A Text Entry Mechanism for Blind Users with Small Handheld Touch Enabled Mobile Devices

A report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Integrated Master of Science

in

Mathematics and Computing

by

Himanshu Mishra

(14MA20014)

Under the supervision of

Prof. Chandal Nahak and Prof. Debasis Samanta

Department of Mathematics

Indian Institute of Technology Kharagpur

Spring Semester, 2018-19

April 30, 2019

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: April 30, 2019 (Himanshu Mishra)

Place: Kharagpur (14MA20014)

DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled **"A Text Entry Mechanism for Blind Users with Small Handheld Touch Enabled Mobile Devices"** submitted by **Himanshu Mishra** (Roll No. 14MA20014) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Integrated Master of Science in Mathematics and Computing is a record of bona fide work carried out by him under my supervision and guidance during Spring Semester, 2018-19.

Date: April 30, 2019
Place: IIT Kharagpur

Prof. Chandal Nahak
Department of Mathematics
IIT Kharagpur

Prof. Debasis Samanta
Department of Computer Science
and Engineering, IIT Kharagpur

Abstract

Name of the student: **Himanshu Mishra** Roll No: **14MA20014**

Degree for which submitted: **Integrated Master of Science**

Department: **Department of Mathematics**

Thesis title: **A Text Entry Mechanism for Blind Users with Small Handheld Touch Enabled Mobile Devices**

Thesis supervisor: **Prof. Chandal Nahak and Prof. Debasis Samanta**

Month and year of thesis submission: **April 30, 2019**

Mobile phones are now touch-enabled, which allow on-screen keyboards for text entry. However, people with visual impairment find it difficult to adapt to on-screen keyboard and become digitally compromised. To bridge this gap, a text entry mechanism has been proposed in Section 2.3 using the concept of directional flick gestures suitable for blind people. In Section 2.1 and 2.2 we studied that the unguided directional flick gestures provide better results than the guided directional gestures. A text entry mechanism – VectorEntry has been proposed, which uses eight distinct unguided directional flick gestures to select a character in the keyboard on a touch-enabled mobile phone whose design is in accordance with the traditional 4x3 telephone keypad. In Section 2.4, we present our approach to improve the text entry rate. We have discussed multiple cases and problems in Text Correction and Text Prediction and its integration in the VectorEntry system.

Acknowledgements

I thank Prof. Debasis Samanta for providing me the opportunity and idea to work on the project. He gave me freedom to pursue my ideas at my own pace without any pressure of deadlines. I could not have imagined having a better advisor and mentor for this project.

I also thank Prof. Chandal Nahak for his constant support and guidance.

I want to especially mention *Akash Chandra (16MA20006)*, third year student of Department of Mathematics and thank him for his hard work on the implementation of Prediction and Correction methods for the system.

In conclusion, I also recognize that this project would not have been possible without the support from Department of Computer Science and Engineering, IIT Kharagpur. Many thanks to all those who made this project possible.

Himanshu Mishra

April, 2019
Indian Institute of Technology, Kharagpur

Contents

Chapter 1

Introduction and Motivation

Touch-enabled mobile phone devices trim the obligation for keyboard and account for the growing popularity and marketability of the technology. It is predicted that it will be going to occupy a very large market space in near future [11]. Although, these devices are gaining popularity day-by-day, blind users find it very difficult to cope up with touch devices. This is because accessing devices through touch screens is a visually demanding procedure and it seems to be an impossible task for people with visual impairment. From the societal point of view, needs of blind people cannot be neglected. As per the recent report from WHO [2], about 285 million people in the world are visually impaired (either completely blind or low vision) . It is reported that 87% of the visually impaired are from the developing countries [1].

Although there are some initiatives known to provide support to the blind people like voice recognition for speech to text is a natural solution that works eye-free and without any involvement of hands, nevertheless, it has several limitations. It may lead to situational impairment, it is difficult to handle the practical challenges of adopting accents and intonations of an individual, background noise, and a huge computational overhead in a mobile setting [10].

1.1 Existing Solutions

A number of research works have been done toward the accessibility support to the blind people [14]. These works address different challenges from various directions and provide assistive supports accordingly. In this section, a brief survey limited to the works relevant to the text entry techniques with the mobile devices has been presented. All these works broadly can be classified into four categories: a) tactile feature based solutions, b) transformed Braille keypad based solutions, c) voice-based solutions, d) gesture-based solutions and d) transformed physical keypad design based solutions. The major works in these categories are briefly surveyed in the following.

1.1.1 Tactile feature based solutions:

As a simple technique, a tactile feature marked on a specific key are used to access a character on the keyboard with audio feedback. The marked key can be identified by the sense of touch in the jumble and accordingly it can help to identify other keys by realizing the associated relative placement of the surrounding keys in the keypad. Both the QWERTY and 43 telephonic layouts are supported with the tactile identifier to assist visually impaired people. These identifiers are placed either on 'F', 'K' or 'G' key in QWERTY layout and placed on '5' key in the telephonic keypad.

1.1.2 Transformed Braille keypad based solutions

Researchers also have tried to transform the notion of Braille into the touch screen based devices. Oliveira et al. [Oliveira et al. 2011b] presented BrailleType, where six dots of Braille are presented as six cells on the device screen. Areas of the Braille cells are large and mapped at the corners and edges of the screen to provide spatial

ease to find. Users need to select all the corresponding Braille cells representing a target letter and then a confirmation for entry. Users must touch on the desired Braille cell and wait for audio confirmation cue to get the cell selected and thus after selecting the required cells, a double tap anywhere on the screen will select the target letter. BrailleTouch [Frey et al. 2011], [Romero et al. 2011], [Southern et al. 2012], BrailleEnter [Alnfiai and Sampalli 2017], Hybrid-Brailler [Trindade et al. 2018], BrailleSketch [Li et al. 2017], etc. are some text-entry mechanism, which have been recently introduced based on six dot-based Braille [Braille 1829] script. Here, users need six fingers to operate the system where each finger is assigned as a single dot; different combi- nations of fingers are used to represent different alphabets. Users hold the device facing display outside from them and can place their finger holding the device by two hands. Now, a combination of touched fingertips on the screen represents a target letter and that letter will get entered by releasing the fingertips together from the screen.

1.1.3 Voice-based solutions

Several voice-based solutions are known to provide accessibility features in mobile devices for blind users. For example, Google introduced Voice- Action [7] for Android device [3]; Apple introduced SIRI , which is a voice-enabled application on iPhone 4s and higher; Apple's VoiceOver [4] can provide audio feedback on touching a widget on the screen. Nuance TALK [Nuance 2014] is an audio-based application software for blind people to access many functionalities of a feature phone.

