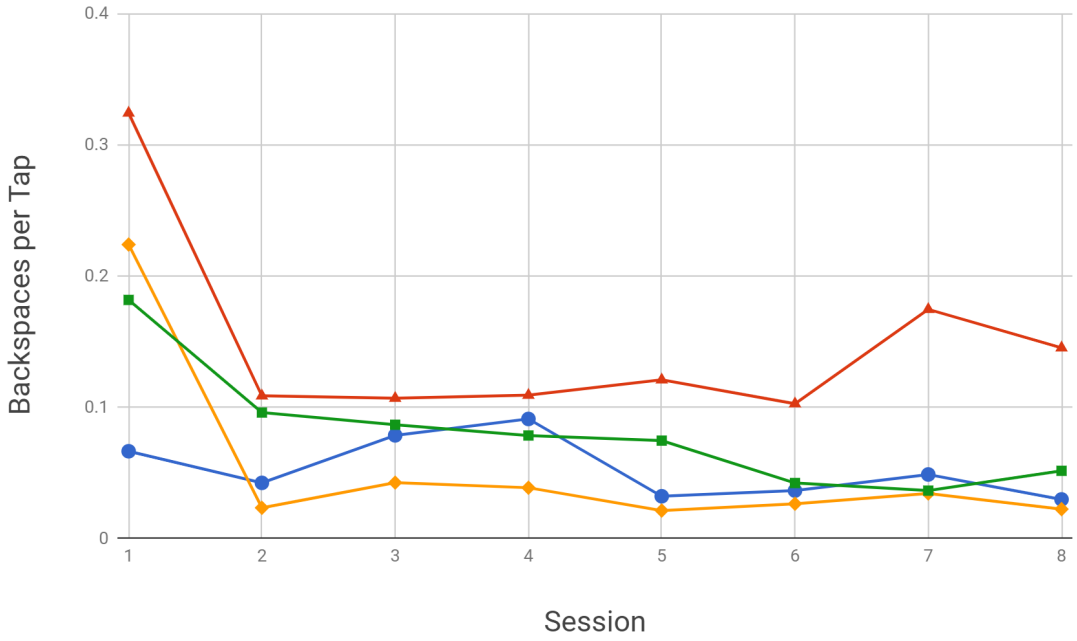


Figure 2.5: Backspaces per Tap by session using Tap123

2.4 Discussion

To further improve the Tap123 interface, we will make some changes in response to these participant comments. We will improve the gesture to close the keyboard to make it easier and more consistent, and, in Chapter 4, we will explore ways to provide suggested words to users with simultaneous audio feedback. We found that in 28.1% of instances where participants backspaced, they backspaced multiple times in a row, and each of these multi-character backspaces averaged 4.4 characters. Because of this, we will also give users the ability to delete an entire word at once at any point during entry.

Additionally, we noticed during the user study that several common words had identical tap sequences and the disambiguation algorithm was not always able to determine the correct one. This lead to users needing to slow down and verify each word was correct before moving on, or go back and delete the word if they began the next word before realizing the recognition was incorrect. In the next chapter, we will explore ways to optimize the character groupings that aim to reduce the frequency of these collisions instead of simply using groupings based on the Qwerty layout.

Chapter 3

Keyboard Optimization

In this chapter we will explore different ways to optimize an ambiguous keyboard for use in an eyes-free text entry interface. The primary goals of this line of work are to 1) determine if it is feasible to reduce the number of ambiguous groups from six to four, and 2) examine the disambiguation performance decrease from constraining the groupings alphabetically. Reducing the number of groups would allow users to enter text using a single hand, as opposed to Tap123's two-handed technique. This is more suitable for truly mobile text entry as users can then hold the device with their other hand. A four-group approach would also enable us to remove location dependency completely, with the user simply tapping with between one and four fingers, inclusive, to indicate the groups. Past work shows that constraining the groupings alphabetically makes them easier to use without extensive training. We

will combine ideas from past work on this topic as well as introduce novel ideas to create multiple proposed groupings that are likely to create more efficient and accurate interface.

3.1 Related Work

3.1.1 N-Opt Algorithm

There has been a considerable amount of past research done on the topic of optimizing keyboard layouts for a variety of metrics such as finger travel distance, keystroke efficiency, and tap clarity. Several papers introduce pieces of information that are directly relevant to this particular application. The first of these papers, written by Lesher et al. [12], shows that fully enumerating and evaluating every possible set of groupings is computationally infeasible, but proposes an algorithm that can be used to efficiently find locally optimized groupings in ambiguous keyboards. The first step in their algorithm is to compute a confusability matrix for some corpus of English text. They do this by stepping through the corpus one character at a time and creating a list of the most likely characters to come next based on a character prediction algorithm. They keep track of how frequently an incorrect character is predicted as more likely than the actual next character in the text. The authors define the mutual

confusability of two characters α and β as

$$M(\alpha, \beta) = C(\alpha, \beta) + C(\beta, \alpha), \tag{3.1}$$

where $C(\alpha, \beta)$ is the number of times α was mistaken for β, and $C(\beta, \alpha)$ is the reverse. Further, for any number of characters placed in a single ambiguous group, the total mutual confusability is the sum of the mutual confusabilities between each pair of characters.

Once the confusability matrix is computed, Lesher et al. [12] run their n-opt algorithm. The algorithm starts with a valid arrangement of characters into groupings. On each pass, it checks every possible tuple of n characters to see if any way of shuffling those characters will result in a better overall arrangement according to whatever metric is being optimized (in this case, the confusability of the groupings). For example, a two-opt pass would check every pair of characters ('AB', 'AC', ..., 'YZ') and a three-opt pass would check every triple ('ABC', 'ABD', ..., 'XYZ') in alphabetical order. If any swaps are made during the course of a single pass, the pass repeats once it has finished, checking all tuples again. The algorithm continues until a pass completes with no swaps being made. The n-opt algorithm requires factorially increasing computations for higher values of n; the highest pass completed by Lesher et al. was five-opt [12]. Since the n-opt algorithm only finds local optima, the authors start

with many initial arrangements and run the two-opt algorithm. They take the result with the best performance and then run the five-opt algorithm to further improve it.

3.1.2 Constrained Layouts

An alternative approach to keyboard optimization is the use of genetic algorithms proposed by Gong and Tarasewich [10]. In this paper, the authors compared an ambiguous layout that was constrained to be in strict alphabetical order with a layout that was unconstrained and freely optimized. Both layouts were optimized to minimize the number of keystrokes required on average to enter a character of input. While they were able to fully enumerate the possible constrained layouts, they used genetic algorithms to optimize the unconstrained layout. They first generated a random population of layouts from which to start. Each successive generation of layouts was the product of reproduction, crossover, and mutations from the prior generation. Details on how these were performed were not provided. The authors reported that this genetic algorithm was quite successful at locating optimal or near-optimal layouts. The authors also found that the constrained layout aided users' ability to learn the interface since it was more familiar.

Other authors found similar results when comparing freely optimized keyboards to ones with familiarity constraints. For example, Bi et al. [2] performed a study with a

Qwerty keyboard, a Quasi-Qwerty keyboard, and a freely optimized keyboard. The Quasi-Qwerty and freely optimized keyboards were designed to minimize the travel distance for a user's finger, thereby increasing the entry rate. However, the Quasi-Qwerty layout had the constraint that letters could not move more than one row and one column from their initial Qwerty position. The authors found that while the Quasi-Qwerty layout had an improved movement efficiency from the standard Qwerty layout, it was not as efficient as the freely optimized layout. However, during user trials, the authors found that users took the longest to locate the initial letter of a word on the freely optimized keyboard, followed by the Quasi-Qwerty layout, and finally the standard Qwerty layout. The authors concluded that the Quasi-Qwerty layout was effective at obtaining an increased movement efficiency while not sacrificing too much time in the initial visual search. While through practice users would improve with the freely optimized layout, using more familiar layouts reduces the amount of practice needed.

3.1.3 Multi-Parameter Optimization

Instead of simply enforcing a strict Qwerty restriction on their non-ambiguous keyboard layout, Dunlop and Levine [8] chose to optimize their keyboard layout on three different parameters: finger travel distance, tap ambiguity, and familiarity.

- They calculated finger travel distance by measuring the distance between each pair of letters, multiplying it by the number of times those letters were adjacent in the language corpus, and summing those products for a given layout. While this metric is relevant to many on-screen optimization problems and is common in related work, it does not apply to an ambiguous keyboard that is not location-dependent.

