JustType - Efficient Text Entry Using 8 Keys (Computer Control Too!)

Cliff Kushler

Mark Illing

Continuous Path

info@ContinuousPath.org

http://continuouspath.org/

1 CONTENTS

2 FOREWORD

For most people, computer access is accomplished with a keyboard and mouse. A keyboard with 100+ keys enables some to type well over 80 words per minute. Most are lucky to reach 40 WPM. For those with severe mobility impairments, access using a keyboard with 100+ keys can be difficult or impossible because they may lack sufficient range-of-motion, fine motor skills, strength or stamina. Minimizing the number of keys needed for computer access significantly improves success for people with mobility impairments because the much smaller number of keys can each be significantly larger and easier to target.

JustType text input uses seven ambiguous character keys with an eighth providing a means to delimit words and select from possible alternate words. This eight key arrangement exists to ease text entry for people with different disabilities where accurate control of more keys is very difficult or time consuming. Using JustType and an appropriate assistive device, a user can enter text using 0.64 – 1.003[1] keys per character, dependent on user configuration, with varying degrees of resulting cognitive load.

> *The goal of this paper is to demonstrate how efficiently JustType can enter text and control a computer through use of only 8 keys and to encourage adoption of the JustType techniques by manufacturers of assistive technology.*

The information provided here is available for use royalty free. Continuous Path and the authors of this paper will provide complimentary consulting for companies working to implement a JustType-like interface for their products or individuals doing research in this area. The authors of this paper may be contacted through info@ContinuousPath.org.

Metrics: Throughout this document, we use keys per character (KPC) as a metric for comparison. 1.0 keys per character is the general output of a standard hardware keyboard. While we would like to compare the results of the JustType concepts to what can be achieved using the prediction engines, the manufacturers are understandably unwilling to provide metrics as there are a great deal of variables upon which any metric is based. Certainly, the prediction engines can achieve much better input rates since the ambiguity of the input is much less when there are significantly more input keys.

The numbers listed in this document, unless otherwise specified, were measured using instrumented software with a specific configuration of JustType. There are a large number of configurable variables to optimize for a specific user's needs/preferences. Numbers within a section may be compared as the unspecified variables are unchanged. Numbers between sections are not comparable as they likely have different configurations. We offer our findings of improvement for various techniques, but these techniques, when combined, do not accumulate linearly. Finally, in our testing, we do provide a training period to the prediction engines (and embedded language models) prior to making our measurements. This training is not extensive and the authors believe that over time, the typical user will experience even greater efficiencies as the disambiguation engines learn their text patterns.

[1] 0.64 – BRZ keyboard, two key select, 7 Selection List words, significant cognitive load
1.003 – BRZ keyboard, one key select, Rote Words disabled, minimal cognitive load

3 WHO BENEFITS FROM THE JUSTTYPE TEXT INPUT SYSTEM?

JustType is an eight key keyboard with text entry rates better than one key per character. The users that will benefit from a keyboard with the JustType concepts incorporated are literate users who face challenges in using a full QWERTY keyboard, or who are using a text input device for which the size of the keyboard is constrained. These users may need an input system for augmentative and alternative communication (AAC) while others will want full computer access.

3.1 MOTIVATIONS

The JustType keyboard is for use in an environment where for some reason there are challenges associated with providing or accessing a full keyboard. A full QWERTY keyboard with 100+ keys or a good speech-to-text package are quite efficient at generating text. But in an environment where a full keyboard is either impractical or inefficient, the JustType keyboard excels using only eight keys. Various challenges are often inherent in a particular user's situation, or they may exist due to design constraints that are inherent in a particular application.

3.1.1 Design Constraints

For example, variations of the JustType keyboard already exist on over 7 billion mobile feature phones. Mobile phones with hard keypads are constrained by the number of keys they contain and generally with specific character assignments to the keys. In this situation, text can still be entered using one keypress per character. You can read more about this in section titled "A Brief History of Constrained Text Input".

Another example of where a JustType keyboard would be effective is on your TV remote control. Any application where an input device has been designed with a limited number of keys is a candidate.

3.1.2 Mobility Impairments

Those that will most benefit from use of a JustType keyboard are people with mobility impairments. This keyboard provides an effective means of text input for users with mobility, fine motor control, manual dexterity or range-of-motion impairments. By creating an input environment requiring only a limited number of targets/keys, we create new opportunities for these users while maintaining excellent input efficiencies.

We're often asked to quantify how many people with mobility impairments would benefit from a JustType keyboard. The following is a list of mobility impairments that, combined, affect over 60 million people in the US. While a good number of individuals with these impairments is able to use existing mainstream keyboards, we know that a significant subset of these people is otherwise limited or excluded from using mainstream keyboards. We just don't know the actual number.

ALS - Amyotrophic Lateral Sclerosis	30,000	http://www.alsa.org
Spinal Cord Injury	300,000	http://www.brainandspinalcord.org
Muscular Dystrophy	50,000	http://www.webmd.com
Cerebral Palsy	764,000	http://www. ucp.org

Multiple Sclerosis	400,000	http://www.healthline.com
Stroke Survivor (each year)	600,000	http://www.strokecenter.org
Fibromyalgia	5,000,000	http://www.fibrocenter.com
Spina Bifida	166,000	http://www.ninds.nih.gov
Parkinson's disease	500,000	https://www.caring.com
Arthritis	52,500,000	http://www.cdc.gov

3.2 Literate User

The JustType Text Input System requires the user be literate, with basic skills in reading and writing. The word prediction engines available today are quite efficient, even when some number of spelling, punctuation, capitalization and even word break errors in input are made. The power of the language models underlying such systems also support disambiguation engines to handle some of these errors even with the added challenge due to each key-press already being ambiguous.

The JustType Text Input System is ideal for users who acquired their condition post schooling. But it is also possible to configure a JustType system to create a valuable tool in initially teaching literacy skills, which is discussed briefly below.

4 Considerations for an Effective Text Input System

It is important to consider what it is that makes an input system effective for an individual with mobility impairments. Let's assume that for our purposes, the definition of "effective" is maximizing input speed without imposing an unacceptable physical or cognitive burden on the individual. While there are other ways that communication can be achieved, let's also assume that the goal is to generate an orthographic representation of the desired output message. It can be argued that there are a relatively small number of factors that determine how an input system will perform in this sense:

1. The size of the selection set. Irrespective of the selection technique used, the average time and effort required to make a selection is a function of the size of the selection set (in the case of a keyboard, this is simply the number of keys on the keyboard).

2. The number of selections required to produce a given output string. Not surprisingly, there tends to be a natural inverse relationship between the size of the selection set and the average number of selections required. For example, the selection set can be reduced to only two inputs. Let's call them "dit" and "dah," arguably the smallest feasible selection set, though in its standard form, Morse code also introduces a timing element that adds very significant requirements to the input process beyond simply selecting the input elements in the correct sequence. Clearly, the sequence of dits and dahs needed to encode a message will be significantly longer than the number of letters in the standard orthographic representation of the message. Other approaches prioritize minimizing the number of selections required, at the cost of greatly increasing the size of the selection set – for example, designing a system with

over 100 icons on separate keys. This also introduces additional cognitive burdens, discussed below.

3. Cognitive load. There are two aspects to the cognitive burden a system imposes on an individual. The first can be thought of as the "learning curve," how difficult it is for an average user to learn how to achieve a given level of proficiency in using the system. While it is of course preferable to minimize this first aspect, if the burden is not unreasonable, it is relatively less important than the second, since once proficiency is achieved the system has the potential to serve all of the individual's communication needs for a lifetime. The second aspect can be thought of as the "attention load," how much time the individual is required to spend attending to the system itself in order to successfully generate the desired output. This is an ongoing cost that directly affects the efficiency of the input system. This also has less tangible costs – to whatever degree the individual must focus their attention on the input process, this directly detracts from their ability to focus on what it is they are trying to communicate, and also from their ability to attend to the conversation in which they are immersed, which often continues unabated while the individual is composing a response.

This framework provides a perspective from which we can consider what JustType brings to the table as an input method. As for the size of the selection set, word level disambiguation or word prediction can be used with a wide range of selection set sizes, including as few as three keys, but the efficacy of the various selection set sizes varies widely. For reasons that will be discussed further, JustType is based on a selection set size of eight keys, chosen as a perhaps optimal compromise between shrinking the selection set for easy selection of each key and having a large enough selection set so that the disambiguation engine can work efficiently.

4.1 Cognitive Load

We will start by considering cognitive load issues. Although there are obvious differences between a JustType keyboard and a standard one-letter-per-key keyboard, both are fundamentally based on knowing and entering the standard orthographic representation of each word. Both require the user to have fundamental literacy skills. For individuals who already have the requisite literacy skills, this is a great advantage – all that needs to be learned is the letter layout of the keyboard. While there is a learning curve in getting to where the use of the keyboard and its layout becomes automatic, in the authors' experience, this happens relatively quickly with regular use of the keyboard, and pales in comparison to the fact that the "encoding" – simply the spelling – of every word is already known to the user, providing efficient access to a virtually unlimited vocabulary. It is hard to overstate the importance of this.

At first blush, it is easy to then conclude that the JustType system offers little to individuals lacking the necessary literacy skills, but we will argue that this is not the case. Children do not learn to talk by sitting silently month after month, listening to the speech around them, until finally they begin conversing in complete sentences. Likewise, it is not appropriate to expect children to acquire literacy solely by exposing them to opportunities to practice reading. Literacy, like all communication, includes both receptive and generative components, and children with motor impairments need access to effective tools that enable them to practice or experiment with both reading and writing. Used as a literacy acquisition tool, JustType could be initially configured with a very small number of words. As a

sequence of keys is entered on a JustType system, one is effectively traversing a conceptual "space" containing all of the possible words. As keys are entered, the number of letters that could validly follow the sequence of keys already selected drops rapidly. When the system vocabulary is constrained to a small number of words the user is actively learning, this drop-off is profound. The keyboard display can be configured such that, after each key activation in spelling a word, keys are displayed with letters that are not "valid" at that point in the sequence grayed out. This gives immediate feedback to the beginning speller that, if their next intended letter is grayed out, that a spelling error has already occurred, and that they need to backtrack or seek assistance. This is all within the context of a system that, as we hope to show, can be configured to be efficiently accessed with a wide variety of physical access techniques that can accommodate the needs of a wide range of individuals. We would argue that, rather than seeing the need for literacy skills as a barrier to using JustType, that JustType should be viewed as a powerful tool in helping with the very acquisition of such skills.