1.1.4 Gesture-based solutions

With the rapid advancement of touch-screen technology and its growing popularity, a number of gesture-based text composing mechanisms have been introduced. Different glyph alike gestures for the alphabets on the screen have been introduced.

1.2 Outline of Work

The plan of our research is as follows:

- To understand the efficacy of two types of gesture-based interactions on flat touch surface namely, guided directional movement and unguided flick gestures.
- Based on the lesson learned, our next task is to develop an interaction mechanism. The objective of our interaction mechanism would be to provide a text entry system with the necessary functionalities.
- After creating an android application prototype for the method, we work on identifying ways to increase the rate and speed of the user. This is accomplished by implementing word correction and word prediction engine into our system.

1.2.1 Study 1: Study the performance of Guided Gestures

Guided gestures use spatial landmarks like edges and corners of the device to make movements. So for example, a left to right gesture would mean their finger starting from the left edge and end up at the right edge. We organized an initial study to better realize the ability of blind people to perform directional movements on a flat, touch-enabled display of a mobile device, guided by the spatial landmarks like edges, corners of the device. Our experiments and results observed are discussed in detail.

1.2.2 Study 2: Study the performance of Unguided Flicks

The eight short directional moves under the unguided flick gestures can be further divided into simple flicks (up (1), down (2), left (3) and right (4)) and diagonal flicks

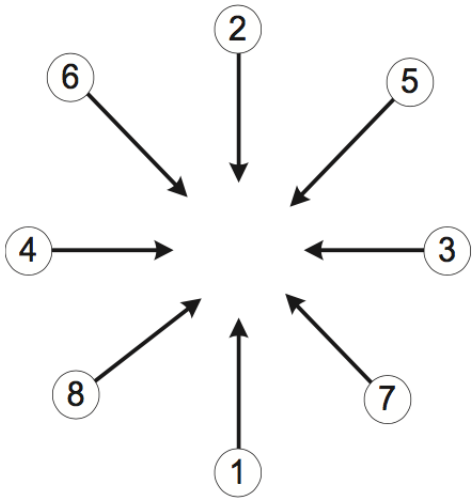


FIGURE 1.1: Eight unguided flick gestures

(top-left (5), top-right (6), bottom-left(7) and bottom-right (8)). The aim of this study was to test the hypothesis that without the favors of edges, corners or other landmarks would blind people prefer to perform the short directional moves (flicks in eight basic directions)? Could they perform them with better accuracy than the guided gestures? If so, then, which one will be of more time efficient? In response to these queries, our experiment and results observed are mentioned in the following.

1.2.3 Create a mechanism of text entry using the unguided flicks

We find out that the unguided flicks perform better than the guided gestures. And thus we use the unguided flicks to create a solution called VectorEntry, an application for text entry.

1.2.4 Increase text entry rate

After the application is developed, there is a huge opportunity to improve the speed of the user and save time. Each letter can take up to several seconds to type. And

hence, there can be significance improvement of text entry rate of the user. We discuss the following cases of correction and prediction based improvements - (We have worked only for the English language).

- Correction of spelling mistakes (e.g. You are walcome – You are welcome)
- Correction of words with context (e.g. You lake beautiful today – You look beautiful today)
- Word prediction (e.g. How are y – How are you)
- History based prediction and correction (The model creates a personal dictionary for every user, and updates it with time and use)
- Abbreviation Expansion (eg – Example, tc – take care, gm – good morning)

Chapter 2

Detailed Methodology

2.1 Study 1: Performance of guided directional movements

The landmarks on the device such as corners and edges aid the perception of blind people. Suitable gestures can be planned for the blind people, which can be guided by these landmarks. In addition to it, blind people have grown the notion of directions from their miscellaneous day-to-day work. We organized an initial study to better realize their ability to perform directional movements on a flat, touch-enabled display of a mobile device, guided by the spatial landmarks like edges, corners of the device. Our experiments and results observed are discussed in details in the following.

2.1.1 Participants

Eight participants (six males and two females) were appointed, ages ranged between 32 to 56 (average age = 40.8). Out of them, three participants were from the organization "Enabling Unit for Persons with Disabilities" (http://www.pondiuni.edu.in/HEPSN CELL/index.html), two participants were from the "National Institute for

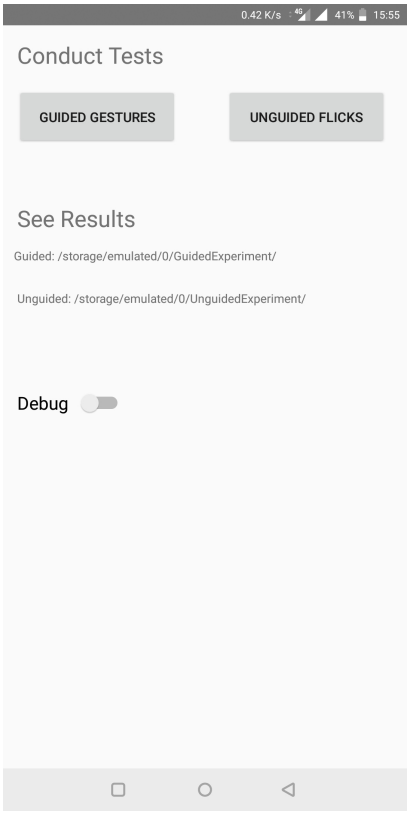


FIGURE 2.1: Screenshot of the application used to study both tests

Empowerment of Persons with Multiple Disabilities (NIEPMD)" (https://niepmd.tn.nic.in/) and three participants were from Indian Institute of Technology Kharagpur (IIT Kharagpur) (http://www.iitkgp.ac.in/), who were pursuing their PG studies. All of them were blind and having satisfactory level of cognitive abilities. All the participants were experienced computer users, assisted with screen reader software on a regular basis. Also, they were using mobile phones on a regular basis with voice synthesis software. All the participants except two participants from IIT Kharagpur were well known to the 43 telephone keypad

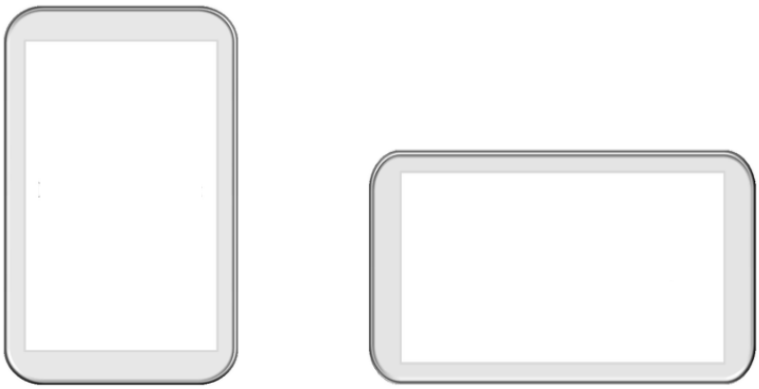
2.1.2 Apparatus

The participants were given a Samsung Galaxy Grand 2 to get oriented themselves with the (screen and bezels of the) device. It ran on the Android platform with 5.25 inch screen. As there was no any tactile difference between the touch-enabled

display area and it's non-active outer surface of the interacting device face, so, it was difficult for our participants to bound their movement only on the active touch enabled screen area, rather than the whole interacting face of the device. To provide a tactile guidance to differentiate the active touch-enabled screen area and the non-sensitive outer surface on the same plane, we fix a strip of 3 mm width and 1 mm height surrounding the active screen area. The android application I created on the device that captured all the movements on the screen and information were automatically recorded in a log file.