- Dunlop and Levine also calculated a metric on tap ambiguity to reduce the number of adjacent characters in a layout that could frequently be swapped with each other to create valid words (e.g. I and O can be swapped between the words 'for' and 'fir'). They first created a table of these commonly interchanged letters, which they referred to as bad bigrams, or "badgrams". After counting the frequency of badgrams in same-length words in a text corpus, they converted each to a probability by dividing by the total number of badgram occurrences. The authors defined their tap clarity metric for a given keyboard layout as

$$M_{tap\_clarity} = 1 - \sum_{\forall i,j \in \alpha} \begin{cases} p_{ij} & \text{if } neighbors_{ij} \\ 0 & \text{else} \end{cases}, \tag{3.2}$$

  where p_{ij} is the badgram probability for letters i and j and $neighbors_{ij}$ is true if i and j are adjacent in the layout.

- The final metric Dunlop and Levine used in their optimization was familiarity.

They calculated the similarity of a given layout to the Qwerty layout by summing the squared distances between each key's position and its Qwerty position and then normalizing the results to the range between 0 and 1. This allowed for potentially high-scoring layouts that had most letters near their Qwerty positions with a few exceptions that may have been restricted by the Quasi-Qwerty constraints.

With these three metrics, Dunlop and Levine [8] used Pareto front optimization to find candidate layouts. The algorithm keeps track of the set of *Pareto optimal* layouts, which are those that for all other layouts in the set, there is no layout that is better on all three metrics. They defined a small set of initial layouts that were then taken through 2000 iterations of changes. At each iteration, one key was swapped and there was a 25% chance of swapping an additional key after each swap. The metrics were then recalculated and the layout was either added to the Pareto front or discarded. After the final Pareto front was formed, the authors selected the keyboard nearest the 45° line (the line with all metrics equal), which they determined was the best overall layout given the metrics.

Qin et al. [16] also used a Pareto front to perform optimization, but they did so in two dimensions and with an ambiguous keyboard. They defined their clarity metric for a given word as the frequency of that word in the corpus divided by the total frequency of identical tapping sequences given an ambiguous layout. They then defined the

clarity of a layout as the sum over all words of the product of word frequency and word clarity. The second metric used in this work was a typing speed metric based on the relative location of frequent character combinations. As with the previous paper, this is not relevant to interfaces that remove location dependency. Instead of choosing the layout closest to the 45° line in the Pareto front as done by Dunlop and Levine [8], Qin et al. opted to select the layout that had the highest average of normalized metrics. Another difference between these works was that instead of optimizing on a third metric that indicated similarity to Qwerty, Qin et al. enforced a strict adherence to the Qwerty ordering of characters and split each row into three ambiguous groups, creating a total of nine groups.

3.2 Procedure

To explore the feasibility of reducing the number of groups, we optimized both four-group (T4) and six-group (T6) layouts. The T6 layouts can be used in the same manner as in Tap123 in Chapter 2. This will also enable us to directly compare our optimized layouts to the Qwerty-based groupings used initially. While only having four groups of characters may make disambiguation more difficult in the T4 layouts, it would allow one-handed input to be performed more easily.

Combining some of the ideas explored in past work, we developed the following procedure.

3.2.1 Corpus Analysis

First, we performed an analysis on the corpus of text written on mobile devices gathered and released by Vertanen and Kristensson [22]. To perform this analysis, we adapted the ideas of Lesher et al. [12]. While they iterated through the corpus one character at a time, predicting the most likely next character, we iterated through the first 100,000 phrases in the training set one word at a time. We used software based on the VelociTap decoder [23] with an n-gram character model and an n-gram word model to predict the most likely word given the preceding words in the phrase. To simplify the process, we worked under the assumption that a simulated user had entered the correct number of characters on a single-group ambiguous keyboard containing the letters A-Z and apostrophe.

After sending the decoder the ambiguous sequence of groups (in this case all the same group) for each word, we queried it for the top 100 predictions for each word, storing each word deemed more likely than the true word under the models. We used these mispredicted words to build a character-level confusion matrix for the corresponding letters between the incorrect and correct words. For example, if 'hello' was predicted

before true word ‘world’, the matrix entries for the character pairs ‘HW’, ‘EO’, ‘LR’, and ‘OD’ would be incremented.

3.2.2 Performance Metrics

During the optimization process, we need a way to compare one layout to another to determine the better of the two. To accomplish this, we developed two metrics to evaluate the potential performance of a given layout.

The first metric is badgram clarity. Using the confusion matrix we generated during our corpus analysis, we calculated the mutual confusability between each pair of characters, as Lesher et al. did [12]. We divided each of these by the total sum, as done by Dunlop and Levine [8], to obtain badgram probabilities p_{ij}:

$$p_{ij} = \frac{C(i,j) + C(j,i)}{\sum_{\forall i,j|i \neq j} C(i,j) + C(j,i)}, \tag{3.3}$$

where $C(i,j)$ is the number of times i was mistaken for j in the corpus analysis (element i,j in the confusability matrix). We then adapted Dunlop and Levine’s tap clarity formula to define our badgram clarity metric as

$$Clarity_{badgram} = 1 - \sum_{\forall i,j | i \neq j} \begin{cases} p_{ij} & \text{if } sameGroup(i,j) \\ 0 & \text{else} \end{cases}, \tag{3.4}$$

where $sameGroup(i, j)$ is true if i and j are on the same ambiguous group.

We also define a second clarity metric with which we can evaluate our layouts, word error rate (WER) clarity. To calculate WER clarity, we utilize the mobile test set from the corpus released by Vertanen and Kristensson [22]. We first define the layout that we are evaluating as an ambiguous keyboard in the VelociTap decoder [23], specifying which characters are in each group. We then iterate through each word in the first 250 phrases in the test set. We simulate the taps that would be needed to type that word on the current layout. We then use the decoder to find the probability of each word that fits the ambiguous group sequence given the words in the phrase to the left of the current word. We look at the most probable word returned by the decoder, and determine if it predicted the true word correctly. We aggregate all of the predictions to define WER clarity as

$$Clarity_{WER} = 1 - \frac{\sum_{\forall word} correct(word)}{n_{words}}, \tag{3.5}$$

where $correct(word)$ returns 1 if the decoder's prediction matches the true word and 0 otherwise, and where n_{words} is the total number of words in the first 250 phrases of

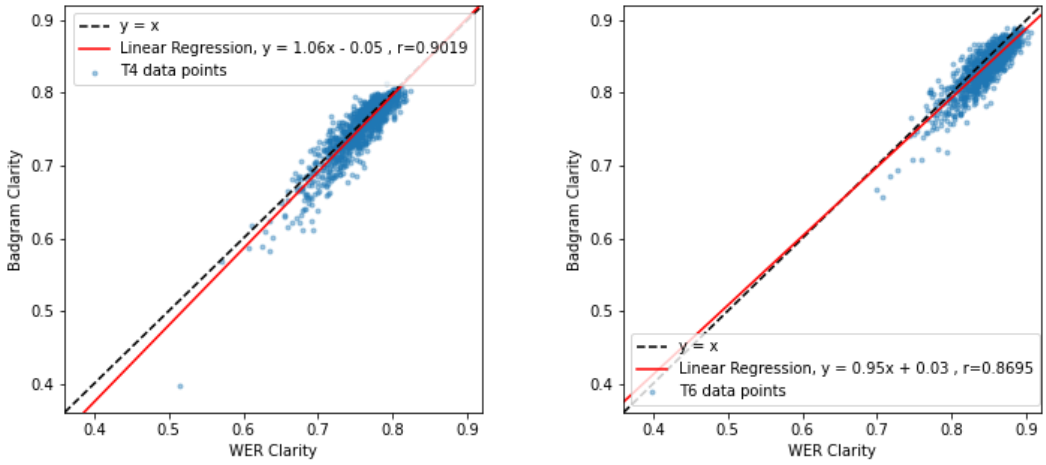
the mobile test set.

Badgram clarity is a more simplified metric and is likely less representative of the collisions a user will face when using a layout in practice. However, since WER clarity requires us to iterate through a test phrase set to make predictions, it becomes computationally expensive to calculate for large numbers of layouts.

To determine how similar the two metrics are, we randomly generated 1,000 T4 and 1,000 T6 layouts. We calculated both clarity metrics on each layout and fit a linear regression model to the data. The results can be seen in Figure 3.1. Both the T4 layouts and the T6 layouts produced results that were near the line $Clarity_{badgram} = Clarity_{WER}$ with relatively high correlation ($r = 0.9019$ and $r = 0.8695$ in the T4 and T6 layouts, respectively). From this, we can conclude that while not perfect, badgram clarity is still a good estimate of a layout's performance.

3.2.3 Unconstrained Optimization

We began by optimizing layouts freely, without any familiarity constraints. The goal of this line of optimization was to determine the upper bound of performance that can be achieved, even if it takes additional training time for users. Since, as Lesher et al. [12] showed, this optimization problem is computationally infeasible, we used a similar n-opt approach to their work. In their work, when they examined the



(a) T4 Layouts

(b) T6 Layouts

Figure 3.1: Scatter plots of different metrics on 1000 random layouts. The dashed line represents $Clarity_{badgram} = Clarity_{WER}$. The solid red line represents the best fit linear regression.

possible swaps between a set of characters, they only mentioned directly swapping the characters. However, since it could be the case that not all groups have an equal number of characters, we added some additional comparisons.