4.2 Selection Set Size

Next, let's consider the size of the selection set for JustType, and in particular the case where the keyboard is designed to function with a set of eight keys. Claude Shannon estimated that the average entropy of a letter in English is 2.62 bits[2], and though other estimates vary considerably, we can take this as a starting point for discussion. A set of seven keys (the keys with letters) corresponds to an entropy of 2.81 bits per key, so this appears to be perhaps an ideal fit for a system where each key press is disambiguated to correspond to a single English letter. Although other estimates of the entropy of English can be as low as 1 bit per character, practical experience also shows that an effective disambiguating keyboard certainly requires more than two letter keys. The close match with Shannon's estimate offers a compelling argument that what JustType is actually doing is effectively eliminating the redundancy inherent in English spelling. This can in a sense be argued to be an optimal approach to increasing the efficiency of text input without requiring the user to learn a new encoding – they are still simply entering the standard spelling of each word (with word completion further enabling them to enter a word without entering the complete spelling). Further advantages of the choice of a selection set size of eight keys will be discussed below.

4.3 Selection Sequence Length

The original implementation of JustType utilized a word database in which words were sorted to appear in a Selection List in descending order of frequency in a representative corpus of the language. This means that at each point in entering a sequence of keys, the word at the head of the Selection List (thus requiring only a single activation of the "Next" key) is the most frequent word among the possible matching candidates. Early research on JustType developed a searching algorithm to identify an optimized assignment of letters to keys such that the cumulative frequency of all words that are "hidden" lower in the list is minimized. In a test where the optimized JustType keyboard was used to draft a scientific paper (ignoring the words not present in the database that would have initially required two keystrokes per letter in order to add them to the database), it required only 1.0051[3] keystrokes per

[2] Shannon, Claude E., Prediction and Entropy of Printed English, The Bell System Technical Journal, January 1951

[3] This number references the original research using word frequency lists. The current JustType implementations which use prediction engines incorporating sophisticated language models, we have measured input 1.00274 keys per character.

character, or approximately one extra keystroke every 32 words. In reference to the second factor listed above – the average number of selections required to produce a given output string – this is very close to one selection per character, virtually the same as a full desktop keyboard.

That is the foundation upon which the system is built, already constraining the length of the selection sequence to be comparable with systems with dramatically larger selection sets. To further reduce the average length of the selection sequence, modern text input systems are able to take advantage of advanced natural language models to aid in predicting the word the user is entering, so that a list of words can be offered for selection before the word has been fully typed (word completion), and often before a single letter (or key) has been entered (word prediction). Due to constraints on maximum database size, early implementations of JustType did not utilize natural language models. Limited word completion functionality could be made available at each point in entering a key sequence by "looking ahead" in the database to identify a limited number of the highest frequency words that could be entered by extending the current key sequence, and this provided some capacity to reduce the selection sequence length in some cases.

4.4 Reducing The Cognitive Load Of A Word Prediction System

Natural language processing technology offers the promise of further reducing – perhaps quite significantly – the average number of selections per character. While it is relatively simple to accurately measure the extent of the reduction that can be achieved, without empirical user studies, the question remains as to whether the reduction in keystrokes translates into a reduction in the time required to generate a given output string. We need to take into account the fact that this reduction comes at the cost of needing to attend to an ever-changing display of possible word predictions in order to observe when the desired word is offered for selection, and of course whatever effort is required to select the word for output.

The impact of the cognitive load associated with a word prediction (or word completion) system goes beyond simply the time required to scan – at least on occasion – the list of predicted words. To be maximally effective, an input system needs to be as "transparent" as possible, allowing the user to focus on what they are trying to say without need to devote too much attention to the mechanics of entering each word. The individual using the system is attempting to communicate, to convey a thought that they are trying to express in words. In many cases, the individual is taking part in a conversation with one or more people that will often continue even as the individual is working to compose a contribution to the conversation. This means the individual's attention is already split between: composing and bearing in mind the message that he or she wishes to express; following the conversation that is ongoing; and devoting the attention required to manipulate the input system to generate the intended message. In this context, adding the burden of scanning and reading a succession of lists of essentially unrelated words can easily be seen to be a potentially serious distraction from what the individual needs to accomplish, and the value of a word prediction system – at least as they currently function – needs to be seriously examined.

While the authors are not in a position to develop another natural language model and implement a new word prediction system, consideration of the issues outlined above leads us to propose certain modifications to the user interface for such systems that seem likely to significantly reduce the potential cognitive burden of such systems while still being able to take advantage of much of the potential

reduction in keystrokes that they offer. Existing systems basically require the user to arbitrarily choose when to scan the list of predicted words in the hope of finding their intended word. Prior research[4], though done with much older and presumably less sophisticated prediction systems, has shown that individuals with mobility impairments are less efficient using such systems than they are when such systems are not used.

It is clear that such systems must calculate some kind of metric for each predicted candidate word – for example, an estimate of the probability that a candidate is the intended word – so that the list of candidates can be rank-ordered in the Selection List. The following approach, while it is likely to fail to take advantage of every opportunity to identify and select the intended word as soon as it is first offered as a predicted candidate, should tend to minimize the amount of time spent scanning prediction lists while still taking advantage of significant keystroke savings:

1. Enable the user to set a probability threshold that the metric for a prediction candidate must exceed in order to be brought to their attention as a "valid" candidate. This threshold can be expressed as "what percent of the time will I tolerate having my attention drawn to a candidate that is not my intended word." Alternatively, this threshold can be applied to a set of candidates that can be simultaneously displayed on the screen, such that the threshold applies to the probability that one of the simultaneously displayed candidates is the intended word.
2. Enable the user to select how a valid candidate is to be brought to their attention. For example, the candidate could be transparently displayed, superimposed over the on-screen image of the keyboard. Alternatively, a valid candidate could be spoken over a private audio channel, or the prediction list could be briefly highlighted, an audio tone could be generated, etc.

The user can then focus on simply entering the text they wish to generate, with their attention being drawn to valid candidates that will – depending on their chosen threshold – most of the time be (or in the case of a limited set, include) their intended word. Of course, the more discriminating their threshold is, the less often they will have their attention drawn to a candidate. But the user has a tool that can effectively limit the amount of time wasted scanning lists of irrelevant words.

Note that the system has continuous feedback available to use both to automatically refine the probability threshold for valid candidates, and to refine its algorithm (or data) for calculating candidate metrics (which such systems are already routinely doing in some fashion). Each time a word is selected for output, the system can review the history of its keystroke-by-keystroke analysis, and, for example, collect statistics on the point(s) in the sequence (if any) at which the scoring metric for the selected word was higher than any other candidate (and what that metric value was), indicating that the current threshold may be set too high, and the intended candidate could have been correctly identified earlier in the sequence. The same statistics can be collected for the highest-scoring competing candidates so that the threshold is not set so low that a candidate is falsely predicted as the intended word. This information can be accumulated in a database that can be analyzed and used to dynamically optimize the value of the threshold for valid candidates to maximize the likelihood of triggering on the intended word with no more than the user-specified threshold of false positives. If the approach is based on

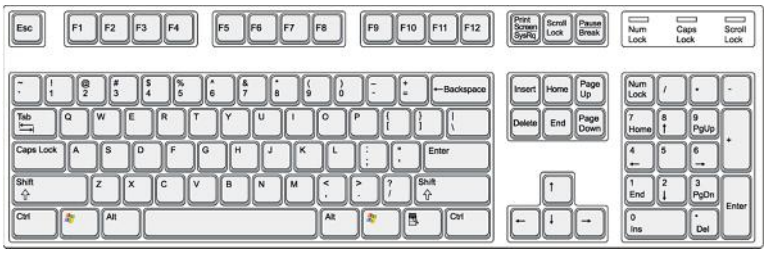
[4] Modeling the speed of text entry with a word prediction interface, Koester and Levine, IEEE Transactions on Rehabilitation Engineering, Sep. 1994

identifying a set of valid candidates, the statistics collected track when the selected word was (or was not) included in the valid set.

Again, the authors are not aware of any commercially available prediction system with these features, but contend that the development of such a system would make a powerful addition to a JustType input system – or for that matter, any system that offers context-based word predictions and completions.

5 JustType Keyboard Description

A standard computer keyboard is divided into sections according to function type - alphanumeric and symbols, a numeric keypad, control keys, function keys and navigation keys. 100+ keys are used so that most all functions of the keyboard are always accessible. Most actions or commands that you can enter through a menu or with your mouse can also be executed using keyboard shortcuts, though often requiring two or more simultaneous key activations (ctrl-, alt-, shift-, etc.). This keyboard is great for mainstream users but next to impossible for people with a variety of mobility issues where 100+ targets is unmanageable. This keyboard generally achieves 1.0 keys per character (KPC).



Standard PC Keyboard

vs.

JustType Keyboard

By comparison, the JustType keyboard provides full computer access through a keyboard with only 8 keys. Eight keys is not only far fewer than standard keyboards, it enables the use of a number of techniques for physical access that can make each individual key activation far quicker and easier (Fitts' Law[5]). While we describe three access methods in some detail below, the general concepts can be extended to other techniques as well. Eight keys do not provide immediate access to all keyboard functionality. Instead, the keyboard uses multiple layers with common functionality generally available in a single layer. In the vast majority of cases, the primary need is for simple and efficient text input, which can be achieved without changing layers. This keyboard generally achieves 0.64 – 1.003 KPC[1].