2.1.3 Procedure for a unit entry

Instructions were limited only within the basic eight directions: four simple moves (bottom to top, top to bottom, right to left and left to right) and four diagonal moves (bottom-right to top-left, bottom-left to top-right, top-left to bottom-right and top-right to bottom-left) in each trial. When they become ready to perform a gesture, a directional description was conveyed by playing a sound, saying the direction name (say bottom to top) to be performed. By hearing that direction, participants had to perform the respective movement. A sound feedback saying OK was delivered to acknowledge the participants against each correct entry. Unmatched entries were expressed by an error sound. We termed this process of single directional movement gesture entry as a unit entry in this experiment. After completing a unit entry, participants had to ready again for the next directional movement gesture. We allowed a sufficient time gap (10 seconds was the adjusted gap for all participants) to get ready for the next entry after completing one unit. Task time for each unit entry was measured from starting of sound prompt saying direction name to the finger release from the active screen area at the end of completion of a gesture.



(a) Height dominant position.

(b) Width dominant position.

FIGURE 2.2: Different positions of a phone set appearance to the participants.

2.1.4 Experiment

We made octuple (a sequence of eight directional movements) as a trial to decide how the directions will be presented to the participants. We made 50 different octuples. Each octuple contains all eight directions. Each direction appeared only once in an octuple. Sequences of directions were arranged at random in 50 octuples. At each trial, participants were assigned to an octuple. This experiment was run on two device orientations: height dominant and width dominant positions (see Figure 2.2)

counter balanced across participants. The device was kept fixed on a front table of the participants. In summary, the experiment was: 8 participants 2 device orientation 5 trials 8 directional moves in each trial = 640 directional movements and among them half were simple directional moves, and half were diagonal moves.

2.1.5 Results and Observation

From Study 1, we observe the following

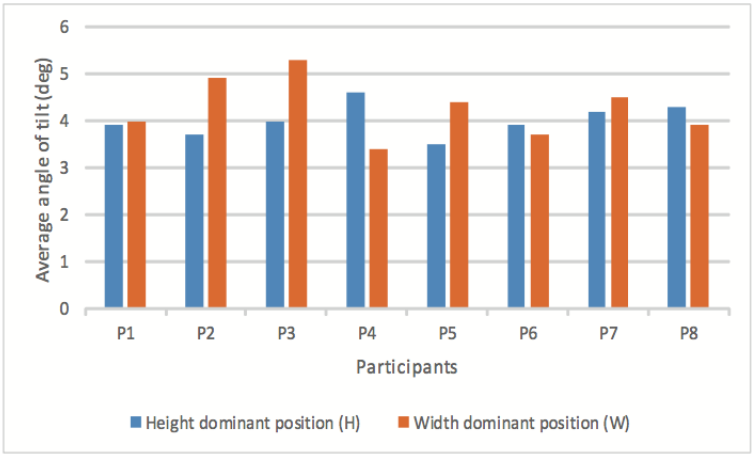
2.1.5.1 Simple Movements

All simple directional moves had been performed successfully by all the participants. We measured the tilt of each line segment joining the start and end touch points. The average angle of tilts for horizontal moves and vertical moves including both device orientations were 4.28and 3.96, respectively. In case of horizontal moves, the amount of tilt in two device orientations had no significant difference (by paired t-test p=0.4, n.s.). Also, amount of tilt in vertical moves for two device orientations had no significant difference (by paired t-test p=0.71, n.s.). At an average, for simple directional moves, in two different device positions did not have a significant angle of tilt difference (by paired t-test p=0.58, n.s.; Figure. 2.3(a)). On the average, the time taken for simple directional moves, in two different device positions had no significant difference (by paired t-test p=0.9, n.s.; Figure 2.3(b)).

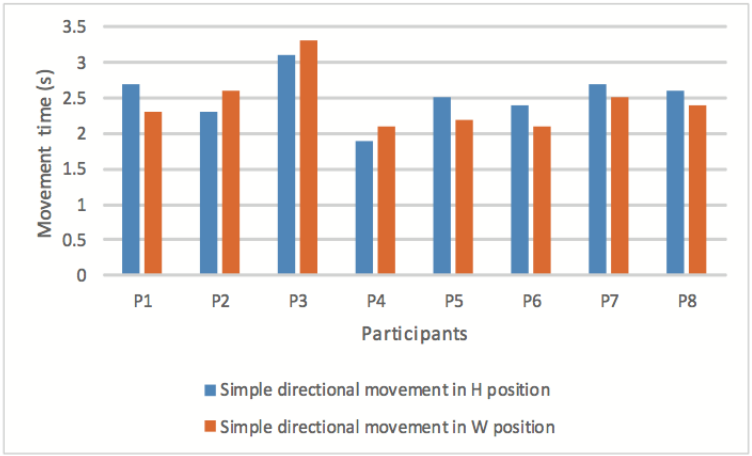
2.1.5.2 Diagonal movements

n case of diagonal moves, starting from a corner, they first reached to the opposite nearer edge, then move to the respective corner by following the edge. It was a general tendency observed during the experiment. We treated the fingertip is at the corner if it touches within 1cm^2 (152 pixel^2) square bounded box of the respective corner. We considered the fingertip touches an edge when it comes within the 76 pixels (5mm) distant from the screen's actual edge and we named it as edge line.

We have measured the angle of the line segment starting from the first touch point in corner region to their first cross of edge line as an angle of a diagonal move. Figure 2.4 describes how the diagonal moves and the respective angles are measured by symbolic examples in both height dominant and width dominant device orientations.



(a) Angle of tilt measured in two device orientations.