For example, let Group A contain α (among other characters) and let Group B contain β (among other characters). For a two-opt pass, we checked the layout where we simply swapped α and β as Lesher et al. did. However, we also checked the layouts where both α and β were in the same group, either both in Group A or both in Group B.

To implement this, we first generated the set of all n-tuples in our set of characters

(where n is the n in n-opt). For each tuple, we then generated all possible permutations (e.g. for tuple (ab), we generated (ab, ba). Then, for each permutation, we examined all possible partitions into k groups (allowing empty groups), where k was the number of groups that contained any character in the permutation for the current groupings (e.g. if a, b, and c were each in a separate group, $k = 3$). An example of all partitions generated on the permutation (ab) is $(|ab,\ a|b,\ ab|)$. Note that there are $k-1$ dividers to create k groups. To increase efficiency, we tracked each partition created for each given tuple, and did not add duplicates to the final list of groupings. If we generated partitions on the permutation (ba) after already doing (ab), we would generate $(|ba,\ b|a,\ ba|)$. The first and last partitions are identical to the first and last partitions already generated from the permutation (ab), since ordering within each group does not matter (i.e. $(|ab)$ is equivalent to $(|ba)$). In this situation only the middle partition, $(b|a)$ would be added to our final list.

3.2.3.1 Badgram Clarity

We first optimized using badgram clarity. We ran our version of the two-opt algorithm on 50,000 random initial layouts, as Lesher et al. [12] did. We then ran our version of the five-opt algorithm on the best result from two-opt. We did this separately for T4 and T6 layouts.

3.2.3.2 WER Clarity

As we discussed previously, the WER clarity metric is computationally expensive and requires iteration through the test phrases to check each layout. This, combined with the factorial growth of the n-opt algorithm prevented us from running a large number of trials and from running anything larger than two-opt. Using WER clarity, we ran 50 random initial layouts through our two-opt algorithm. As with the badgram clarity, we did this separately for T4 and T6 layouts.

3.2.4 Constrained Optimization

Our next optimization task was to add a familiarity constraint. In Chapter 2, we used groupings based on the Qwerty keyboard. However, as we move towards a fully location-independent technique, Qwerty-based groupings will likely have less of a positive impact on performance. Instead, in this section, we will focus on alphabet-based groupings. While Gong and Tarasewich [10] implemented a strict alphabetical constraint, we will allow for a small number of characters to shift outside of alphabetical order.

Constraining the layout in this way drastically reduces the number of possibilities, which allowed us to fully enumerate and compare all of them. We first laid out the

alphabet from A-Z with apostrophe on the end. We then used a similar partition function that we used on our unconstrained layouts, but this time we did not allow any groups to be empty, as moving a letter from a group where it is alone will never increase performance. For a T4 layout, there are 2,600 possible partitions (3 dividers over 26 positions = ${}_{26}C_3$), and for a T6 layout there are 65,780 (${}_{26}C_5$). For each partition, we selected every combination of s characters, where c was the maximum number of characters allowed to move from their group. We tested values of s from 0 to 3 inclusive on both T4 and T6 layouts and $s = 4$ on only T4 layouts due to computational feasibility. We then compared every possible way to shuffle the s characters in each combination. While our shuffles in the n-opt algorithm only allowed characters to move to groups occupied by another member of the combination, here characters could move to any group. The number of possible combinations was ${}_{27}C_s$, which for $s = 2$ was 351, for $s = 3$ was 2,925, and for $s = 4$ was 17,550. The number of possible shuffles for each combination was equal to t^s, where t was the number of groups in the layout. The total number of possible layouts was then equal to:

$$Layouts \;=\; {}_{26}C_{t-1} \times \; {}_{27}C_s \times t^s. \tag{3.6}$$

Due to this once again combinatoric growth, we were only able to complete this optimization using the badgram clarity metric.

3.3 Results

Using these algorithms and procedures, we produced the following layouts.

3.3.1 Unconstrained

We will first explore the layouts optimized using badgram clarity and then proceed to those optimized using WER clarity.

3.3.1.1 Badgram Clarity

For the T4 layout, the best achieved badgram clarity was 0.8191. There were 208 of the 50,000 initial layouts that when run through our two-opt algorithm produced final layouts that matched this exact clarity. While the order of the groups and the order of the characters within each group varied between these, alphabetizing each group and set of groups showed that these layouts were all identical. This layout was:

$$(a, h, k, n, s, x), (b, e, f, g, j, m, q, u), (c, d, o, t, v, z, '), (i, l, p, r, w, y)$$

When we ran this layout through the five-opt algorithm, there were no swaps made

in the entire pass.

We found similar results in the two-opt optimization of the T6 layout. There were 3,509 of the 50,000 initial layouts that converged to a single layout with a badgram clarity of 0.9067.

$$(a, d, l, q), (b, e, j, m), (c, g, k, o, w, z), (f, i, s, x, '), (h, n, p, v, y), (r, t, u)$$

Once again, when we ran this layout through the five-opt algorithm, no swaps were made. As we expected, the peak clarity of the T6 layout was much higher than that of the T4 layout. One observation that we made on these groupings was that wherever possible, vowels were placed into different groups. In the T6 layout, each vowel, even including y, occupied a separate group. In the T4 layout, there were not enough groups to separate all of the vowels, so e and u as well as i and y were forced to share, but every group contained a vowel.

3.3.1.2 WER Clarity

While we had hoped to see multiple random initial groupings converge to the same optima as we had with the badgram clarity metric, this was not the case. For both the T4 and T6 layouts we optimized, the best clarity was only achieved by a single layout. With T4 layouts, we were able to achieve a peak WER clarity of 0.8542 with

the following groups:

$$(a, d, f, h, k, q, y, '), (b, c, e, i, j, n, x), (g, l, o, s, v, w), (m, p, r, t, u, z)$$

For the T6 layouts, the best WER clarity was 0.9346, which was achieved by this set of groups:

$$(a, k, m, w, y), (b, l, q, s, v, z, '), (c, e, f, j, x), (d, i, n, p), (g, o, r), (h, t, u)$$

We can note the same observation for these sets of groups that we did for those optimized on badgram clarity: the vowels are separated wherever possible. We computed both clarity scores for these four layouts, as well as for the Qwerty-based layout used in Tap123. For easier comparison, the results are shown in the upper portion of Table 3.1.

3.3.2 Constrained

We created a total of five constrained layouts: three T4 and two T6. The difference between each layout was the number of characters that we allowed to stray from alphabetical order. The best groupings are shown in Table 3.2, with swapped characters in bold:

Constraint	Groups	Optimization Metric	Swaps	Badgram Clarity	WER Clarity
Qwerty	6	-	-	0.8427	0.8418
None	4	Badgram	-	**0.8191**	0.8209
None	4	WER	-	0.7990	**0.8542**
None	6	Badgram	-	**0.9067**	0.9106
None	6	WER	-	0.8955	**0.9346**
Alphabetical	4	Badgram	0	0.8030	0.7872
Alphabetical	4	Badgram	1	0.8112	0.8046
Alphabetical	4	Badgram	2	0.8130	0.8093
Alphabetical	4	Badgram	3	0.8136	**0.8170**
Alphabetical	4	Badgram	4	**0.8148**	0.8143
Alphabetical	6	Badgram	0	0.8923	0.8874
Alphabetical	6	Badgram	1	0.8960	0.8886
Alphabetical	6	Badgram	2	0.8984	0.8975
Alphabetical	6	Badgram	3	**0.9007**	**0.8994**

Table 3.1
Both clarity scores for the top unconstrained and constrained layouts and the Qwerty-based layout from Chapter 2. The best performance on each metric for each combination of constraint and groups is in bold.