5.1 8 Keys

The JustType keyboard design was motivated by work that led to the development of efficient interface devices that were easy to access, but which only provided a limited number of separate inputs. The ease with which these interfaces could be manipulated despite severe motor impairments made it clear

[5] The time required to move to a target is a function of the distance to the target and its size.

that there was a need to develop an input method such that text could be generated efficiently despite the limited number of keys. The result was that 8 keys provides enough functionality to accomplish most tasks in one layer of the keyboard. Eight keys also provides a number of flexible key arrangements and physical interfaces that make the keyboard accessible for most with mobility impairments.

Note that throughout this document, we refer to "keys" which might seem to refer only to physical keys on a keyboard. Our use of the term "key" refers to any method of activation of an accessible device that can be captured and interpreted as a key activation. We anticipate interactions with joysticks, touch screens, head/eye trackers, single or multiple switch scanning and reduced keyboards.

5.2 Key Layout

Here, we discuss three keyboard arrangements and the advantages of each. The remainder of the document has examples using the Square arrangement, though they apply to any arrangement. The location of the keys is unimportant; the functionality associated with each key is the focus.

Square

The square arrangement of keys is a 3x3 arrangement omitting the center key, essentially a tic-tac-toe grid without the center square. Another description is a key at each of the compass points: N, S, E, W (sides), and NW, NE, SW, and SE (corners).

This arrangement was conceived to be used with a joystick, eye-gaze or head-tracking, or through directly activating a keyboard or touchscreen when that is a preferred option. Anyone that can navigate a wheelchair with a joystick should also be able to access this layout with a joystick. Note that, while a joystick has often been used to navigate among the keys of a keyboard array (requiring a separate action to activate the key to which the user has navigated), in this arrangement, a single movement of the joystick in one of the eight compass directions immediately activates the corresponding key. This is a critical distinction from previous input methods using a joystick to access a keyboard array.

Linear

The linear arrangement of keys is a 1x9 arrangement. This arrangement is primarily for individuals for whom single-switch scanning is the preferred method of access. Keys are highlighted sequentially and the user activates the switch to access the key. In this case, the ninth key allows for the inclusion of a delete key. For individuals with sufficient control of time-based switch activation (e.g. controlled execution of a "long press" on the switch when selecting a key), various alternate functions can be introduced to increase the flexibility and power of the keyboard, such as switching to an alternate set of key definitions for the next selection.

Custom

The custom key arrangement can accommodate any interface that allows the user control over 8 inputs. One example would be a touch screen device with a custom key guard in place that allows for the user's effective access to 8 keys. The fact that 8 is a power of 2 also makes many other input methods possible, enabling the system to make the most of the input actions that are most comfortable for a

particular individual. For example, an individual who can quickly activate either of two switches can activate any of the eight keys with three activations. Conceptually, this is equivalent to a fixed-length Morse code (all codes consist of three dits or dahs), or to a sort of "binary search" approach where each successive switch activation eliminates half of the keys (which can be displayed visually, so that no abstract codes need to be memorized). Note that while this approach requires two switches, there is no requirement for control over the timing of the switch activations, only the sequence.

5.3 Primary And Secondary Actions

The 8 keys provide 8 actions based on a "tap" of the key. These are the primary actions for the key. The 8 keys also provide 8 actions based on a "long press" of the key. These are the secondary actions for the key. This enables 1 layer of the keyboard to support 16 actions. Conceivably, you could add a "very long press" to obtain 8 additional actions, though we believe the cognitive load for the user increases significantly and limit ourselves to 2 actions per key.

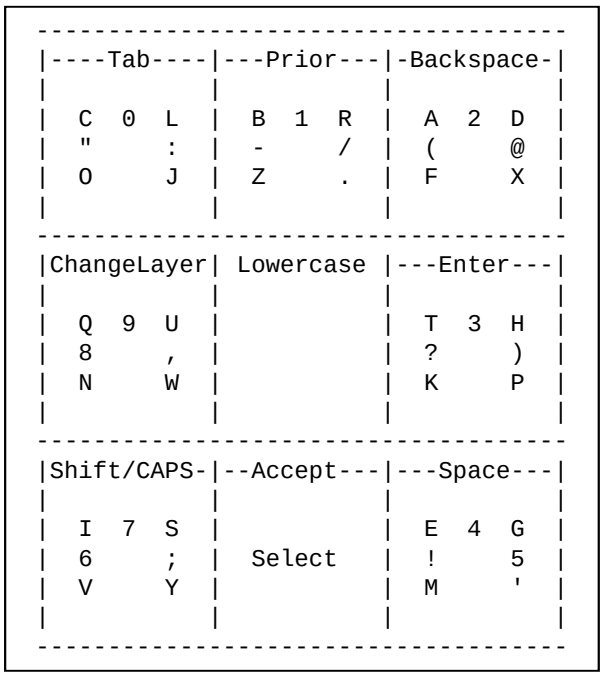
Note also that the term "long press," which implies control of timing, is used because this is probably the most common approach to providing a mechanism to associate a secondary function with a key activation. However, there are often other creative approaches to implementing an access method for a secondary (and lesser used) key activation. For example, in the two-key, fixed-length "Morse code" approach of the previous section, an alternate function could be invoked by activating both keys simultaneously before releasing the third switch activation of a sequence.

To define our terms, a key "tap" is any activation of a key that is easiest for the user and a "long press" is any activation of the key that is distinguishable from the "tap." Some user's conditions may limit their ability to easily create a key "tap" distinguishable from a "long press" of a key based on time. In this situation, we might suggest adding a ninth key (any additional input that can be activated) where activation of this ninth key implies the next key is interpreted as a "long press".

The next section, JustType Text Entry, only makes use of key "taps". The "long press" is necessary for infrequent events and is discussed more in the (Computer Control Too!) Section.

6 JUSTTYPE TEXT ENTRY

JustType provides complete functionality by enabling the user to switch between various alternate keyboard layers. Note: the design of the software is flexible to allow new keyboard layers to be defined with additional functionality.



Note: This box shows the text entry layer. Each key is shown with a border and a text based description of what characters and/or functions are assigned to the key. The top line indicates the secondary action for the key when it is long pressed. The key face is labeled with the characters ambiguously associated with the key when it is tapped. The "south" key is the Select key, which is explained below.

On ambiguous letter keys, notice that the letters on each key are arranged in the same "upside-down 'U' shape" as the ambiguous letter keys themselves (the seven keys excluding the Select key). This is to provide a means to identify a character unambiguously through a pair of successive key taps.

While most text entry is interpreted ambiguously and resolved using a disambiguation/word prediction engine, each successive pair of key taps is simultaneously interpreted unambiguously. The first key tap of a pair identifies the key on which the intended letter appears, and the second key tap identifies the position of the intended letter on the first key. See 7.4 Explicit Word for more on simultaneous unambiguous input.

JustType Text Entry – To enter a word, the user successively taps the keys associated with each letter in the word. The user ignores the fact that each of these taps is ambiguous, and in and of itself is not sufficient to fully specify each intended letter. After entering a key for each letter of the word, the user activates the Select key to denote the end of the word, and if the intended word is not the first word in the Selection List, activates the Select key additional times to move through the list to choose the intended word. Once selected, the user may begin entering the next word and the selected word will be output. As explained below, there are also other modes in which the Select key can operate.

6.1 WORD LEVEL DISAMBIGUATION

The text entry layer contains multiple letters on each key and multiple words typically match the sequence of keys entered. The power of this entry technique lies in word level disambiguation (WLD). As the keys are tapped, WLD identifies possible candidate words that match the input sequence and also considers the context of the text already entered to create a Selection List containing the most likely words the user may be trying to enter.

While WLD does require a literate user, the benefits are considerable. In general, less than one key per character is required to access all words of a language, allowing information rich content to be generated efficiently. Communication partners can be kept engaged. Poetry and technical papers can be written. A higher selection rate of keys occurs due to fewer keys and minimal cognitive load.

6.2 Text Entry Rate

When using a full keyboard, a user will enter one key press per character output to the screen, ignoring use of the shift, control, alt, and other function keys. Using only 8 keys, a user will still enter approximately one key press per character. Over time, this will improve to less than one key press per character as the language model within the disambiguation engine learns a user's writing patterns, and begins to predict the intended word before it is fully typed.

You are correct if you point out that the user must use the select key to choose the word from the Selection List, consuming key presses. Surely this requires greater than one key press per letter overall. Well, yes and no. For some words, the key count will be greater than the number of letters output, but in others it will be less. Depending on the configuration, we have measured an average 1.003[6] keys per character with a low cognitive load and no prior language model training to as low as 0.64 keys per character with a commensurate increase in cognitive load. Furthermore, since spacing is automatic, it is not even an "extra" key activation (see below). The following sections discuss the techniques we recommend, both from the disambiguation engine and support software to reduce the key entry count.

6.3 Auto Spacing

When entering a word, the user must choose the word from the Selection List. Often[7], the desired word is the first in the list, so one press of the Select key chooses the desired word. The user need not enter a space key, a long press on our keyboard. Instead, after selecting the desired word, beginning the next word causes the selected word to be output with appropriate spacing. No explicit space key press is necessary. So, in the best case, no extra key presses are required for selecting a word because the one key press to select the word is compensated for by the free (no key press) space provided.

6.4 Shift/CAPS

The word prediction engine provides auto capitalization for most detectable needs. Proper names, beginning of sentence, and acronyms are all detected and capitalized without user input. The user does have the ability to capitalize other words for emphasis using the Shift key. This is a tristate modifier. Pressing the key transitions to the next state: Shift, CAPS LOCK, lower case, providing the user control of the generated text.

[6] In the original research by King, Kushler and Grover from the mid 1990's, this number was 1.0051 keys per character, dependent on word level disambiguation with known word frequency data. The number listed is as measured using word level disambiguation using modern linguistic engines rather than word frequency.
[7] Between 95% and 99% of the time, depending on user configurable options, the first word in the Selection List is the user's desired word following entry of all the letters in the word. See One Key Select for more detail.