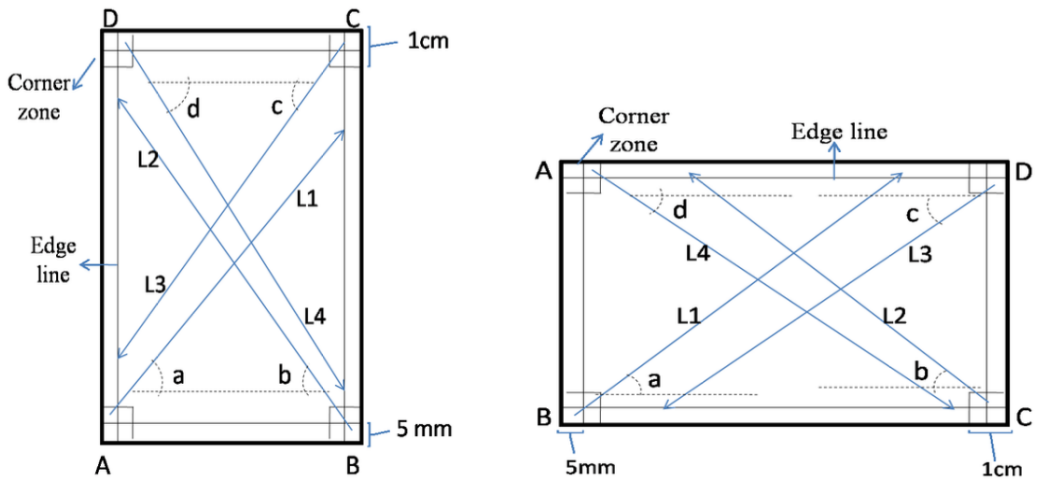


(b) Time taken to perform simple movements in two device orientations.

FIGURE 2.3: Performance of the participants with simple moves in two device orientations.

2.1.5.3 Time for movements

We measured the times taken for the diagonal movements of each participant in both device orientations. There was no significant difference for average time taken at an average for all the successfully entered diagonal movements in two device orientations (by paired t-test p=0.52, n.s.). The simple directional moves took fewer times than the diagonal moves for all the participants and the difference was significant in both



(a) Measurement of angles for the diagonal movements in height dominant position.

(b) Measurement of angles for the diagonal movements in width dominant position.

FIGURE 2.4: Measurement of angles in diagonal movements in different device orientations.

the device orientations (for height dominant position: (by paired t-test p ¡0.05, s.) and for width dominant position: (by paired t-test p¡0.05, s.).

2.1.6 Discussion

Results indicate that participants performed simple movements easily with a very less amount of tilt. In case of diagonal movements, we observed that the difference between the measured angles in two device orientations was much more when they were measured in terms of directions. But, when this angle measurement is taken with respect to the device corners, the angle difference was very less. It clearly indicates that the participants were trying to realize the screen size and accordingly they tried to fit their movements in different device orientation. Although, blind people strongly prefer the gestures that used landmarks like edges, corners of the screen, but instead of their endeavor, rather than going directly from one corner to its diagonally opposite corner, they find an elementary way to come from one corner to its opposite corner (by taking the help of opposite edge). In other words, our users failed to take the advantage of corner-to-corner movement guided only by the

corner landmarks (rather, they performed corner-edge-corner movement), and as a result, they had to traverse relatively longer paths which took more times. Also, there is a possibility to have an extra load for the participants to percept the screen dimension and coping up with the different orientation of the device; this may be a reason that made the diagonal movements much slower than the simple directional movements.

2.2 Study 2: Performance of unguided flick gestures

As an alternative to the directional gestures (which are guided by the device landmarks), it would be interesting to explore another type of gesture called flick gesture, which is not guided by any device landmark). In other words, a flick gesture is a short directional move without the assistance of landmarks like edges or corners. It may be noted that such a gesture needs less demand for the location accuracy, and at the same time, it does not need to scrawl through a long path without any assistance, which in turn may lowering the time to complete a gesture. In this study, we have explored this possibility.

2.2.1 Apparatus

Apparatus setup was similar to Study 1. The only difference was the border strip through the screen boundary was withdrawn. Study 2 was conducted with both the device orientations like Study 1.

2.2.2 Participants

We continued the experiment with the same participants, who participated in Study 1. The participants did not experience any technical changes at the period in between the end of Study 1 and start of Study 2. Study 2 was started just after 10-15 days from the end of Study 1.

2.2.3 Input mechanism and unit entry

The experimental procedure was similar to Study 1. Participants were given instruction to perform eight basic directional flicks freely at anywhere on the touch-enabled device screen. They were informed that the tilt of their performed flick gesture will be measured by the angle of tilt of the line segment joining the first and last touch points. A name of a direction was played and by hearing that, participants performed the respective flick. A sound feedback saying OK or an error sound was generated against each incorrect entry. This process was termed as a unit entry in this study. Task time for each unit entry was measured from the starting time stamp of the sound prompt of direction name to the time stamp of touch release from the active screen area.

2.2.4 Observation and Results

2.2.4.1 Detection of gestures

We log all their moves on the screen and we measured the angle of tilts of the line segments joined by the first and last touch points. To detect the direction of a flick, we consider the following scheme. The whole two-dimensional space is 360and divided into eight segments each of 45. Now, if the tilting of the line segment is within the range of (20with a vertical line (horizontal line)), then it will be considered as

a vertical flick (horizontal flick). If the direction of the gesture is at up ward (down ward), then it will be considered as an up flick (down flick); Other conjugate angle zones are clearly separated by these angle zones and diagonal flicks (up-right, up-left, down-left, and down-right) flicks were measured similarly by measuring the angle of a tilting and the direction.

2.2.4.2 Simple flicks

Angles for all the simple flicks were measured and accordingly flicks were decided as a correct or incorrect gesture. We checked the accuracy of the flicks, that is, whether the flicks were rightly performed or not. And we found all simple flicks were performed successfully.

2.2.4.3 Diagonal flicks

In case of diagonal flicks, most of the moves were correctly inputted. The angles for the diagonal flicks in both device orientation were measured in the similar way described in Study 1 (Figure 2.2) . We measured all the angles of the movements for both the device orientations. Average of angles of all the successful diagonal flicks considering all the diagonal directions and all the participants was 47.2in the height dominant position and this measure was 41.4for the width dominant position. A diagonal flick was treated as a successful entry when it's angle range and direction would match and decided as the participant was instructed to perform.

2.2.4.4 Time for flicks

We also measured the time taken to perform the flick gestures for all moves of each participant. Although, time taken at an average of all the right moves had significant differences in-between the simple flicks and diagonal flicks for both the device orientations (for height dominant position: by paired t- test p¡0.5, s.) and

for width dominant position: by paired t-test p¡0.5, s.)), but the average time taken for the flicks in two device orientations did not have significant difference (by paired t-test p=0.19, n.s.;).

The most important finding is that the average time taken for the gestures was significantly less for the current study than the same captured in Study 1 (from paired t-test p¡0.001;). The mean time to complete one gesture in Study 1 was 3.2s, whereas in the current study respective time was 1.9s. We only considered the rightly entered moves to compute the average time taken for the gestures for both the studies.