While these layouts were all optimized based on their badgram clarity, both clarity scores for the final layouts are presented in the lower portion of Table 3.1. For the T4 layouts, there were only very small gains in badgram clarity for each additional character allowed to move after the first. When the WER clarity was measured, it actually decreased when a fourth character was allowed to switch groups due to the differences between the two metrics. If we had been able to optimize using the WER clarity, this would not have been the case. For the T6 layouts, each additional character allowed to move contributed to small gains on both clarity metrics. When compared to the Qwerty-based T6 groupings, all alphabetically-constrained T6 layouts had much higher clarity metrics, but none of the T4 layouts were particularly

Swaps	Group 1	Group 2	Group 3	Group 4
0	a, b, c, d, e	f, g, h, i, j, k, l, m	n, o, p, q, r	$s, t, u, v, w, x, y, z,'$
1	a, b, c, d, e	f, g, h, i, j, k, l, m	$n, o, p, \mathbf{s}$	$q, r, t, u, v, w, x, y, z,'$
2	a, b, d, e, f	$\mathbf{c}, g, h, i, j, k, l, m$	$n, o, p, \mathbf{s}$	$q, r, t, u, v, w, x, y, z,'$
3	a, b, d, e, f	$\mathbf{c}, g, h, i, j, k, l$	$m, n, o, \mathbf{s}, \mathbf{w}$	$p, q, r, t, u, v, x, y, z,'$
4	a, b, e, f, g	$h, i, j, \mathbf{n}, \mathbf{s}$	$\mathbf{c}, \mathbf{d}, k, l, m, o$	$p, q, r, t, u, v, w, x, y, z,'$

Swaps	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
0	a, b, c, d	e, f, g, h	i, j, k, l, m	n, o, p, q	r, s	$t, u, v, w, x, y, z,'$
1	a, b, c, d	e, f, g, h	i, j, k, l, m	n, p, q, r	$\mathbf{o}, s$	$t, u, v, w, x, y, z,'$
2	a, c, d	$\mathbf{b}, e, f$	g, h, i, j, k, l	$m, n, \mathbf{r}$	o, p, q, s	$t, u, v, w, x, y, z,'$
3	a, b, c, d	e, f	g, h, j, k, m, n	o, p, q, s	$\mathbf{l}, t, u, v$	$\mathbf{i}, \mathbf{r}, w, x, y, z,'$

Table 3.2
The best constrained layouts (by badgram clarity) found by fully enumerating the search space. T4 groups are shown in the upper table and T6 groups are in the lower table. Characters swapped from alphabetical order are in bold.

close. In all of the T6 constrained layouts produced here, the five core vowels were all separated, though one of them was paired with y in each layout.

3.4 Discussion and Limitations

Optimizing for badgram clarity, our T4 and T6 layouts were able to achieve scores of 0.8191 and 0.9067, respectively. For comparison, the badgram clarity score of the Qwerty-based layout used in Chapter 2 was 0.8427. While our T6 layout was able to beat this, the T4 layout was not. From this, we would expect to encounter more collisions using our freely optimized T4 groupings than we did with the initial Qwerty-based T6 groupings. However, the badgram clarity metric does not fully represent the

collisions we could expect to occur. Using the WER clarity metric, the Qwerty-based T6 groupings registered a score of 0.8418. In this case, both our T4 and T6 freely optimized layouts were able to outperform the baseline previous groupings, showing that it is feasible to decrease from six groups to four.

For the alphabetically constrained layouts, the gains seen in the clarity metrics for each additional character swapped were quite marginal. Since each swap would make the groups more difficult for a user to remember, it is our opinion that it would not be worth this additional cost to move any characters out of their alphabetical order.

From our observations on the separation of vowels in both our freely optimized and constrained layouts, we conjecture that one of the factors leading to the poor performance of the Qwerty-based T6 groupings was the fact that four of the vowels ($y, u, i,$ and o) were located in a single group.

Seeing such a large number of random starts ultimately converge to the same layout in the badgram-based two-opt algorithm supports the hypothesis that we were able to find the global optima for the badgram clarity metric. While we cannot prove this without fully enumerating all layouts, the fact that no additional swaps were made by the five-opt algorithm adds further support to this claim.

One of the two main limitations in these studies was the computational power. Ideally, we would have been able to perform unconstrained WER-based two-opt on more

initial layouts in hopes of showing the convergence we did with the badgram clarity. Additionally, we were unable to use WER clarity at all in our constrained optimizations since we were fully enumerating the search space. It is possible that we would see more gains with increased numbers of character swaps, especially in the T4 layouts. If we were optimizing on WER clarity, it would have been impossible for the metric to decrease with an additional character swap. However, over the course of nearly a day of computing, our algorithm was unable to complete even a single partition of a T4 or T6 constrained optimization with only two character swaps. Finally, additional computing power or better disambiguation time efficiency would have allowed us to optimize a constrained T6 layout with four character swaps.

The other limitations are the assumptions we make on user input. The user input is simulated and assumed to be perfect, which is not the case in practice. To fully explore the performance of these layouts, we need to test them with real users, which we plan to do in future work. We are also assuming that user input is complete. In these experiments, we did not explore the possibility of users selecting suggested words in the middle of entering a word. To best explore this possibility, we could also add an optimization metric for keystroke savings that calculates the number of keystrokes needed for the true word to be the most likely word suggested by the decoder.

Chapter 4

Audio Word Suggestions

4.1 Motivation

In our initial study of the Tap123 interface in Chapter 2, participants noted difficulty selecting their intended word from the list of likely words returned by the decoder. Our work in the previous chapter aimed to increase the likelihood of predicting the correct word. However, the correct word is still not a guarantee and there will still be instances that require users to select a word other than the one initially returned.

Additionally, suggested words and word completions could be offered with each keystroke prior to recognition, or if the user pauses. If the user hears the word they are typing they could select it immediately, without finishing the remainder

of the taps. This would operate similar to many commercially available standard touchscreen keyboards, but using audio instead of visual feedback.

One benefit of these commercial touchscreen keyboards is that they place a number of word suggestions, commonly three or four, just above the top row of keys. This allows the user to quickly glance and check them without distracting too much from the task of typing. However, if these words are read sequentially using standard text-to-speech software, it would take quite some time to finish; the user would likely be able to type the next key or the remainder of the word by the time the list was finished being read.

To attempt to solve this problem, we will investigate the possibility of providing the user with audio-only word suggestions using simultaneous voices. This would enable the interface to provide word suggestions much quicker and help to further increase the efficiency of this entry technique. In this work, we will begin with a perceptual study to measure user performance listening for a target word among several potentially similar words. We want to determine if they are able to hear the word and if they are able to identify which voice said it. Based on the results, we will then be able to integrate this suggestion technique into a text entry interface to see how it impacts users' text entry performance.

4.2 Related Work

There have been several studies on the human ability to perceive speech amid other speech, the first dating back to Cherry in 1953 [6]. Cherry conducted a series of experiments on the *Cocktail Party Effect*, with the aim of identifying the factors that contributed to the phenomenon. The results of the studies showed that participants were able to clearly listen to either of two spoken messages played into opposite ears via headphones (each was recorded by the same speaker). However, participants were not able to notice when the language of the message in the ear that they were not listening to (the "rejected" ear) changed from English to German. This prompted further studies, in which the author found that participants were able to correctly identify if the rejected message was normal human speech (as opposed to reversed speech or an oscillating tone), as well as whether it was a male or female voice. They were not, however, able to identify any words or phrases in the rejected message or what language it was in.

This line of work was continued by Egan et al. [9], who played sentences simultaneously for participants. Each of these sentences contained one of two 'call signs', and the participants were instructed to write down the words following a particular call sign, ignoring the sentence that contained the other call sign. In one experiment,

the authors' results indicated that high-pass filtering of either message allowed participants to better understand the target message. Another experiment confirmed the results found by Cherry [6] that participants were better able to understand the target message when the two messages were played in opposite ears.

Brungart [5] conducted experiments using phrases of a set structure. They contained one of eight call signs, one of four colors, and one of eight numbers. Two phrases were played simultaneously, and participants were instructed to listen for the phrase with a particular call sign and report the corresponding color and number. The author varied the speakers of the target phrase and the masking phrase, finding that participants were best able to distinguish the two when the speakers were of different sexes. Participants performed the worst when the same speaker was used for both the target and masking phrases. Further work by Darwin et al. [7] attribute this performance difference to the combination of differences in vocal tract length and fundamental frequency between speakers.

Brungart and Simpson [4] experimented with two, three, and four speakers in different spatial arrangements around the listener. They used digital signal processing and head-related transfer functions to "move" the stimuli in space around the users' heads. They used the same phrase structure (call sign, color, number) that Brungart did in their earlier work [5]. Their results showed that participants scored 92% with two speakers, 72% with three speakers, and 62% with four. Additionally, they found that

participants scored higher in spatial configurations where the speakers were more spread out, and specifically when the target phrase came from one of the extremes (left or right as opposed to middle).

Guerreiro and Gonçalves [11] adapted this work to the field of accessible computing, testing how well blind people could understand concurrent speech using screen readers. They used both different-sex voices as well as spatial separation to increase the users' ability to understand the phrases. The authors also adjusted the speech rates of the voices with the aim of presenting as much information to the user as quickly as possible (while still maintaining comprehension). They found that using two or three voices with slightly faster speech rates had a larger information bandwidth than using a single voice with a much faster speech rate. They reported that the best combination of comprehension and speed was using two voices at 1.75 or 2 times the default speech rate.