6.5 Spelling Correction

The word prediction engine not only identifies words that match the key input sequence, but also recognizes other common errors a user might make while entering text and offers corrected words as well. The disambiguation engine identifies character substitution and transposition errors as well as omitted and inserted key corrections. The number of these corrections can become quite large, which impacts the Selection List length.

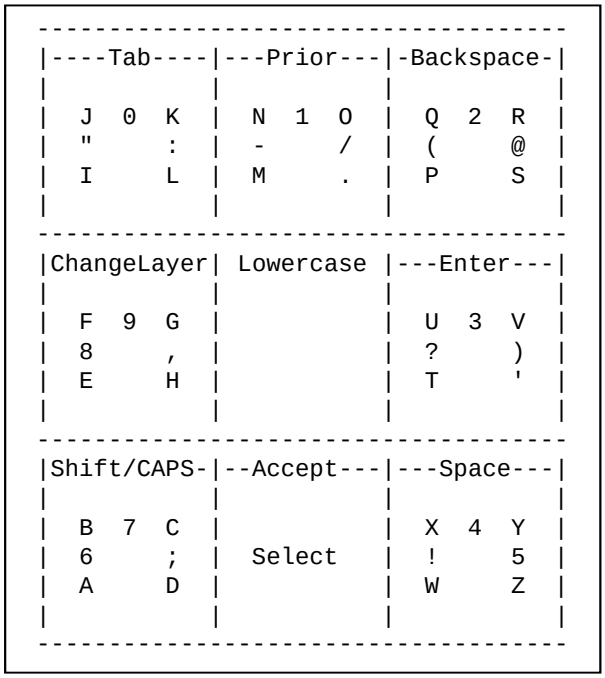
The user may identify the types of errors they are unlikely to make. These are then filtered out in order to reduce the number of corrected words added to the Selection List (see below). For example, someone who is an excellent speller might filter out transposition and substitution errors while someone who has difficulty accurately entering keys may want to retain the omitted and inserted key corrections.

Properly tuning (or turning off) spell correction can significantly reduce the numbers of words added to the Selection List.

6.6 JustType Text Entry – Alternate Arrangement

The JustType text entry layer introduced above is referred to as the BRZ arrangement, where the letters appear to be randomly assigned to keys. There is some logic to the arrangement, but it has nothing to do with typing 'too fast' like the QWERTY keyboard arrangement. Instead, the BRZ arrangement, referring to the letters associated with one of the keys, is found to be the most efficient arrangement of letter/key assignments for entering text based on an English language word list by minimizing key presses.

This arrangement tends to minimize the number of frequently used words that are entered by the same key sequence, which minimizes the number of times that the Select key needs to be activated to access a word. When the underlying language model is working effectively, the word list should end up being sorted according to the likelihood that each word is the intended word given the current input context, which likewise should tend to minimize the number of Select key activations required. Based on measurements, the disambiguation engines do not currently return results in pure word frequency order. Also, they insert additional word completion words that also affect the minimization of select key presses. While not specifically optimized for languages other than English, the BRZ arrangement is reasonably efficient for other languages based on the Latin character set.



The ABC arrangement shown here is an alternate assignment of letters to keys. It is just like the BRZ text entry layer except that the letter assignments are in alphabetical order. Users can select which arrangement they prefer to use.[8] While this arrangement is not as efficient for text entry as the BRZ arrangement, it may make it easier (or simply seem easier) for some users to locate the correct key for their desired letter. The BRZ arrangement, however, can be learned reasonably quickly by most users and is recommended as the default.

When measured using the language model, the BRZ keyboard requires ~1% fewer KPC than the ABC keyboard when Two Key Select is active and ~1% more KPC when One Key Select is active. This difference is minor and shows the power that language models provide since the original research was done.

7 JustType Selection List

As keys are entered, a Selection List is continuously presented with possible words based on the key sequence entered thus far. The user may choose a word from the list with the Select key. Successive taps of the Select key moves through the list, highlighting the referenced word. Activation of any other key causes the highlighted word to be output.

Different classes of text objects that can be presented in the Selection List include:

7.1 Next Word Predictions

When no keys have been tapped, the Selection List contains words predicted by the language model for the current context. These Next Word Predictions improve as the language model adapts to the user's writing patterns. Note that this requires the user to indicate that the previous word has been selected with a "Final" selection so that the Selection List can be re-populated with Next Word Predictions as occurs with Two Key Select discussed below. The "Final" selection is obvious when the user enters any white space character unambiguously: space, tab or carriage return.

7.2 Complete Words

Complete words are words in the Selection List for which every letter in the word has been entered with one key press per letter.

[8] Norman and Fisher (1982) found that people do not know the alphabet as well as it was assumed and the alphabetical layout involves a mental as well as a visual search compared to the purely visual search for random arrangements such as the QWERTY layout.

7.3 Predicted Words

Predicted words are words in the Selection List entered with one key press per letter, where fewer than all of the letters have been entered. The word prediction engine uses the current context, key sequence, a language model and the user's text patterns to anticipate words the user is likely trying to enter. Selecting these words potentially generates words with less than one key press per letter.

7.4 Explicit Word

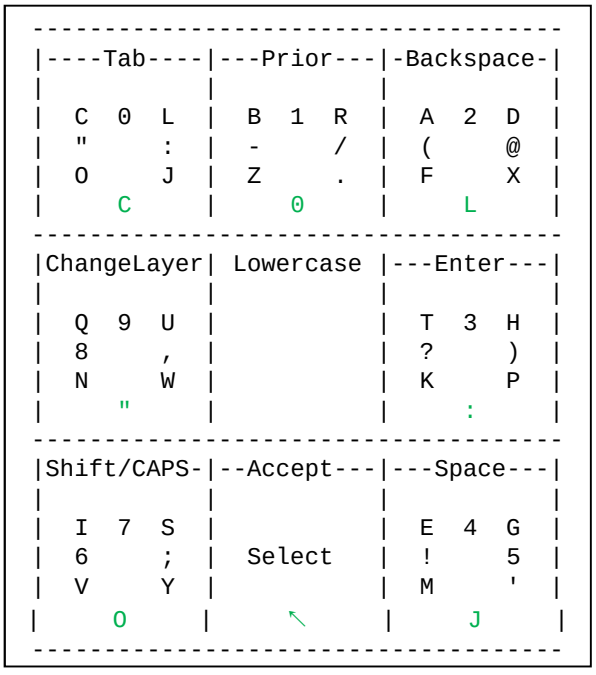
The word prediction engine is only as powerful as the word database it uses. Most all users will have words they use that are not in the database: names, places and events are common examples of missing words. Also, entering numbers and punctuation does not benefit from the word prediction engine.

To enable the entry of words that are missing from the database, numbers or punctuation, JustType also interprets each input key sequence as a sequence of pairs of key taps that each unambiguously specify a single character. Each pair of key taps is interpreted as a first key identifying the key on which the desired character appears, and a second key identifying which of the seven characters was intended. Following the first key tap, the ambiguous keys are re-displayed showing the unambiguous character that will be generated at the bottom center of the key if that key is selected as the second key tap, as seen in the figure. These key pairs provide simultaneous unambiguous input which constructs the Explicit Word for the Selection List.

Example

To unambiguously enter the word "cak" (the initials of one of the authors):

- tap the CLOJ key to indicate you want one of the characters on that key
- tap the CLOJ (upper left) key to indicate the desired character 'c' is the upper left character from the CLOJ key
- tap the ADFX key
- tap the CLOJ key for upper left character 'a' from the ADFX key
- tap the THKP key
- tap the ISVY key for lower left character 'k' from the THKP key



Recommended UI – Keyboard

Because creating an Explicit Word is an infrequent task, the user can easily get confused with where they are in their construction. To reduce confusion, the bottom row center position of each key should indicate what each key will contribute. In our example for entering "cak" unambiguously, nothing is shown before the CLOJ key is tapped; after, the keys would show the unambiguous character associated with each key as the second keystroke of an unambiguous pair. This is shown in the figure with the green characters. These characters should be

subtle so that their changing with each key tap is less distracting, but sufficiently visible so that they support the user when needed.

The Explicit Word appears as the last entry in the Selection List and may be selected. Once it is selected, it is added to the stored word list so that the user may enter it in the future with 1 key tap per letter. As the Explicit Word is the last entry in the list, it may be accessed quickly using a long press on the BRZ key for the "Prior" function to move backwards through the Selection List.

Recommended UI – Selection List

The on screen keyboard display should provide feedback to the user that supports correct construction of the Explicit Word. The Explicit Word should always appear distinct, perhaps shown in the Selection List in a different color (matching the keyboard hint color), so that the user can see its construction when they are entering unambiguous text and ignore it when they are not.

The Explicit Word should appear at the end of the Selection List so that it can be easily accessed using a long press to access the "Prior" function. The "Prior" function moves backwards through the Selection List. One long press to access "Prior" will select the last entry in the Selection List. When using Two Key Select for the Selection List, the Explicit Word may be located either at the end of the list or at position 6 (the 7th entry, which is the "last key on the first page of choices) so that it can be accessed quickly with two key taps.

Numbers, Symbol Strings and Selection List Position

Numbers and symbols are often desired, but they are difficult for the disambiguation engines to recognize based on the letter/number/symbol ambiguity on the JustType keyboard. When many numbers/symbols are desired, it may be best to switch to the Symbols/Numbers Layer. When only one or two are desired, they can be entered using unambiguous text in the Explicit Word.

Diacritic Characters

This description for unambiguous text entry excludes discussion of unambiguous entry of diacritic characters. As most languages besides English make use of diacritic characters, a third key tap may be required for unambiguous entry. Should the second tap identify a base character accepting various accents used by the active language, a third tap identifies the appropriate form of the character. The same approach is used, where the various diacritic forms are displayed at the bottom of the unambiguous keys. The demonstration software does not yet support this functionality. Note that in normal ambiguous text entry, the database includes the correct spelling of all words with diacritics, and the standard JustType keyboard can be used to efficiently enter text with no additional complexity or effort on the user's part.