2.2.5 Discussion

In spite of their strong preferences toward the gestures that use landmarks like edges and corners of the device, we found that blind participants performed the eight basic directional flick gestures that were not assisted by any edges or corners, faster with similar accuracy than the long directional scrawl from one corner to its diagonally opposite corner and one edge to its opposite edge.

2.3 VectorEntry - Our method

In this section, we propose VectorEntry – a text entry mechanism as a touch-based interaction to small handheld mobile devices for the people with visual impairment. The principle behind this approach is to use the popular character distribution of 4x3 telephone keypad, where characters in it can be navigated by the set of directional flick gestures. In the following, we discuss the entire procedure of our proposed mechanism followed by a detail user evaluation

TABLE 2.1: Distribution of alphabets by groups and associated flicks. NA = Not Applicable.

Alphabets	**Group**	**Flick direction/type**
, . ? ”	Key 1	top-left
a,b,c	Key 2	up
d,e,f	Key 3	bottom-right
g,h,i	Key 4	left
j,k,l	Key 5	long press
m,n,o	Key 6	right
p,q,r,s	Key 7	bottom-left
t,u,v	Key 8	down
w,x,y,z	Key 9	bottom-right
‘space’	NA	two-finger tap
’backspace’	NA	double tap
: ; !	Key *	two-finger bottom left
+ - /	Key 0	two-finger down
’save & quit’	Key	two-finger bottom right

2.3.1 Entry mechanism: A two-step procedure

VectorEntry follows a two-step entry mechanism; the first step is the selection of an alphabet group, the second one is the selection of an alphabet in the selected group. The selection of a group and an alphabet in a group is discussed below.

2.3.1.1 First selection; of group

A group of three or four letters is assigned to a key in 43 telephone keypad. We keep a similar distribution of alphabets in our VectorEntry interaction, except Key 1, *, 0 and (see Table 2.1). A key in the virtual keyboard can be accessed by a directional flick or a long tap on the screen. The detailed mapping for a key selection with the gestures set is shown in Table 2.1. It may be noted that eight keys numbered from 1-4 and 6-9 will be selected eight different flicks and Key 5 will be selected by a long press anywhere on the screen. The three other keys namely *, 0 and can be selected by a two-finger flicks in the bottom- left, down and bottom-right, respectively. The system will provide a sound feedback saying the key name just after performing the

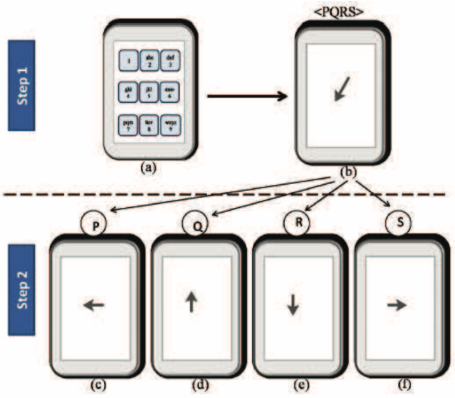


FIGURE 2.5: An example of two-step method of selecting a character in VectorEntry system.

gesture. It may be noted that users are allowed to perform a gesture at anywhere on the active touch screen area and hence there is no need for any location accuracy.

2.3.1.2 Second selection; of character

Within 10s after selecting a key, if there is no gesture performed, then the system will enter the digit as the key is marked. This way a single digit can be entered. On the other hand, the selection of a character will be done by the left, up, down (only if there are four characters on a key), and right flick gestures within 10s after the selection of a particular group. For an example, after the selection of Key 7 through a top-left flick (see Fig. 2.5(b)), a second flick will enter 'P', 'Q', 'R', or 'S' when the immediate next flick is left, up, down or right, respectively (see Fig. 2.5(c)-(f)). A two finger tap anywhere on the screen will enter a 'space' character and a double tap at anywhere on the screen will back the user at the previous state of the text entry. An automatic sound feedback saying the character that gets entered will be delivered to the user for confirming the entry.

2.4 Prediction and Correction - Improving text entry rate

In the following sections, I will describe our methods to improve text entry rate of the user.

2.4.1 Correction of spelling mistakes

e.g. "You are walcome" → "You are welcome"

This is the first problem we want to solve. Spelling mistakes are very common. We make mistakes all the time, especially when we are typing. There are human errors and we accidentally press the letters which are nearby to our intended key. In our VectorEntry method, a misinterpretation of the direction by the mobile application can cause errors in typing. Some times, words are complicated and the users would not know the accurate spelling as well. Let us discuss the methodology used to solve spelling mistakes. We are considering each word individually in this case. And so, the sentance or the adjacent words play no roles. We will later use the context and adjacent words in the next section.

After discussions with Akash Chandra, my research partner, and looking for good algorithms, we found two state of the art algorithms to use. The first, simpler one, [9] computes what is known as the *optimal string alignment distance* or *restricted edit distance*, while the second one [15] computes the *Damerau–Levenshtein distance* with adjacent transpositions. Adding transpositions adds significant complexity. The difference between the two algorithms consists in that the optimal string alignment algorithm computes the number of edit operations needed to make the strings equal under the condition that no substring is edited more than once, whereas the second one presents no such restriction.

Take for example the edit distance between CA and ABC. The Damerau–Levenshtein distance *LD(CA,ABC) = 2* because $CA \rightarrow AC \rightarrow ABC$, but the optimal string alignment distance *OSA(CA,ABC) = 3* because if the operation $CA \rightarrow AC$ is used, it is not possible to use $AC \rightarrow ABC$ because that would require the substring to be edited more than once, which is not allowed in *OSA*, and therefore the shortest sequence of operations is $CA \rightarrow A \rightarrow AB \rightarrow ABC$. Note that for the optimal string alignment distance, the triangle inequality does not hold: *OSA(CA,AC) + OSA(AC,ABC) < OSA(CA,ABC)*, and so it is not a true metric.

To prototype the algorithms, we used the Python library pyxDamerauLevenshtein (github.com / gfairchild / pyxDamerauLevenshtein). pyxDamerauLevenshtein implements the Damerau-Levenshtein (DL) edit distance algorithm for Python in Cython for high performance. See the Results section 3.2.1

2.4.2 Correction of words with context

e.g. You lake beautiful today → You look beautiful today

The previous case creates words which do not exist in the dictionary. In the current case, we discuss word errors i.e. all the words in our sentence exist in the dictionary. This type of problem is more complex than simple spelling correction. Let us take an example sentence - "You lake beautiful today". Here, all the words individually exist in the dictionary. But if we look at the complete sentence, we can say that "lake" is an spelling error and the correct word is "look". Hence, we need to look at the context of the word and the adjacent words or the whole sentence.