4.3 Study 1

While these past studies have explored various factors that contribute to the clarity of simultaneous speech, they have all done so using phrases or full sentences. When designing a text entry interface, we are more concerned with the clarity of single words. Additionally, the problem is slightly more difficult because word suggestions

tend to be similar to one another than randomly selected words would be (i.e. if a user has begun to type a word, several of the suggestions will likely start with that prefix).

To determine how well users were able to comprehend single words among other simultaneous words, we developed the following study. Participants were asked to wear headphones for the duration of the study. In each trial, users were given a target word. They then listened to between two and four voices each speak a word and selected which voice, if any, they thought spoke the target word. All voices used originated from the Android Operating System's built in text-to-speech service:

- Voice 1 — English US, Male 1
- Voice 2 — English US, Female 1
- Voice 3 — English US, Male 3
- Voice 4 — English US, Female 2

The there were an equal number of trials in each condition with two, three, and four voices, but the order in which they occurred was randomized. There were three conditions:

- 180Degree — Voice 1 and Voice 2 played entirely in the participants' left ears, while Voice 3 and Voice 4 played entirely in the participants' right

ears (creating 180 degrees of separation between each pair). For trials with two voices, VOICE 1 and VOICE 4 were used. Trials with three voices also used VOICE 2. The aim of this ordering was to maximize both pitch and spatial differences between playing voices. All voices began playing simultaneously.

- 60DEGREE — VOICE 1 and VOICE 4 remained entirely on the left and right, respectively. VOICE 2 was located 30 degrees to the left of center and VOICE 3 was 30 degrees to the right of center (creating 60 degrees of separation between each voice). For VOICE 2 and VOICE 3, the location was simulated by splitting the audio into left and right channels, adjusting the volume of each, merging then back together, and then normalizing the result to the same total volume as the other voices. The individual channel volumes were adjusted by factors of 0.67 on the left and 0.33 on the right for VOICE 2, and the opposite for VOICE 3. The same ordering of voices was used as in 180DEGREE and all voices began playing simultaneously.

- SEQUENTIAL — All voices were located directly in front of the participant (equal volume in both ears). Voices were added and played in numerical order as described above (i.e. if there were only two voices, VOICE 1 and VOICE 2 were used). Each voice waited for the previous voice to finish completely before beginning to speak.

The trials were based on phrases taken from the Enron mobile data set [21]. To create

the words spoken for each trial, a random phrase was chosen from the set, along with a random character position in that phrase. We fed the portion of the phrase up to and including that position into the VelociTap decoder [23] (including any partial word). The n-best word completions returned by the decoder (the n words with the highest probability given the context and any prefix) became the words spoken by each voice, where n was the number of voices in that particular trial. It was not possible that the random position was the beginning of the sentence since the decoder would just return the most likely words to start a sentence. If there were not enough words returned by the decoder to fill every voice in a trial a new phrase and position were chosen. We felt that this would be most representative of a user receiving audio word predictions in a real-world text entry task. The partially completed word or next word (if the position fell between words) became the target word. Once the words were selected, which voices spoke them were randomized so that VOICE 1 was not always the most likely word returned by the decoder. It is important to note that the target word was not always present in the words that were spoken, as would be the case in a text entry task.

4.3.1 Procedure

We developed a web application to conduct this study remotely. The application first walked participants through informed consent and a demographic questionnaire.

It then provided instructions for the first of the three counterbalanced conditions; the ordering of conditions was predetermined based on participant number. Included with the instructions was a button for each voice, as shown in Figure 4.1. When clicked, these buttons played text-to-speech for 'Voice N', where N was the number of the voice clicked. This text-to-speech was synthesized using the corresponding voice, and was spatially located in its position for the current condition. Participants were required to click all four buttons before they were allowed to proceed, but they were able to listen to each voice as many times as they wanted. This instruction page was included at the start of each condition.

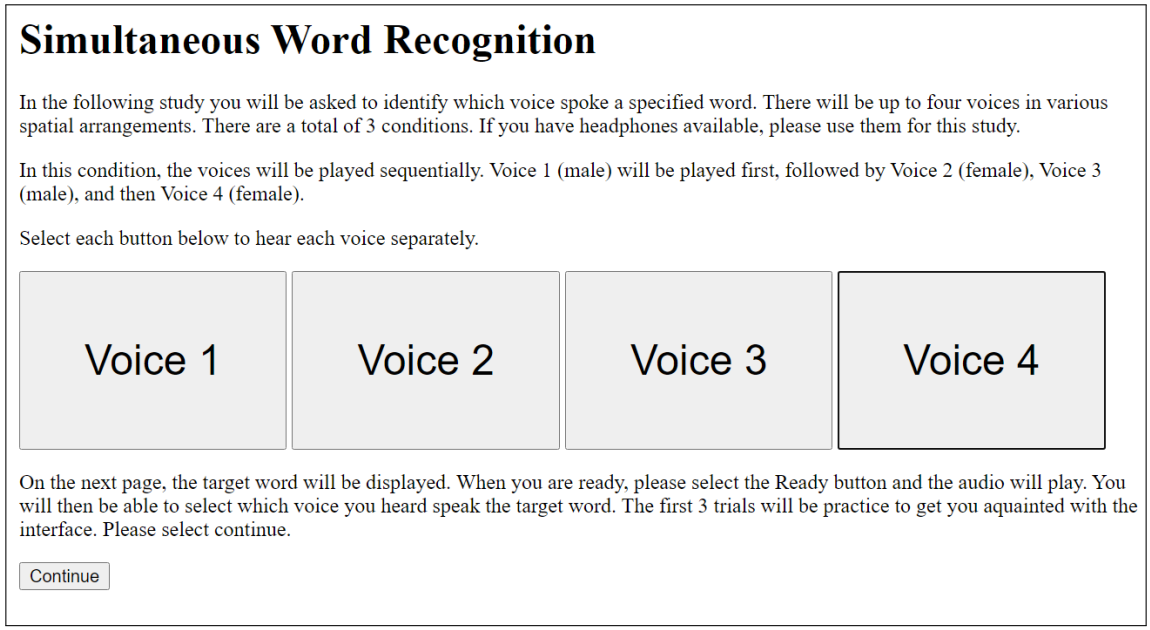


Figure 4.1: The web interface at the beginning of the SEQUENTIAL condition. The text below the Voice buttons as well as the Continue button do not appear until all four Voice buttons have been clicked. This screen is meant to familiarize users with which voice corresponds to which number.

After listening to each voice on the instruction page, participants then progressed to

six practice trials. They were first shown the target word (see Figure 4.2). When they clicked a 'Ready' button, the audio for the trial was played and they were shown six buttons, as shown in Figure 4.3. These corresponded to 'Not Present', the four voices, and 'I Don't Know'. If they answered correctly, they proceeded to the next trial. Otherwise, the correct response was shown (Figure 4.4) and they repeated the practice trial until they responded correctly. The purpose of this was to ensure that participants were familiar with the voices that were being presented. Practice trials were not included in any of the data analysis. Once the practice trials were complete, participants then completed 24 evaluation trials. The interface was identical to that used in the practice trials, but participants were not able to retry any incorrect trials. Participants then completed a brief questionnaire before repeating this process, beginning with the instructions page for the next condition.

4.3.2 Results

We recruited 24 participants by word-of-mouth. A total of 5 participants did not fully complete the study due to technical issues. We removed their data and recruited new participants to obtain a total of 24 participants. They were each paid $10 for their participation. Participants ranged from 18–25 years of age. 9 identified as male, 14 as female, and 1 as other. None of the participants reported being blind or low vision.