7.5 Rote Words

A subset of approximately 400 of the highest frequency words can be added to the Selection List as the first entry for each of the first three key presses of a word. After one key tap, the most frequent word in the language for the current key sequence is shown as the first word in the Selection List. In English, for example, "the" is always displayed as the first Selection List word after a first tap on the THKP key. After 2 key taps, 49 of the next most frequent words are shown as the first word after each of the possible

two-key JustType key sequences. In every case, the initial letters of the displayed word must match the key sequence for which it is displayed. After 3 key taps, 343 of the next most frequent words are shown as the first word after each of the possible three-key JustType key sequences. Note that all of these sets of words are disjoint: once a word is displayed as the default (e.g. “the” after a first tap on the THKP key), it is not displayed as the default even when a subsequent key still matches the word (e.g. a second hit on the THKP key would show “this” rather than “the” even though “the” is a higher frequency word than “this”). The “demoted” Rote Word (“the” in our example) still appears in the Selection List as long as it still matches the current key sequence, but not in the first position.

These Rote Words are powerful as the user quickly learns that these very common words are always in these positions. They learn the patterns that generate these words and access them very quickly, saving time. This can be significant, since a relatively small number of high frequency words accounts for a disproportionately large percentage of the words generally used in communication. The choice of whether or not to display Rote Words in this fashion is a feature that can be configured by the user.

Analysis

The power of Rote Words comes from the key input patterns the user learns for generating these words and the repeatability. The current language models are very good at identifying desired words very quickly and allowing access to these words in the Selection List. The distinction with Rote Words is that they always are accessible with the same input key pattern, and once learned, do not require the user to attend to the Selection List. This can significantly speed up text input in a way that is not captured by the KPC metric. With the language model, they may not appear consistently in the same position in the Selection List and attention must be spent scanning the list for the word. The KPC for the Rote Word option:

KPC	Rote Words	Select Style
0.863 KPC	Enabled	1 key, Rote or all keys *
1.003 KPC	Disabled	1 key, all keys *
0.869 KPC	Enabled	1 key, List Length 7 †
0.848 KPC	Disabled	1 key, List Length 7 †
0.637 KPC	Enabled	2 key, List Length 7 †
0.635 KPC	Disabled	2 key, List Length 7 †
0.629 KPC	Enabled	2 key, List Length 14 †
0.627 KPC	Disabled	2 key, List Length 14 †

* These entries measure the authors recommended entry technique of entering the entire word or the Rote Word keys and then selecting the word. For most users, this is faster. Rote Words provide for 14% fewer key entries.

† These entries are made with the user scanning the Selection List of length 7 or 14 (as noted). Fewer keys are required with an increase in cognitive load. Also note that the

increase from 7 to 14 Selection List items provides negligible improvement with a doubling of the list to be scanned.

This data indicates that there is a distinct advantage using Rote Words when 1 key select is active. When 2 key select is active, there is minimal improvement in the KPC figure, but the potential for the user to more quickly enter Rote Words once the input patterns are learned is still there. See One Key Select and Two Key Select below to distinguish these selection modes.

Note that Rote Words are not a feature of the disambiguation engines. Rote Words are added in the JustType software. The rote words were chosen from a word frequency list generated from a very diverse corpus. In theory, the user would enter a rote word 55% of the time, based on this list. We measured rote word use of 60% using our test corpus. These numbers are certainly variable, depending on the subject matter and style of the user's writing.

7.6 Numbers and Punctuation

The JustType text entry layer is optimized for entering text efficiently. Words comprise the majority of text and JustType has been optimized for word entry. Numbers and punctuation, however, are vital for enabling unconstrained text entry. JustType enables access to numbers and punctuation without too much pain.

7.6.1 Rote Punctuation

Similar to rote words described above, rote punctuation make access to the most common punctuation marks[9] predictable. This allows most punctuation to be entered without need for the user to confirm the location of the punctuation in the selection list – it is always in the same place. The user quickly learns the input patterns to access the most common marks.

The following punctuation marks are accessible using rote punctuation: comma, period, question mark, exclamation mark, semi-colon, double quote, and commercial-at. They are accessed with 2 or 3 key taps, depending on whether rote words are disabled or enabled. Following the first key tap for a word, JustType adds the most common punctuation mark from the key as the first (rote words disabled) or second (rote words enabled) word in the selection list[10]. So access to the punctuation is to tap the key with the desired punctuation mark followed by one or two select key taps.

Analysis

The value of punctuation in text perhaps depends on the purpose of the text. For AAC, the value may be low compared to the effort required to enter the characters. For letters to loved ones or formal documents, punctuation has a greater impact and thus value. The goal for JustType concepts is to support punctuation as an important feature to create appropriate text for the task at hand.

[9] Frequency analysis of punctuation marks varies significantly and is strongly dependent on the corpus analyzed. With the popularity of SMS, tweeting and other now common communication pathways, we have chosen to focus on text input for communication and grammar rather than metadata and tagging.

[10] In the case where a single letter word, e.g. "a" or "I", is on the key, the rote punctuation appears after the complete words. When defining the keyboard layout, one consideration was to place less frequently used punctuation on keys with single letter words.

Up to this point, the metrics presented in this document have been for pure text entry, ignoring punctuation. The table in Section 7.5 provided a number of metrics for rote words in order to show the power of rote words in various situations. That table is duplicated here with the addition of metrics that include punctuation:

w/o Punctuation	w/ Punctuation	Rote Words	Select Style
0.863 KPC	0.938 KPC	Enabled	1 key, Rote or all keys *
1.003 KPC	1.038 KPC	Disabled	1 key, all keys *
0.869 KPC	0.943 KPC	Enabled	1 key, List Length 7 †
0.848 KPC	0.892 KPC	Disabled	1 key, List Length 7 †
0.637 KPC	0.710 KPC	Enabled	2 key, List Length 7 †
0.635 KPC	0.706 KPC	Disabled	2 key, List Length 7 †
0.629 KPC	0.702 KPC	Enabled	2 key, List Length 14 †
0.627 KPC	0.700 KPC	Disabled	2 key, List Length 14 †

* These entries measure the authors recommended entry technique of entering the entire word or the Rote Word keys and then selecting the word. For most users, this is faster. Rote Words provide for 14% fewer key entries.

† These entries are made with the user scanning the Selection List of length 7 or 14 (as noted). Fewer keys are required with an increase in cognitive load. Also note that the increase from 7 to 14 Selection List items provides negligible improvement with a doubling of the list to be scanned.

The most common punctuation can be accessed with 2 or 3 key presses, depending on whether rote words are disabled or enabled, respectively. Based on our corpus text, punctuation entry consumes the following percentage of key taps:

7.99 – 10.3% of keys	Rote Words enabled	1 Key Select – 2 Step Select
3.37 – 10.1% of keys	Rote Words disabled	1 Step Select – 2 Key Select

As shown, if rote words are disabled, punctuation can be relatively efficient since the rote words are not blocking the punctuation in the Selection List for one step select. We encourage use of rote words for the general reduction in cognitive load and speed improvement they provide, unless the user has a strong preference not to learn to use them effectively.

7.6.2 Numbers and Uncommon Punctuation

Similar to explicit words, numbers and punctuation can be entered using unambiguous text entry. This is fully described in section 7.4.

7.7 Corrected Words

Wouldn't the world be grand if we were all perfect spellers? As we are not, the word prediction engines also provide spell corrected words. They look for the typical spelling mistakes: transposed letters, missing letters, extra letters, phonetic errors and just plain mistakes. These words are added to the selection list to accommodate the user.

8 Choosing from the Selection List

There are two methods for choosing a word from the Selection List: one key and two key select. There are advantages to each method.

8.1 One Key Select

One key select treats the Selection List as a linear list. When a desired word is identified in the list, the user taps the select key one or more times until the desired word is highlighted. The first tap marks the first word, the second tap marks the second, etc. The word may then be accepted, but need not be accepted. With one key select, a highlighted Selection List word will automatically be accepted and output whenever any key other than the select key is tapped, thus entering the first character of the next word. More than 95 – 99%[11] of the time, after the user enters the entire word, the first word in the list is the desired word and the Select key need be tapped only once.

For example, if I enter the word "bright", this table shows the Selection List following each key:

	'b'	'r'	'i'	'g'	'h'	't'
1	be	bring	British	brie	bright	bright
2	.	Brian	bring	brig	Brighton	Brighton
3	'	Brown	Brian	brim	brightly	brightly
4	-	Bro	bringing	bright	brightness	brightness
…	…	…	…	…	…	…

Notice that "bright appears as the fourth entry after entering the 'g' key. The user could tap the "Select" key 4 times to choose, the word "bright", but that would require 4 taps. Entering the 'h' key moves the word to first in the list, requiring only one tap of the "Select" key. Also note that completing the word with the 't' key also has "bright" as the first word in the list.

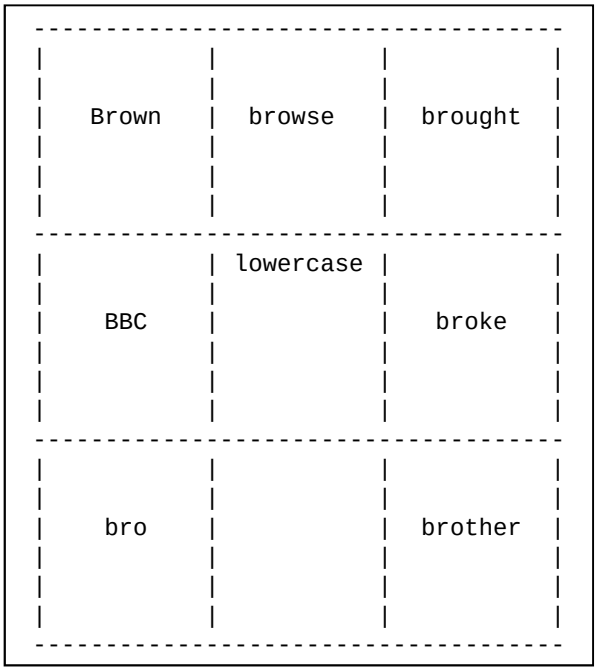
[11] Rote Words reduce the percentage of words accessed in the first position in the Selection List as they occasionally block another short (2 or 3 letter) word. Overall, however, the value of Rote Words is that they significantly reduce the number of characters entered by up to 14% and they speed input as the user learns the key input pattern for entering these frequent words. See Rote Words for more information.