We discussed the problem and chose to solve the problem using bi-grams and tri-grams proposed by [Samanta, P. and Chaudhuri, B. B. (2013)][12]. The algorithm initially chooses a confusion set for each candidate word using Levenshtein distance equal to one from the dictionary words. Then it calculate the ranks of the elements of

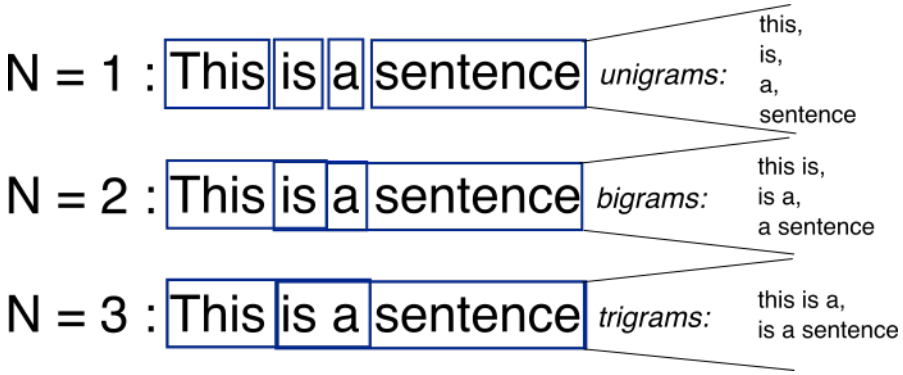


FIGURE 2.6: Examples of n-grams of a sentence

the confusion set. Based on that it detects an error and suggests some words against the detected error. Both detection and suggestion are computed simultaneously.

2.4.2.1 Calculate Confusion Set by Levenshtein Distance for the words

For simplicity, let us set maximum edit distance to one. However, the actual number can not be decided accurately without a lot experimentation and testing. With the increase of the maximum edit distance, the complexity of the model increases. The confusion set for a word W will be a set of all the words which can be formed using the set edit distance and given that they exist in our dictionary.

Let us take this sentence as example - *"His notebook has been corn."*. In this sentence the real word error is the word "corn", which should be "torn". The algorithm starts calculating confusion sets for all the words in the sentence. For the word *his*, the confusion set is *[his, him, this, is, has]*, and so on.

2.4.2.2 Form N-gram model

N-grams are all combinations of adjacent words of length n that can be found in the text. Let's take an example - *This is a sentence*. For this text, n = 1 or unigrams are *[This, is, a, sentence]*. n = 2 or bigrams are *[This is, is a, a sentence]*. n = 3 or trigrams are *[This is a, is a sentence]*. See Figure 2.6

The basic point of n-grams is that they capture the language structure from the statistical point of view, like what letter or word is likely to follow the given one. The longer the n-gram (the higher the n), the more context we have to work with. Optimum length really depends on our model and how fast/efficient we want it to be – if our n-grams are too short, we may fail to capture important differences. On the other hand, if they are too long, we may fail to capture the "general knowledge" and only stick to particular cases.

In our case, we form three types of n-grams for every word in the sentence (Example sentence - *"This is a sentence"* ; Example word *"is"*)

- Left Bigram : Word and its preceding word (e.g. This is)
- Right Bigram : Word and its succeeding word (e.g. is a)
- Trigram : Word with its preceding and succeeding word (e.g. This is a)

2.4.2.3 Estimate N-gram probabilities from the confusion set

Similar to Markov chain, we assume that occurrence of any word depends only on its previous and next words and independent of any other words in the sentence. Thus, for every word, we calculate three probabilities P1 for left bigram, P2 for right bigram, and P3 for trigram.

$$\text{P1} = \frac{count(left\ bigram)}{Sum\ of\ all\ counts\ of\ left\ bigram\ possible\ with\ words\ from\ the\ confusion\ set}$$

$$\text{P2} = \frac{count(right\ bigram)}{Sum\ of\ all\ counts\ of\ right\ bigram\ possible\ with\ words\ from\ the\ confusion\ set}$$

$$\text{P3} = \frac{count(trigram)}{Sum\ of\ all\ counts\ of\ trigram\ possible\ with\ words\ from\ the\ confusion\ set}$$

Here count implies occurrence of the n-gram in the corpus.

We calculate P1 of each element of confusion set for each word using left bigram count. The denominator here represents the summation of all bigrams consisting of

the previous word and one word from the confusion set. We get the counts directly from our corpus. In the same way we compute P2 using right bigram count in the corpus. We compute it for every elements in respective confusion set. For P3, we use the trigram count of the corpus. The denominator here represents the summation of all trigram consisting of the previous word, one word from the confusion set and the next word. We get numerator value as before. We do it for every elements in confusion set.

We assign a score to every possible word in the confusion set. The score is simply P1 + P2 + P3. We can see that $0 \leq Score \leq 3$ since P1, P2, and P3 are probabilities.

2.4.2.4 Weighted combination score

As n increases in n-grams, the count in the corpus decreases due to increase in context. The counts are more robust but become sparse. Hence, we need to assign weights to P1, P2 and P3. Let's call them $\lambda 1$, $\lambda 2$ and $\lambda 3$. By trial and error [12], we take $\lambda 1 = \lambda 2 = 0.25$ and $\lambda 3 = 0.5$.

And the final score to be $\lambda 1 * P1 + \lambda 2 * P2 + \lambda 3 * P3$

2.4.2.5 Stemming of words

There are multiple grammatical forms of the same word. Singular and plural forms, past and past participle forms, adjectives and adverbs, etc. all words exist. They are usually formed by adding the following suffixes *[d, n, r, s, y, ed, es, ly, ies]*. In our case, we do not care about grammatical mistakes, but we care about non word errors. Hence, if possible, we will stem all the words to its root before checking its count in the corpus. Examples - 1) "quarterly" $\rightarrow$ "quarter" 2) "hacked" $\rightarrow$ "hack"

This will also require a significantly smaller corpus size for our model.