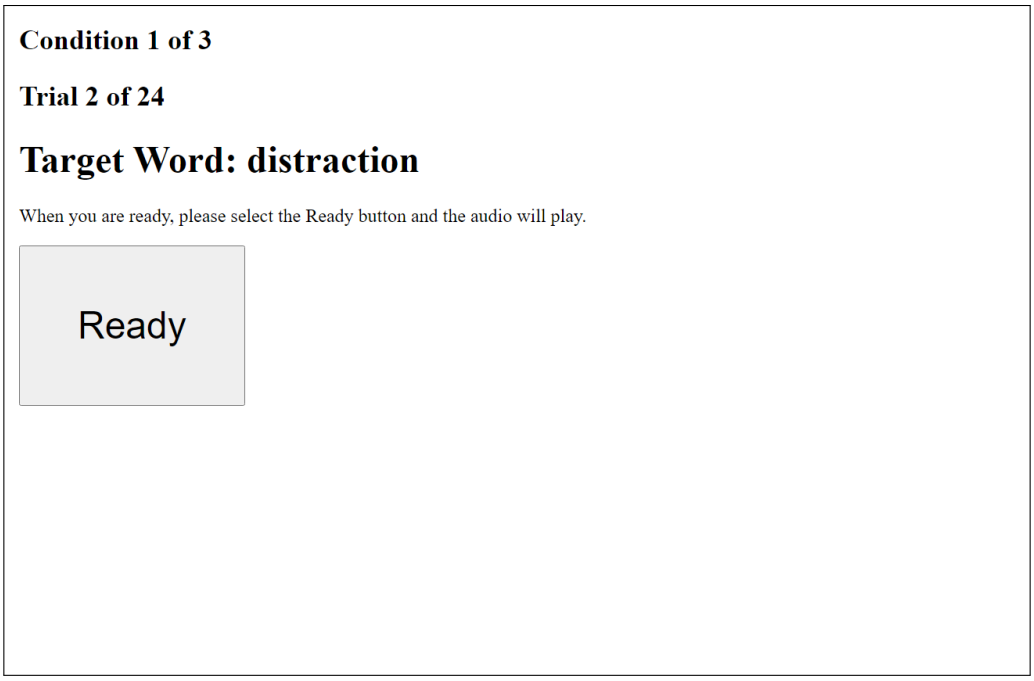


Figure 4.2: The web interface at the beginning of a trial. It displays the target word and allows the participant to prepare to listen. The audio will play once they click 'Ready'.

There were two independent variables in this study: spatial arrangement and number of voices, and one dependent variable: accuracy. We computed accuracy as the number of correct responses divided by the number of total responses. The mean accuracy for each combination of spatial arrangement and number of voices can be seen in Figure 4.5. Mauchly's sphericity test showed that both main effects met the assumption of sphericity ($W = 0.96, p = 0.66$ for spatial arrangement and $W > 0.99, p = 0.99$ for number of voices). Additionally, the interaction also met the assumption of sphericity ($W = 0.82, p = 0.90$), so we did not need to correct the degrees of freedom. A type III ANOVA for our 3 (spatial arrangement) $\times$ 3 (number of voices) design showed a significant difference between spatial arrangements, $F(2, 46) = 83.80$, $p < .001$, as

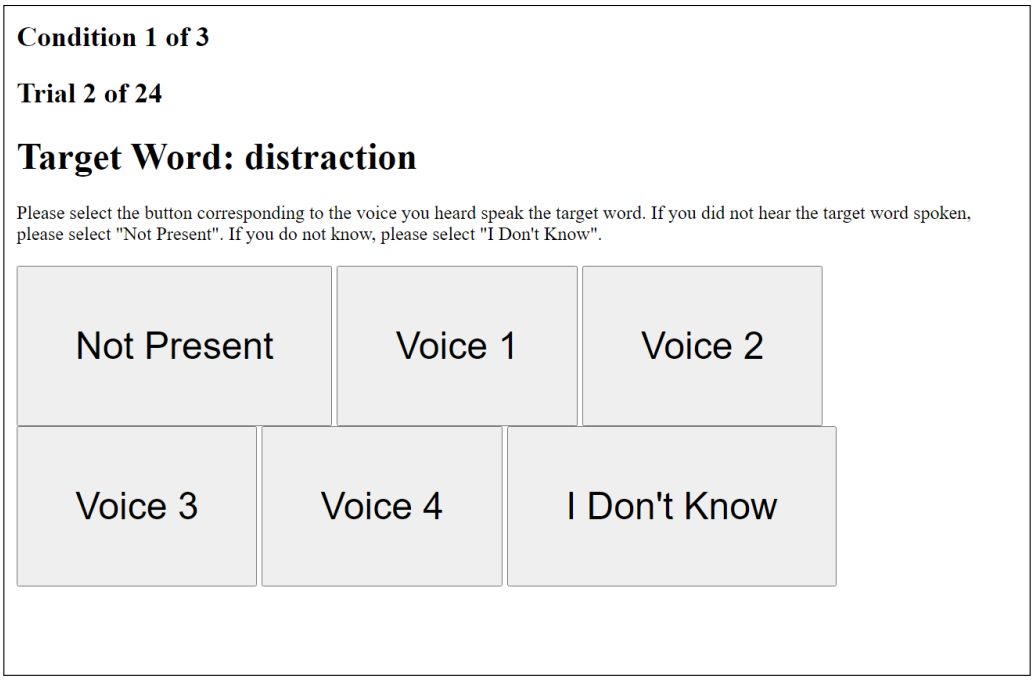


Figure 4.3: The web interface once the participant has clicked 'Ready' and the audio has played. The participant now selects which voice they heard say the target word.

well as number of voices, $F(2, 46) = 102.35$, $p < .001$. More importantly, it showed a significant interaction between our independent variables, $F(4, 72) = 19.38$, $p < .001$. This indicates that the number of voices had a different effect on participants' accuracy depending on the spatial arrangement used, so we did not summarize within each independent variable.

We ran the pairwise *post hoc* t-tests with Bonferroni correction to determine where exactly the significant differences were found. The results of these tests are shown in Table 4.1. Notably, within both 180Degree and 60Degree the difference between each number of voices was significant, but none of these differences were significant

Figure 4.4: The web interface after the participant has indicated their response. Clicking 'Continue' will take them to the next trial.

within SEQUENTIAL. Between the two simultaneous conditions, 180DEGREE and 60DEGREE, all differences were significant only where the number of voices was not equal. Finally, the difference between each simultaneous condition with two voices and SEQUENTIAL with all numbers of voices was not significant, but three and four voices within each simultaneous condition were significantly different from SEQUENTIAL with all numbers of voices.

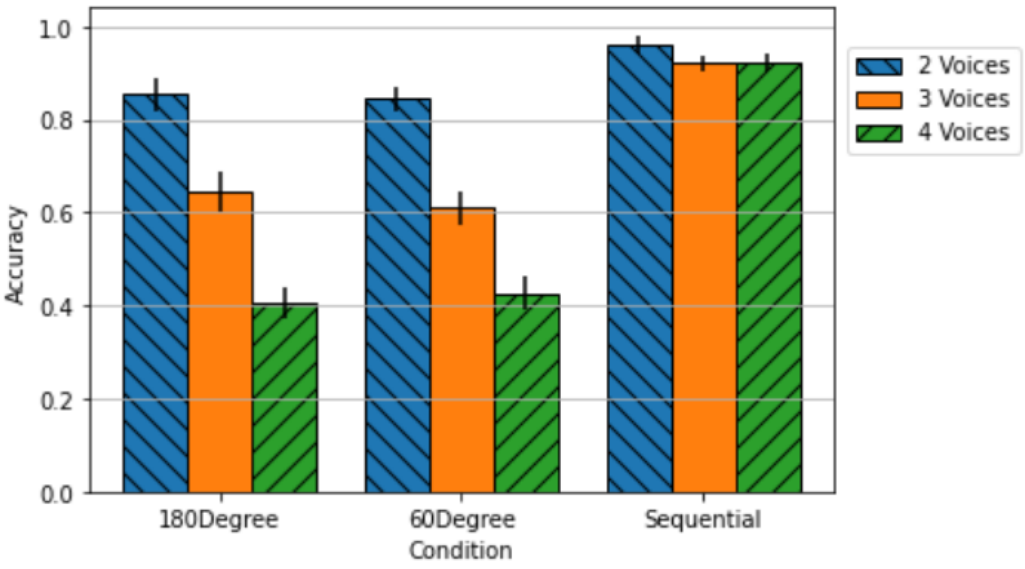


Figure 4.5: The average accuracy of participants in Study 1. Error bars represent standard error of the mean.

	Condition	180Degree			60Degree			Sequential	
Condition	**Voices**	**2**	**3**	**4**	**2**	**3**	**4**	**2**	**3**
180Degree	**3**	**.002**	-	-	-	-	-	-	-
	4	**<.001**	**<.001**	-	-	-	-	-	-
60Degree	**2**	1.00	**.010**	**<.001**	-	-	-	-	-
	3	**<.001**	1.00	**.007**	**<.001**	-	-	-	-
	4	**<.001**	**.010**	1.00	**<.001**	**.024**	-	-	-
Sequential	**2**	.536	**<.001**	**<.001**	.055	**<.001**	**<.001**	-	-
	3	1.00	**<.001**	**<.001**	1.00	**<.001**	**<.001**	1.00	-
	4	1.00	**<.001**	**<.001**	.786	**<.001**	**<.001**	1.00	1.00

Table 4.1
The p-values from pairwise post hoc t-tests in Study 1. Bold values indicate a significant difference ($p < 0.05$).