% 1st in List	Rote Words	KPC	Entry Style
99.47%	Disabled	1.003	All Chars
95.5%	Enabled	0.863	Rote Word or All Chars

Advantages: The one key select method supports minimizing the cognitive load on the user. For users with less difficulty in entering keys, we recommend they enter the complete word, ignoring the Selection List until the word is completely entered. This avoids spending a significant amount of time continually checking for the word in the list, which is generally slower than just entering the full word.

8.2 Two Key Select

Two key select is a much faster way for moving through the Selection List when the desired word is selected from a position lower in the list with some frequency. With each press of the "Select" key, seven words from the Selection List are loaded onto the seven keys, replacing the ambiguous keys. If the desired word is not among the first seven words, each additional tap of the "Select" key loads the next seven words onto the ambiguous keys. Unlike one key select, two key select requires a minimum of two key taps to choose a word: the first tap (on the Select key) loads the ambiguous keys with words and the second tap chooses one of the words.



Following the selection of the word from the keys, the keys return to their normal state for ambiguous character entry. Also, as there are no ambiguous taps active, the Selection List is populated by the language model with likely Next Word Predictions based on context. These words can be accessed without any ambiguous key taps, and require only the two or three taps to select the word.

Based on a corpus of text used for Augmentative and Alternative Communication (AAC)[12], the following table identifies how often the next word is predicted based on the Selection List size:

List Size	3	4	5	6	7	8	9	10	11	12	13	14
% Predicted	35.8	39.4	42.6	44.8	46.9	48.9	50.1	51.2	52.7	53.8	54.7	55.5

This entry method enables text entry at fewer than 0.7 keys per character, but at the cost of a large increase in cognitive load. The user must scan the Selection List for their desired word with each key tap, significantly reducing text entry rates for most users. This is certainly not required of this mode, but is required to achieve this level of efficiency in keys per character. Note that the returns diminish quickly as the length of the Selection List increases while the cognitive load increases. Since the efficacy of this approach depends entirely on the time demands of the cognitive load, quantitative studies would

[12] The corpus used for testing JustType comes from A Crowdsourced Corpus of AAC-like Communications. Approximately 90% of the corpus was used to train the language models of the disambiguation engines we tested and 10% was used for the final test performance.

be needed to try to determine an optimal list size. See “How many words should be displayed in the selection list?” for further recommendations.

Advantages: The two key select method supports the lowest keys per character rates because it enables moving through the Selection List quickly and also enables access to word predictions following each word entry, at the cost of increased cognitive load. This method may justify itself for users where the physical difficulty of accessing keys is profound and the increased cognitive load is preferred. We have also found that some users are not concerned with their input rate and like seeing the system anticipate their words for them.

9 Selection List Sorting and Filtering

The original research into this style of keyboard provided a Selection List where the words were listed based on the frequency of use of the word within a large corpus of data without looking beyond word frequency. Spelling errors were not generally accommodated and word completions tended to be buried deep in the list.

Disambiguation engines today tend to utilize language models that no longer contain raw monogram word frequency data. Instead, they have a general knowledge of roughly how often a word is seen in a language and they look at and maintain information about a user’s word patterns and sort the Selection List based on probabilities of the likely word based on input thus far. Spelling errors are recognized and spell corrected words are also listed. Predicted words jump ahead of complete words based on the likelihood the engine anticipates for this word. This works well for situations where the user can easily interact with the Selection List when their desired word is not near the top of the list (though this does not take into account the cost of visually scanning the list to find the word). Mobility impaired users benefit from a Selection List word order that caters to their specific needs with a little more predictability.

9.1 Sorting

The JustType software recognizes the unique needs of mobility impaired users and rearranges the Selection List based on options the user can specify. The disambiguation engines generally provide full words, spell corrected words and word predictions in an order determined by the language model’s predictions of the likelihood of each word in the current context. JustType adds Rote and Explicit Words to the list as well, then rearranges the list according to the user options. The default word order from JustType is:

- Rote Word (if enabled and if defined for the current sequence)
- Complete Words
- Predicted Words
- Spell-corrected Complete Words
- Spell-corrected Predicted Words
- Explicit Word

The target audience for JustType is a literate user with mobility issues. The authors consider this Selection List order to be an ideal compromise considering both efficiency and cognitive load, but the

best configuration will vary for different users. For someone who spells well, spell corrected words are still available, but much further down the list. Others may frequently transpose letters but never mistakenly press one key instead of another. Each of these common errors can be optionally adjusted to move those corrections further up the list.

Word predictions are moved to near the end of the list so that they do not obscure complete words that a user may be intending. This avoids the user having to skip them to get to a complete word, possibly requiring multiple Select Key activations to get beyond the blocking words. If the predicted word is truly the intended word, adding another letter or two of the word will move it up the list significantly.

The user also has the option of just trusting the disambiguation engine language model. With this option enabled, the Selection List word order is:

- Rote Word (if enabled and if defined for the current sequence length)
- Word List from Language Model
- Explicit Word

9.2 Filtering

In the next section, Selection List and Cognitive Load, we discuss the issue of spending too much time scanning the Selection List for the desired word rather than just entering more keys for the word. This can be a significant time drain. If the desired word is deep in the Selection List, not only is the search time consuming, but accessing the word deep in the list will take several Select Key activations. Once one has paid the price of visually scanning the list, getting to an entry late in a list on a touch screen is as simple as tapping the word on the screen, but not so when the user must tap the Select Key multiple times to access the word due to mobility limitations.

For these cases, it is often faster (fewer key entries) to add one or more characters to the input sequence such that the word prediction is much nearer the top of the list. To this end, JustType filters out of the Selection List any word predictions that require more Selection Key activations to access the word than remaining keys to completely enter the word. Words that are significantly far down the list will move up faster through additional key input.

When we filter any prediction or completion words that require more Select Key taps to access the word than remain to complete the word, the results are significant. The key tap savings are 12 – 15% of keys. This is because the word moves very quickly up the list with each additional key tap.

9.3 Word Anchoring

Another technique for accessing words more quickly where the amount of ambiguity creates longer selection lists is word anchoring. If I want to enter the word “interestingly”, I can enter most of the characters to get the word near the top of the Selection List and choose it. But I can also inform JustType that an entry early in the list has the same letters as the word I am seeking.

Using “interestingly” as my example, after I enter the first two keys for ‘in’, the Selection List contains “in” as the first entry (“in” is the Rote Word for that key sequence) and “interestingly” as the 25th entry. If I tap the Select Key to choose “in”, and then long press the Select Key, this informs JustType that I have not finished the word but I am marking “in” as an anchor for the word I do want. JustType then

only shows words that start with ”in” in the Selection List, rather than any of the other possible two-letter combinations matching the first two ambiguous keys. I can then continue to add more keys and possibly more anchors to get my desired word.

The final tally of key taps and long presses, without painstaking detail, has “interestingly” entered using 2 taps and 2 anchor entries (long press) and 3 Select Key taps to get the word. This approach to word input, while efficient, will likely only appeal to a certain type of individual, and certainly imposes a significant cognitive load.

10 Selection List and Cognitive Load

The Selection List is a necessary element of a disambiguating keyboard, yet the best use of the Selection List is a much debated topic. Display the list, don’t display it, how many words do we display, etc.

With a QWERTY keyboard, most users can type their desired text as they compose it. Most cognitive effort goes into composing their thoughts with a small effort towards correctly spelling the words. The actual words wind up on the page with no additional cognitive effort.

With a disambiguating keyboard, composing text is still most important, but this keyboard requires adding cognitive load to ensure that each word output is correctly chosen from the Selection List. The effort expended in locating the word can vary from small to large, depending on the user’s strategy as well as any user physical limitations for input control.

A goal for many mobility-impaired users of this keyboard will be to achieve the highest possible input rate. They may use the keyboard to control an AAC[13] device and want to minimize the time it takes to compose what they want to say. One user will find that only looking at the Selection List when they are done entering all the letters of the word is best because then the word is almost always (95-99%) at the top of the Selection List, and they can enter the next character of a word faster than they can scan the Selection List for their word. Another user may find that it is faster for them to scan the list each time because their difficulty in entering a key takes them longer than scanning the list. Note, however, that finding the word deep in the list and using multiple selects to choose the word is generally much slower than entering one or more additional keys and having the word move up in the list.

Compounding the problem of “the best Selection List strategy” is that language models keep improving and are predicting the desired word very often within a few key taps if not immediately. This encourages a user to frequently switch tasks between entering keys and searching for their word. It also has the potential to change what a person actually writes. We have found that when we look at predicted words, we often find not the word we were going to enter, but one “just as good” and select it. Language models can have an impact on what users choose to write and speak.

With the relationship of Selection List and cognitive load just described, here is some guidance we suggest for various choices for the Selection List:

[13] Augmentative and Alternative Communication

10.1 Should I use one key or two key select?

Certainly, it depends on the user and their goals. If we focus on input speed, more users will likely benefit from using one key select and extended practice to develop efficient input patterns. If we focus on perceived ease of use, two key select will likely be the choice. The language models can achieve better than 50% word prediction for general use and the two key approach allows access to the predicted words before the first character of a word is entered. Also, moving through the Selection List is faster with the two key approach. These benefits come at the cost of increased cognitive load.

This would make an excellent study to determine what method and training would benefit specific user skills and yield the greatest efficiency in generating the intended text.

10.2 How many words should be displayed in the Selection List?

The ideal number of entries shown from the Selection List is certainly dependent on the user's abilities and goals. At one extreme, no entries could be shown until the user taps the Select key and the whole list is shown. At the other extreme, the entire list could be shown. Our recommendation is a mix, and preferably configurable by the user.