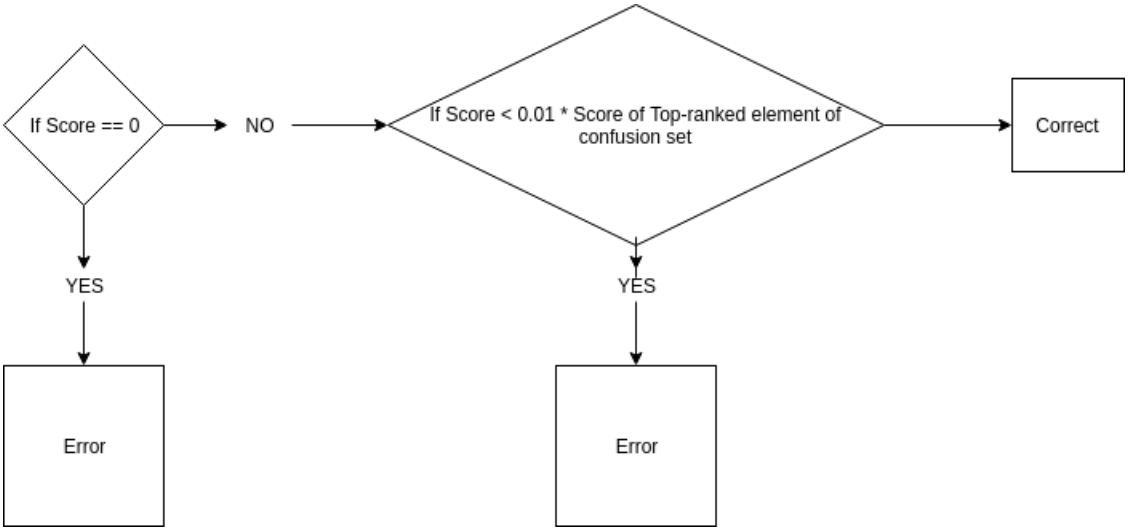


FIGURE 2.7: Algorithm to detect non word error

2.4.2.6 Detecting errors and choice of suggestions

To confirm a word as a real-word error, we set some rules. At first, we arrange the score for members of confusion set in a descending order. We have used the apriori belief that the observed test word is not a real word error. In their experiment Mays et al. obtained optimum value of this belief as 0.99 [5] which is used in our case as well. In other words, we believe that the test word can be a real word error in 1% cases. This value is used in normalizing the score in the real-word error detection algorithm described below in the flow chart Figure 2.7

If stemming is possible on a word, we make sure to use the stemmed word instead of the original. We then calculate the normalized score for the word and that is the score we use in the algorithm.

2.4.3 Word Expansion

lgtm → Looks good to me

Adding support for expansion for abbreviations in our text entry can significantly boost the text entry rate. Some of the popular and widely used abbreviations are

btw (By the way), afaik (As far as I know), fyi (For your information), tc (Take care), ttyl (Take to you later), etc. Instead of typing the long phrases, users can enter 2-3 letters instead of 15-20 characters.

We found [Gouws, S. et al] [8] as a solution to our problem. The goal is to normalise lexically ill-formed text to its most likely standard English representation, e.g. "c u at da house 2nite" -¿ "see you at the house tonight". In this paper, the authors investigate the writing conventions that different groups of users use to express themselves in microtexts. The study investigates properties of lexical transformations as observed within tweets. The study reveals that different populations of users exhibit different amounts of shortened English terms and different shortening styles. Thus this problem is very subjective to the user and depends on their writing style. Due to the lack of time of my thesis, no prototype or extended discussion could be done for this problem.

2.4.4 Word Prediction

Word prediction is one of the most difficult problems among all of our cases. It has only been a year since Google added predictive texts to Gmail, and that too on a very small scale and for usual phrases and sentences. In our model, we can enhance the text entry rate by predicting the users as they start typing. Here are a few examples of it

- Than → Thanks and Regards,
- I hope you a → I hope you are doing good.

We discussed the methods proposed by Sharma, Manoj Kumar and Samanta, Debasis [13] in Word prediction system for text entry in Hindi. The paper uses Machine Learning approach to predict words. The method works even if the users make spelling mistakes in the beginning letters and that is why it is closely related to

our case. The approach is to create a confusion set and calculate probability scores for candidates. However, it calculates different types of scores for each candidate, namely P-score, E-score, L-score and B-score. Refer to the paper for the detailed definition of the scores. The variety of scores helps in ranking accurately among the candidate predictions.

There was not enough time to pursue the approach and implement. We discussed that the case where this word prediction differs from a visually able user's device is the Input/Output. The language used is mostly the same by both types of people but the way people with vision interact with the predictions differ. For example, when we start typing on our smartphones, we can see the suggestions and often multiple suggestions. We can easily select one of it and move ahead very fast. However, for a blind person, the visual input option is missing. The possibilities limit to the use of sound for suggestions and vibration for casual nudges to the user that there is a suggestion.

2.4.5 History based correction and prediction

The idea here is that a new user will start with a pre-trained model for the prediction and correction. So, all the users start with the same functional capacity of the model. However, as the user spends more time using the VectorEntry method, the app can be personalized especially for that particular person and adhere to the personal habits of the user.

A solution is to create and maintain a personal corpus of the user on their local device. The best example of why we need this are Names. Names do not exist in the dictionary and hence a personal dictionary is required for everyone. With the personal corpus, the abbreviation expansion can be improved as well. The user can insert their own acronyms which they frequently use.

The challenge is to re-train the model on the mobile devices where the resources like RAM and CPU are not appropriate. The lack of resources on phone can result in not fully utilizing the personal corpus.

Chapter 3

Results

3.1 VectorEntry app

The results of the two experiments indicate that we should use the directional flicks over guided gestures because of the faster entry rate and similar error rate. The experiment was conducted using an android application I created shown in Figure 2.1

After this conclusion, I created an Android application (Figure 3.1) which uses the VectorEntry method to enter the texts. The look and feel of the application has been ignored as of now because it will be used by people without sight. And thus only the movements and gestures are important.

3.2 Prediction and Correction

All the implementations and codes are hosted publicly at

https://github.com/OrkoHunter/vectorentry-backend

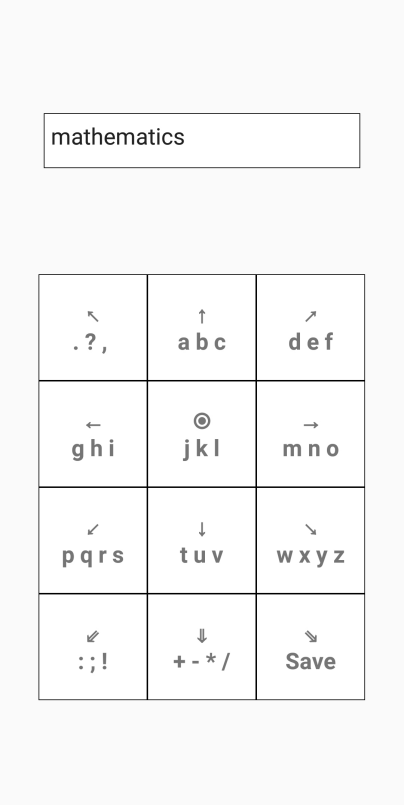


FIGURE 3.1: A screenshot of VectorEntry method in action

We need a large English dictionary to proceed with. We used the WordNet [6] corpus provided freely at wordnet.princeton.edu. We used the python nltk library, a toolkit for Natural Language Processing, to process the corpus and use it in our module. Upon first attempt, we realized that the corpus size is over 4 Gigabytes. This makes it extremely hard to run it on a personal laptop with about 8 GB of memory. A low end computer also limits our maximum edit-distance for correction algorithm. We then used a high-end server provided by the Department of Computer Science, IIT Kharagpur to run our module.