4.3.3 Discussion

This first study showed that performance was quite poor in both simultaneous conditions when there were more than two voices, while performance did not significantly change for the control SEQUENTIAL condition. We were hoping to have success similar to past work by creating both spatial and pitch differences in our speakers. However, we conjecture that the difference arose from the length and similarity of the stimuli. All of the previous studies we examined earlier in this chapter used sentence or phrase stimuli that may have allowed participants to adjust to hearing multiple speakers. Additionally, many previous studies, like those by Brungart [5] and Brungart and Simpson [4], used a very finite set of words in their phrases that were not easily confusable. This was not the case in our study, and that may have also contributed to the poorer performance observed in the simultaneous conditions.

4.4 Study 2

We wanted to further explore audio word suggestions to see if we could provide more than two suggestions. We devised another experiment that introduced a small amount of temporal spacing between words in the simultaneous conditions. This would still allow a series of suggestions to be presented to the user quicker than fully sequential,

but with potentially better comprehension than fully simultaneous. We used the same voices and trial creation process as in the Study 1. All conditions used the same spatial arrangement as the Study 1's 60Degree condition. Voices were also added in the same order as 60Degree to maximize spatial separation (Voice 1, Voice 4, Voice 2, then Voice 3). The conditions in this study were as follows:

- DelayShort - Voices began playing in numerical order. After each voice began, there was a 0.15 s delay before the next voice began playing.
- DelayLong - Voices began playing in numerical order. After each voice began, there was a 0.25 s delay before the next voice began playing.
- Sequential - Voices played in numerical order. Each voice waited for the previous voice to fully finish before beginning to play.

4.4.1 Procedure

This study used the same web interface as Study 1. Participants completed informed consent, a demographic questionnaire and received instructions before completing six practice trials and 24 evaluation trials in each condition. There was also a short questionnaire after each condition. In this study, we recruited 96 participants by posting Human Intelligence Tasks (HITs) on Amazon Mechanical Turk, a crowdsourcing platform. They were paid $3.50 for their time. Similar to the crowdsourcing procedure

used by Vertanen et al. [20], we eliminated workers that failed to meet a control standard. The standard we chose was 50% accuracy in the SEQUENTIAL (control) condition. We returned rejected HITs to the pool until we had 96 participants that met the standard. No participant was able to complete more than one HIT in this study, and, to our knowledge, no participants had previously completed Study 1 since we recruited from different pools for each study.

4.4.2 Results

The 96 participants in this study ranged from 21 to 69 years of age. 68 identified as male, and the remaining 28 as female. A total of three participants identified as low vision. Eight of the participants reported using some form of screen reading software either on mobile or desktop devices.

In this study, our independent variables were temporal arrangement and number of voices, and our dependent variable was still accuracy. The mean accuracy results broken down by each combination of temporal arrangement and number of voices can be seen in Figure 4.6. As in the previous study, Mauchly's sphericity test showed that both main effects met the assumption of sphericity ($W = 0.99, p = 0.65$ for temporal arrangement and $W = 0.99, p = 0.64$ for number of voices). Additionally, the interaction also met the assumption of sphericity, $W = 0.86, p = 0.11$, so once

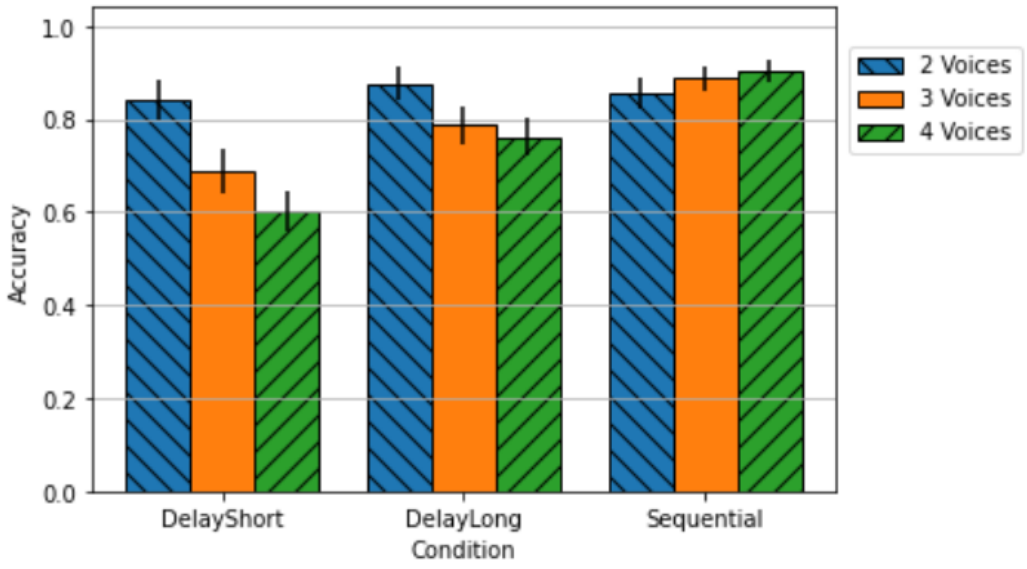


Figure 4.6: The average accuracy of participants in Study 2. Error bars represent standard error of the mean.

again we did not need to correct the degrees of freedom. A type III ANOVA for our 3 (temporal arrangement) $\times$ 3 (number of voices) design showed a significant difference between temporal arrangement, $F(2, 190) = 78.42$, $p < .001$, and between number of voices, $F(2, 190) = 37.60$, $p < .001$. Again, there was a significant interaction between our independent variables, $F(4, 380) = 30.13$, $p < .001$, showing that the number of voices had a different impact on accuracy for different temporal arrangements.

We ran the pairwise *post hoc* t-tests with Bonferroni correction to locate the significant differences. These results are presented in Table 4.2. Interestingly, mean accuracy increased slightly with number of voices in the SEQUENTIAL condition, though none of these differences were significant. In the DELAYLONG condition, participants' accuracy with two voices was significantly higher than with three or four, but there was

no significant difference between three and four voices. In the DelayShort condition, accuracy significantly decreased with each voice added. The most important difference from Study 1 was that three voices in the DelayLong condition was not statistically significant from two voices in Sequential. While this does not necessarily show that they are the same, it provides a focal point for further exploration.

	Condition	**DelayShort**			**DelayLong**			**Sequential**	
Condition	**Voices**	**2**	**3**	**4**	**2**	**3**	**4**	**2**	**3**
DelayShort	**3**	**<.001**	-	-	-	-	-	-	-
	4	**<.001**	**.042**	-	-	-	-	-	-
DelayLong	**2**	1.00	**<.001**	**<.001**	-	-	-	-	-
	3	.228	**.002**	**<.001**	**<.001**	-	-	-	-
	4	**.004**	.063	**<.001**	**<.001**	1.00	-	-	-
Sequential	**2**	1.00	**<.001**	**<.001**	1.00	.063	**.004**	-	-
	3	1.00	**<.001**	**<.001**	1.00	**<.001**	**<.001**	1.00	-
	4	**.015**	**<.001**	**<.001**	1.00	**<.001**	**<.001**	.255	1.00

Table 4.2
The p-values from pairwise post hoc t-tests in the Study 2. Bold values indicate a significant difference ($p < 0.05$).

4.4.3 Discussion

Adding the slight delay before starting to play each subsequent voice produced much more favorable results than the spatial and pitch distance alone, especially for greater numbers of voices. While we did not directly compare the DelayShort and DelayLong conditions to the 60Degree condition from Study 1, the mean accuracies were much higher–60.2% and 76.0% with four voices in DelayShort and DelayLong,

respectively, compared to 42.7% in 60Degree from the Study 1. Most importantly, as noted at the closing of the Results section, DelayLong with three voices did not have a statistically significant difference from Sequential with two voices. This shows that we are able to provide an additional suggested word in less time without significantly decreasing performance.

4.5 Limitations and Future Work

While the results shown in these studies (particularly the latter) are promising, they are still only a simplification of a real-world text entry task. They allowed participants to focus all of their attention on listening to the suggestions for the target word. In a more complete task, users attention will be split between not only that, but also the content that they are writing and performing the input itself. An additional limitation is that in these studies, participants were instructed to wear headphones. This may not always be practical, which could impair their ability to hear separate left and right channels. As future work, we plan to incorporate audio word suggestions into a full text entry task. Based on the results of this study, we will compare DelayLong with three voices to Sequential with three voices. We will also compare these to a standard screen reader approach where participants are able to explore elements on the screen, including three word suggestions. This study will inform design choices for the most efficient and practical way to present word suggestions.

Chapter 5

Conclusions

While we have not yet developed a fully functional, release-ready interface, we were able to explore several of the key components that will help to make one possible.

5.1 Discussion

In Chapter 2, we found success with our non-Braille, mostly location-independent, ambiguous keyboard. Users were able to achieve an entry rate of 19.09 words per minute with a 2.08% error rate after eight hours of practice. The main things that we took away from this study were the abundance of disambiguation errors and the need for word suggestions. These two needs fueled the studies that we conducted in

Chapters 3 and 4 as we sought to improve our interface in every way possible.