During word construction, the list should consist of 4 – 7 entries[14]; 4 for one key select and 7 for two key select. The first entry is the Rote Word if the feature is enabled and a Rote Word exists, the last entry should be the Explicit Word entry, and the remaining entries are complete word, rote punctuation, predicted word and corrected word entries as prioritized from the word prediction engine. The words should be displayed so that they can be read quickly, preferably in a vertical list, and located close to the keyboard to minimize eye travel. An exception to these recommended list size limits is to expand the size of the list as needed to show all of the possible complete words and word corrections at each step. Word predictions should only be shown if space in the list is available. There is seldom a large number of complete words, particularly when using an optimized BRZ key layout.

10.3 Efficiency vs. User Needs?

The authors have significant experience with creating alternative keyboards and have always strived to make text entry very efficient based on constraints of the environment. On occasion, we have been blinded by our goal for efficiency at the expense of not meeting a user's needs. We would argue that a user should want to do things in the most efficient manner possible. But we lose sight of the fact that a user's needs, a user's goals, need to take priority over our personal goals for the user.

So any implementation of JustType should leave as much control and configurability to the user without becoming unwieldy. Not all users will make the same choices, and that is a good thing!

[14] It is important to note here that these numbers are meant to limit the Selection List size in the general case. In some situations, the limit needs to be larger to include all complete words, corrections and explicit words. For two key select, the list could be multiples of 7, adding predicted words to reach the next multiple of 7, as the access method moves through the list in increments of 7.

11 (Computer Control Too!)

JustType makes use of multiple keyboard layers optimized for various tasks. Thus far, we have discussed the text input layer. Additional layers provide other task specific input. The layers may switch under user control or automatically when the software detects a user action that suggests a different layer. The suggested keyboard layer definitions can be freely modified, augmented or edited to accommodate the needs or preferences of the end user.

11.1 Switching Layers

Before diving deeply into this topic, it is important to point out that for many individuals, text generation for communication is by far the most important function of the keyboard, and a single layer provides this functionality. In order for a keyboard with 8 keys to completely replace a keyboard with 100+ keys, the keyboard must be constructed in layers. Each layer generally provides functionality to support different tasks: text entry, editing, navigation, etc. Each layer also must provide a means to switch layers. This is typically done with the secondary action on a key.

While our focus is demonstrating a functional keyboard using 8 keys, another way to switch layers would be to provide additional keys beyond the 8 for layer transitions to the most frequently used layers. For example, we demonstrate JustType using a gaming controller with a joystick and multiple support buttons. While we can make all transitions with the joystick, we also have mapped actions for some of the support buttons. This enables versatility for users able to manage additional control inputs.

11.2 Change Layer

The Change Layer provides quick access to switching to any other layer.

Many users with mobility issues will use a speech output communication device and this layer shows integration of that functionality here. In practice, integration of the communication device would be done on a modified text entry layer to avoid a layer transition on each utterance.

Speak Sentence

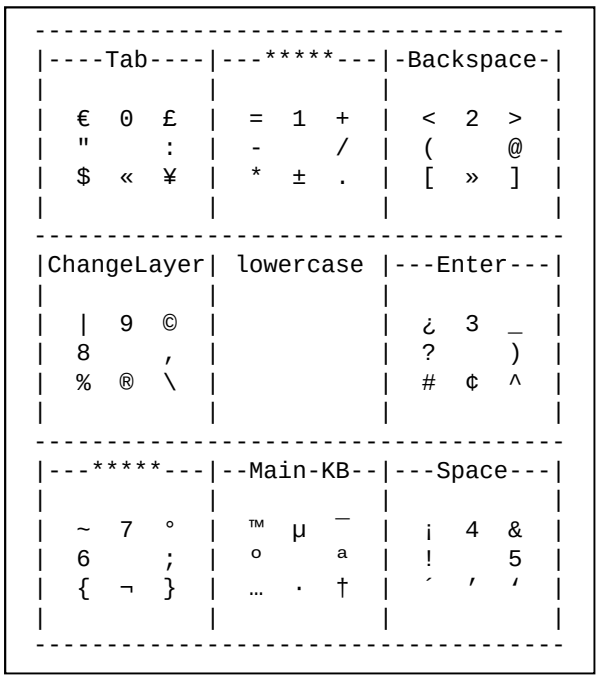
Send the text for the current sentence to the speech engine for vocalizing.

Speak Buffer

Send the text for the current buffer to the speech engine for vocalizing.

Clear Buffer

Push the current buffer to the buffer history and present an empty buffer.



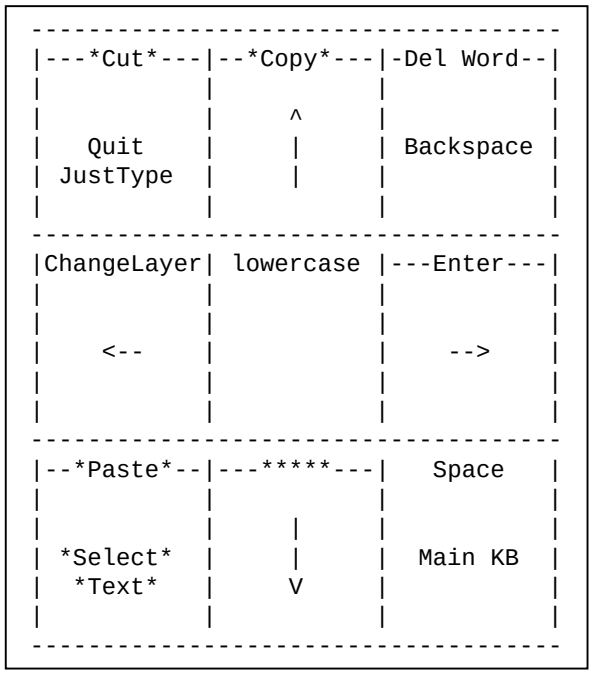
11.3 Symbols/Numbers Layer

The Symbols/Numbers layer provides access to numbers as well as a variety of symbols. Each character is entered unambiguously, requiring two key selections for each character. The first key tap identifies the key on which the character resides and the second key tap identifies which of the eight characters is desired.

Note that when a character or function occurs on multiple layers, it is best to keep the character/function in the same position. The numbers and symbols that are shown on this layer and also appear on the text entry layer (both ABC and BRZ layouts) are on the same keys and at the same positions on both layers.

11.4 Edit Layer

The edit layer provides the ability to navigate the text field efficiently and to provide moderate utility in modifying the text.



Cut

Remove selected text from input field, if any.

Copy

Copy any selected text from input field, if any.

Paste

Paste any previously cut or copied text to the input field, if any.

← (Left)

Move the cursor one character to the left of the current position.

→ (Right)

Move the cursor one character to the right of the current position.

↑ (Up)

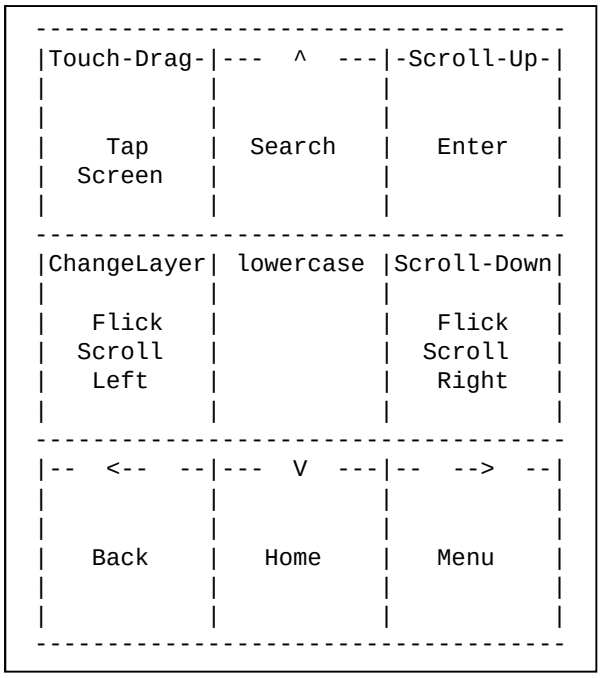
Move the cursor up one line from the current position. If no lines above position, move cursor one word left.

↓ (Down)

Move the cursor down one line from the current position. If no lines below position, move cursor one word right.

Select Text

Toggle between selecting text and not selecting text with the movement of the cursor.



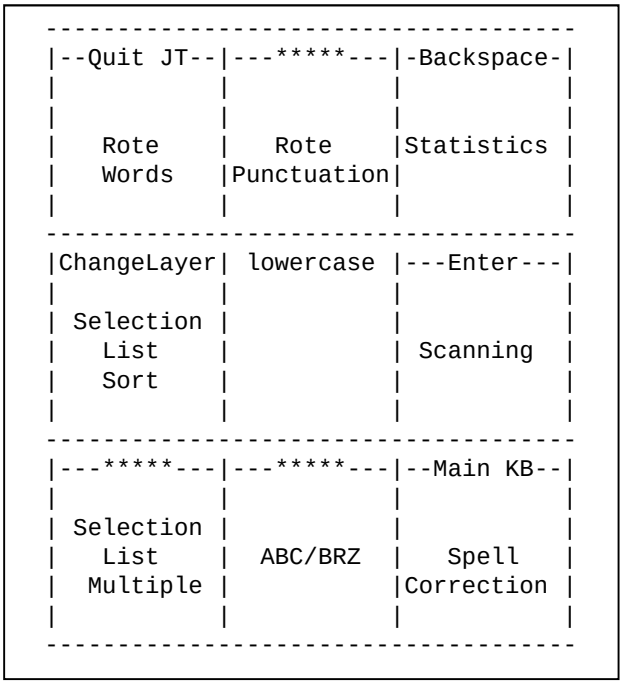
11.5 System Control Layer

The system layer is automatically entered when not in a text entry field, such as when the keyboard is dismissed. The keyboard could be shown transparent to aid the user in navigating over the display.