We implemented the algorithms from Section 2.4.1 and 2.4.2 and hosted it on our local server. The most popular available tools were written in Python and Android app requires Java, hence we were unable to port our code to the app. The repository is stand-alone but can easily be integrated with the app using REST API. However, this would mean internet calls for each time we want our prediction and correction

engine to run, which may not be feasible.

As of now, we run the algorithm on a whole sentence. So, given a stream of text as input, we provide the results as an output. As of now, the first word of a sentence is not being corrected because of the comparatively smaller corpus size. We are also unable to preserve the case of the text. So, for example a camelCase or Uppercase word is transformed to the lowercase, and in the results, we can not know which case belongs to which letter. A possible solution is to index the words in the input text, classify their case, and in the results for the same index, transform the word into its previous case.

Our algorithm is useful only if there are spelling mistakes, but they will not correct grammatical mistakes. Due to the size of the corpus, we have to stem the words (trim the endings like -ly or -ed) and thus we lose the grammatical form of it. By increasing the computing resources and corpus size, we will not need to stem the words, and by increasing the maximum edit distance, we can cover grammatical mistakes too.

The algorithm in section 2.4.2 (context based) solves the problems of 2.4.1 (spelling mistakes) as well. Thus, we have not implemented a separate edit-distance based spelling checker and relied on the bigrams and trigrams based approach.

Chapter 4

Knowledge and Skills Acquired

The applications are programmed using the following tools which I learned in the process:

- Android SDK: The Android SDK (software development kit) is a set of development tools used to develop applications for Android platform.The Android SDK can be broken down into several components. Some of them are

  ADB: The Android Debug Bridge (ADB) is a program that allows to communicate with any Android device. It can be used to access shell tools such as logcat, to query device ID and even to install apps.

  Android Emulator: The Android emulator allows to test and monitor apps on a PC, without necessarily needing to have a device available. Using emulator, we can run the application on a wide variety of virtual devices with different hardware configurations.

- Java: Java is the most popular language used to develop android applications. Native applications can be build only using Java. Android SDK contains a large number of Java libraries for app development.

- XML: XML or eXtensible Markup Language is used for designing the user interfaces in android. XML serves as an easy yet powerful tool to build simple as well as complex interfaces. These interfaces are binded with the logic the Java code. XML is also one of the most commonly used data formats on the Internet. Thus, any application using Internet will most probably require XML.

- Android Studio: This is the official IDE for android development developed by Google. Android studio provides advanced text editor features and binds all the tools to create a uniform interface. It provides features like instant run, code autocomplete, debugging modes, SDK manager, instant design result for XML and other advanced features.

- Flask: Flask is a web service framework in Python. We used flask to create API end points for correction and prediction.

- NLTK: The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language.

- NumPy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. NumPy is incredibly fast and memory efficient and useful for dealing in scenarios with limited resources.

The most challenging part was to determine the angles and the directions of the gestures in both the device orientations. For the Prediction and Correction engine, I personally felt that the research requires a vast knowledge of Natural Language Processing and Machine Learning, and I being a beginner in the field, struggled to do a high-impact work.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

We have created a keyboard for our blind users. We want them to use it without interruptions. We want everything to run fast. Since, the input methods and algorithms run offline, they can easily write. Models to predict and correct words require a lot of memory and disk space. However, if we keep the engine online and communicate over API calls, there are going to be interruptions. Internet requests are not reliable all the times. Sometimes the user may not have access to the internet. Sometimes the server would take too much time to respond. And hence, our breakthough implementation will be when everything runs offline as the user types.

In conclusion, I want to discuss the uniqueness of our problem with the current research in the field. We are developing a text entry mechanism but the limit is that our users have limited methods in which they can receive Input (visual impairment). However, their output method of speaking is not impaired. And thus, their speaking and writing style have the same characteristics as that of a visually-abled person. Hence, I/O methods of our approach of prediction and correction is very crucial to the adaptation by a user. There are primarily two senses by which they can receive

input 1) Sound 2) Feel (vibration). While they are typing, a low-intensity vibration can mean that there is a suggestion and they can make an action to review the options (which will be provided by sound). A high-intensity vibration can mean that they have made an error and whether they want to review what our engine is proposing as a correction.

5.2 Future Work

Possible future work include:

- Offline implementation of the correction engine.
- Work on Text Prediction algorithms and its implementation in the application.

Bibliography

[1] WHO 2016. Blindness in developing countries. *webaim.org/projects/screenreadersurvey5/*, 2016.

[2] WHO 2018. Blindness and visual impairment. *http://www.who.int/en/news-room/fact-sheets/detail/blindness-and-visual-impairment*, 2018.

[3] Android. Android operating system. *http://www.google.com/mobile/search/*, 2014.

[4] Apple. Apple's voiceover. 2014.

[5] R. L. Mercer E. Mays, F. J. Damerau. Context based spelling correction. *Information Processing and Management*, 27(5):517–522, 1991.

[6] Christiane Fellbaum. Wordnet: An electronic lexical database. *Communications of the ACM*, 38(11):39–41, 1998.

[7] Google. Google's voiceactions. *http://code.google.com/p/eyes-free*, 2014.

[8] S. Gouws, D. Metzler, C. Cai, and E. Hovy. Contextual Bearing on Linguistic Variation in Social Media. In *Proceedings of the ACL-11 Workshop on Language in Social Media*. Association for Computational Linguistics, 2011.

[9] B. John Oommen; Richard Loke. Pattern recognition of strings with substitutions, insertions, deletions and generalized transpositions. *Pattern Recognition, ISSN: 0031-3203*, 30(5):789–800, 1997.

[10] Caleb Southern Mario Romero, Brian Frey and Gregory D Abowd. Brailletouch: Designing a mobile eyes-free soft keyboard. *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, page 707–709, 2011.

[11] Nielson. Smart phone users in india. *Nielson*, 2016.

[12] Pratip Samanta and Bidyut B. Chaudhuri (Indian Statistical Institute Kolkata). A simple real-word error detection and correction using local word bigram and trigram. *The Association for Computational Linguistics and Chinese Language Processing (ACLCLP)*, 2013.

[13] Manoj Kumar Sharma and Debasis Samanta. Word prediction system for text entry in hindi. *ACM Transactions on Asian Language Information Processing (TALIP)*, 13(2):8, 2014.

[14] Radu-Daniel Vatavu. Visual impairments and mobile touchscreen interaction: State-of-the-art, causes of visual impairment, and design guidelines. *International Journal of Human–Computer In- teraction*, 33(6):486–509, 2017.

[15] Robert A. Wagner and Roy Lowrance. An extension of the string-to-string correction problem. *Journal of the ACM (JACM)*, 22(2):177–183, April 1975.