In Chapter 3, we sought to improve this performance by optimizing the layout to reduce disambiguation errors. We conducted offline experiments to optimize both four-key (T4) and six-key (T6) layouts, both with and without alphabetical constraints. Our T6 layouts (both constrained and unconstrained) were able to easily out-score our previous Qwerty-based layout on both clarity metrics. Our constrained T4 layouts did not perform quite as well, but one of the unconstrained T4 layouts was able to best the T6 Qwerty layout on the WER clarity metric. This showed that it is feasible to reduce the number of keys to four, which would, in our opinion, make it much more practical for mobile use. It would also allow us to easily remove the small location dependence that was present in the original Tap123 interface. We could shift our group mappings to rely only on number of fingers while still enabling users to enter text with a single hand. While a constrained layout would aid users in learning the group mappings, our T4 constrained layouts had a lower clarity score than the Qwerty layout we used previously.

In Chapter 4, we explored different ways to present word suggestions to users using only audio feedback. We found that users had quite poor performance with more than two fully simultaneous voices. However, if we added a 0.25 s delay before starting each subsequent voice, users performed about as well with three voices as with two fully sequential voices. This creates a compromise between the speed of fully simultaneous

suggestions and the accuracy of fully sequential suggestions.

5.2 Future Work

While we were able to explore layout optimization and audio word suggestions, there was a third area of improvement for the Tap123 interface that we were unable to explore in this document. While the disambiguation algorithm we used contained probabilistic elements in its search, it treated each tap as deterministic. Past work with the VelociTap decoder [23] allowed the decoder to insert, delete, or substitute tap events within its search. Hypotheses that were a result of these modifications incurred a penalty to their overall likelihood, which reduced their chances of being selected as the most likely result. This allows users to make mistakes during text entry and continue on, as long as the mistakes are not too severe, which can further increase entry rate. To implement this, we would need to gather data from users entering text on our final keyboard design and then perform offline experiments to tune the decoder penalties associated with insertions, deletions, and substitutions. This may take several iterations of tuning and testing, as users' behaviors and ability with the keyboard will change as they become more familiar with it and as our model improves.

The next step in our layout optimization is to perform user trials. Since the goal is to

ultimately use only four groups, we will compare the freely optimized T4 layout that scored higher than the T6 Qwerty to the constrained T4 layout that allowed three characters to be out of alphabetical order. Of our constrained T4 layouts, this one scored the highest on the WER clarity metric, which is likely to be more representative of in-practice disambiguation collisions. This will be a between-subjects longitudinal study in which we will track participants' performance over time to determine any differences in learning the two layouts as well as peak performance.

For our audio word suggestions, we now will need to place them into a full text entry task. We use three word suggestions in each condition and compare the presentation methods of a 0.25 s delay and fully sequential to a control condition similar to a commercial screen reader. We will evaluate the trade-offs between entry and error rate, how frequently suggestions are used, and how frequently incorrect suggestions are selected. We will perform this study with a standard keyboard interface, as opposed to our novel interface, to be able to isolate the effect of the word suggestions.

Once we have concluded our investigations into each of these components, we will be able to combine them into a single interface, which we can evaluate as a whole and compare to other eyes-free text entry methods.

References

[1] Azenkot, S., Wobbrock, J. O., Prasain, S., and Ladner, R. E. Input finger detection for nonvisual touch screen text entry in perkinput. In *Proceedings of Graphics Interface 2012* (CAN, 2012), GI '12, Canadian Information Processing Society, p. 121–129.

[2] Bi, X., Smith, B. A., and Zhai, S. Quasi-qwerty soft keyboard optimization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2010), CHI '10, Association for Computing Machinery, p. 283–286.

[3] Bonner, M. N., Brudvik, J. T., Abowd, G. D., and Edwards, W. K. No-look notes: Accessible eyes-free multi-touch text entry. In *Pervasive Computing* (Berlin, Heidelberg, 2010), P. Floréen, A. Krüger, and M. Spasojevic, Eds., Springer Berlin Heidelberg, pp. 409–426.

[4] Brungart, D., and Simpson, B. Cocktail party listening in a dynamic multitalker environment. *Perception & Psychophysics 69*, 1 (2007), 79–91.

[5] Brungart, D. S. Informational and energetic masking effects in the perception of two simultaneous talkers. *The Journal of the Acoustical Society of America 109*, 3 (2001), 1101–1109.

[6] Cherry, E. C. Some experiments on the recognition of speech, with one and with two ears. *The Journal of the Acoustical Society of America 25*, 5 (1953), 975–979.

[7] Darwin, C. J., Brungart, D. S., and Simpson, B. D. Effects of fundamental frequency and vocal-tract length changes on attention to one of two simultaneous talkers. *The Journal of the Acoustical Society of America 114*, 5 (2003), 2913–2922.

[8] Dunlop, M., and Levine, J. Multidimensional pareto optimization of touchscreen keyboards for speed, familiarity and improved spell checking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2012), CHI '12, Association for Computing Machinery, p. 2669–2678.

[9] Egan, J. P., Carterette, E. C., and Thwing, E. J. Some factors affecting multi-channel listening. *The Journal of the Acoustical Society of America 26*, 5 (1954), 774–782.

[10] Gong, J., and Tarasewich, P. Alphabetically constrained keypad designs for text entry on mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2005), CHI '05, Association for Computing Machinery, p. 211–220.

[11] Guerreiro, J., and Gonçalves, D. Faster text-to-speeches: Enhancing blind people's information scanning with faster concurrent speech. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility* (New York, NY, USA, 2015), ASSETS '15, Association for Computing Machinery, p. 3–11.

[12] Lesher, G. W., Moulton, B. J., and Higginbotham, D. J. Optimal character arrangements for ambiguous keyboards. *IEEE Transactions on Rehabilitation Engineering 6*, 4 (1998), 415–423.

[13] Mascetti, S., Bernareggi, C., and Belotti, M. TypeInBraille: Quick eyes-free typing on smartphones. In *Computers Helping People with Special Needs* (Berlin, Heidelberg, 2012), K. Miesenberger, A. Karshmer, P. Penaz, and W. Zagler, Eds., Springer Berlin Heidelberg, pp. 615–622.

[14] National Federation of the Blind Jernigan Institute. The braille literacy crisis in america. `https://nfb.org//sites/www.nfb.org/files/images/nfb/documents/pdf/braille_literacy_report_web.pdf`, 2009.

[15] Oliveira, J., Guerreiro, T., Nicolau, H., Jorge, J., and Gonçalves, D. BrailleType: Unleashing braille over touch screen mobile phones. In *Proceedings of the 13th IFIP TC 13 International Conference on Human-Computer Interaction - Volume Part I* (Berlin, Heidelberg, 2011), INTERACT'11, Springer-Verlag, p. 100–107.

[16] Qin, R., Zhu, S., Lin, Y.-H., Ko, Y.-J., and Bi, X. Optimal-T9: An optimized T9-like keyboard for small touchscreen devices. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces* (New York, NY, USA, 2018), ISS '18, Association for Computing Machinery, p. 137–146.

[17] Southern, C., Clawson, J., Frey, B., Abowd, G., and Romero, M. An evaluation of BrailleTouch: Mobile touchscreen text entry for the visually impaired. In *Proceedings of the 14th International Conference on Human-Computer Interaction with Mobile Devices and Services* (New York, NY, USA, 2012), MobileHCI '12, Association for Computing Machinery, p. 317–326.

[18] Tinwala, H., and MacKenzie, I. S. Eyes-free text entry with error correction on touchscreen mobile devices. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries* (New York, NY, USA, 2010), NordiCHI '10, Association for Computing Machinery, p. 511–520.

[19] Vertanen, K. Counting fingers: Eyes-free text entry without touch location. In *CHI '16: Extended Abstracts of the the ACM Conference on Human Factors*

in Computing Systems (May 2016).

[20] Vertanen, K., Fletcher, C., Gaines, D., Gould, J., and Kristensson, P. O. The impact of word, multiple word, and sentence input on virtual keyboard decoding performance. In *CHI '18: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2018).

[21] Vertanen, K., and Kristensson, P. O. A versatile dataset for text entry evaluations based on genuine mobile emails. In *MobileHCI '11: Proceedings of the 13th International Conference on Human-Computer Interaction with Mobile Devices and Services* (2011).

[22] Vertanen, K., and Kristensson, P. O. Mining, analyzing, and modeling text written on mobile devices. *Natural Language Engineering 27* (2021), 1–33.

[23] Vertanen, K., Memmi, H., Emge, J., Reyal, S., and Kristensson, P. O. VelociTap: investigating fast mobile text entry using sentence-based decoding of touchscreen keyboard input. In *CHI '15: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2015), pp. 659–668.