The system layer should support the most common screen interaction support for the specific system: Windows, Android, iOS, etc. Those shown here are one example for useful functions on this layer: tap, double tap, long press, scroll, pan, flick, 2 finger tap, 2 finger scroll, pinch, spread, rotate.

11.6 Options Layer

The options layer(s) provide the selectable user features for JustType. Those shown here are a small set of what would be useful for a fully implemented computer access keyboard.



12 INPUT CONTROL

There are a variety of input devices that are able to effectively support an 8 key JustType keyboard. Physical key activations may be accomplished through a touchscreen, keypad, joystick, head/eye tracker, scanning switch or other appropriate device. The choice of input device should take into consideration use of the device for long periods. While a user may be capable of using several of these devices, the best choice is the one that minimizes their cognitive and physical effort. Reducing the total number of keys required to only eight means that these alternative input devices require less fine motor control.

For most of the devices below, a ninth key could be used for users who have difficulty distinguishing between a tap and a long press. In those cases, the ninth key is activated to indicate the next key tap is to be considered a long press. For the square layout, the ninth key is recommended at the center of the square. For the linear layout, the ninth key is recommended at the end of the line of keys.

Unless otherwise noted, the square key layout is generally assumed.

12.1 TOUCHSCREEN

A touchscreen is the most configurable of the devices for input. The touchscreen may be configured uniquely for an individual to provide eight locations for the keys that are easy for the user to activate. They can be spaced close together, far apart, or in unique locations easily accessible to the user. It is also possible to create a screen guard to prevent accidental input or aid in key access.

The touchscreen input easily enable custom layouts for the keyboard.

12.2 KEYPAD

A keypad may be configured to control the eight JustType keys. A keypad could be a full keyboard, a numeric keypad, a phone keypad, eight unique switches, or other keypad device. The keypad or multiple switch input would need to be mapped to the eight JustType keys. With a mapping table, it is possible to have multiple keys that would map to any of the eight keys.

12.3 JOYSTICK

We like to say, if you can control a wheelchair, you can use a joystick for JustType input. A single movement of the joystick in any of eight directions[15] and a return to center is a key activation. Holding the joystick movement in any of the eight directions for a threshold time before return to center is a key long press.

When we first integrated a joystick with our JustType demonstration software, we thought we would need some time to filter all the joystick messages to identify key activations, and we planned for it to take some time. Fortunately, we happened across a tool, Xpadder, which allows you to map keyboard keys and mouse button actions to your game controller buttons and joysticks. In a matter of hours we had Xpadder configured for our joystick and "driving" JustType very effectively. Integration with an

[15] For reference, we use the compass points to identify specific keys. The key in the middle column of the top row is N – North while the key in the left column middle row is W – West.

actual wheelchair joystick will require further device driver work as they do not appear to the operating system as a gaming controller.

One thing we did learn was that using a joystick blindly with JustType was somewhat frustrating. We realized that when using a joystick to control a wheelchair, the chair motion provides feedback to the user and they course correct while in motion. Our input accuracy improved when we added key highlighting to the screen to indicate which key the joystick was referencing. Otherwise, it was easy to mistakenly enter the wrong key.

12.4 Head/Eye Tracking

A head tracker monitors the position and orientation of a user's head to provide a control input. An eye tracker is a device for measuring eye position, position, and/or point of gaze. Either a head tracker or an eye tracker could be configured to activate the eight JustType keys. The small number of keys allows each target to be significantly larger and correspondingly easier to select, so that in most cases the "acceptance" or "dwell" time can be significantly reduced without placing a burden on the user, so that each key selection act can also be faster.

12.5 Scanning Switch

Switch access scanning is used to select a key for entry by an assistive technology user with significant mobility impairment. It only requires the user be able to control one movement in order to activate a single switch, with the additional constraint that the switch activation has to occur during a specified time window during the scanning process. The system highlights individual keys in succession, and when the desired key is highlighted, the user activates the switch. The switch can be a physical switch they locate where they have best control, or it can be a switch activated through video or other monitoring that detects a controlled movement, perhaps a muscle flex or head movement.

In many systems, a two dimensional row-column matrix of keys may be scanned in groups to reduce the number of scan steps. Multiple keys are highlighted in groups, the user presses the switch to select the group, then the keys in that group are successively highlighted and the user presses the switch again to select the desired key. However, since JustType already groups multiple letters on each key, the small number of keys required means that the savings in scan steps is minimal. The number of steps is reduced for the less-frequently used keys, while it is slightly increased for the more-frequently used keys. When the number of scan steps to reach each key is weighted by the relative frequency (in English) of the letters on each key, the reduction in scanning steps is only about 10%. Since this is at the cost of having to make two switch activations for each key selected, row-column scanning is only appropriate for an individual for whom activating a switch is virtually effortless.

The input methods above all provide a means of directly selecting each of the eight keys. Use of a scanning switch introduces a new goal for the user, reducing the number of scan steps. To be efficient, the keys should be ordered in such a way that the fewest scan steps are required for text entry[16]. For

[16] "Modeling Text Input for Single-Switch Scanning", Scott MacKenzie, York University, Toronto, Ontario, Canada

example, the “Select Key” is used most often to access the desired word. Placing it at the end of a linear scan would require waiting through all the other keys before the “Select Key” was highlighted.

Based on letter occurrence frequency[17], the preferred order of keys[18] for linear scanning to minimize scan steps are:

- ABC Keyboard: Select ABCD EFGH MNO. PQRS TUV IJKL WXYZ
- BRZ Keyboard: Select THKP ISVY ADFX CLOJ MEG. QUNW BRZ?

Note that the ABC layout is virtually in alphabetical order. The lone exception is the IJKL key is second to last. If the purpose of using the ABC layout is to make the keyboard appear simple and intuitive, we recommend placing the IJKL key in its ordinal position and accepting a slight inefficiency. We do, however, recommend the BRZ Keyboard for scanning for an increase in scanning efficiency.

13 Additional Notes

13.1 Explicit Text and Other Languages

Most languages make use of diacritics, also known as accented letters. English does not. The method described within this document for explicitly identifying characters on the keyboard only works for a keyboard with all the desired characters displayed.

For languages with diacritics, explicit words will require triad interpretation of key input. The first two keys identify the base character and the third key identifies any diacritic.

It is possible for languages with only a few diacritics to replace some of the less frequently used punctuation characters with the diacritic characters. Based on the infrequent nature of adding new words to the system, we recommend at least maintaining the core punctuation marks and digits so that layer changes to access these characters is minimized.

13.2 Prediction/Disambiguation Engines

Text input using 7 alphanumeric keys creates a significant amount of ambiguity when one key tap is used for each letter. Even more ambiguity arises when less than one key tap per letter is entered. Fortunately, most languages are redundant in many ways, so a disambiguation engine may be created to take a user’s input and current context to present the likely candidates for desired output.

The primary purpose of a disambiguation engine for JustType is to identify the most likely words based on the key sequence entered. There are a significant number of other functions that the disambiguation engine should provide. Continuous Path used two commercial engines for generation of the statistics

[17] Peter Norvig, “English Letter Frequency Counts: Mayzner Revisited or ETAOIN SRHLDCU”. Note that different sites count letter use differently. Norvig includes word frequency when counting letter frequency which best maps to use within a keyboard.

[18] The order is based on weighting the frequency of letter use in English by 2 and adding the frequency as the first letter in a word. This accounts for letter usage as well as accommodating the language model influence of locating words quickly.

found in this paper. A home grown disambiguation engine could also be used. The qualities of a disambiguation engine should include:

- Character mapping to keys – identifying which characters occur on which keys.
- Word-level disambiguation – identify word candidates for the key sequence and current context.
- Custom words – ability to learn new words through Explicit Word entry so that they may be entered ambiguously.
- Spell correction – recognize common spelling errors and provide alternative words to the input key sequence.
- Word completion – words the user may be trying to enter, presented before the user has entered all the characters for the word.
- Word prediction – words the user may be trying to enter, presented before the user has entered any characters for the word.
- Automatic spacing – provide support to identify where spaces are desired as well as where they are not wanted.
- Automatic capitalization – provide support for proper nouns and initial sentence capitalization.
- Punctuation support – to simplify access to appropriate punctuation for a given context.
- Language model learning – based on user input, learn the vocabulary and word patterns (n-grams) of the user to improve disambiguation engine efficiency and performance.
- Additional languages

Besides these traits, there are other additional features that a disambiguation engine may provide that further enhance the user's ability to enter text efficiently.

JustType demonstration software exists using both Nuance Communications and SwiftKey Ltd. word prediction engines. The concepts discussed here should apply to any similar commercial or home grown engine.

13.2.1 Nuance Communications – XT9®

XT9® supports text input in more than 100 languages. The disambiguation engine is platform agnostic, running on the vast majority of mobile and desktop platforms. XT9 (and precursor T9) have shipped on over 7 billion devices.

13.2.2 TouchType Ltd. – SwiftKey® SDK

SwiftKey® supports text input in more than 100 languages. The disambiguation engine is available on *nix, Linux, embedded, Microsoft™

Windows®, Apple® OS-X® / iOS™, and Android™ integrations. SwiftKey has shipped on over 250 million devices.

13.3 XPADDER

Xpadder emulates a mouse and keyboard with the buttons, joystick and directional thumb pad of your game controller. It supports multiple profiles, rumble feedback, and chorded input.

When we started to look at integrating a joystick with JustType, we were expecting to integrate joystick drivers and a lot of work. We found that the Xpadder software would transform any game controller into mouse and keyboard actions. As the JustType software is driven by keyboard keys, we found this to be an ideal solution. By mapping joystick movements into keyboard actions, we were able to quickly drive JustType with our gaming controller. Note that JustType runs as a Windows console application. Other platforms may or may not support such easy integration.

14 A BRIEF HISTORY OF CONSTRAINED TEXT INPUT

In 1946, ENIAC, arguably the earliest electronic general-purpose computer, used the teletype to input data. It comprised 39 keys, primarily alpha-numeric, and little more. Over time, the computer keyboard morphed to include more characters from typewriters. Text entry is generally 1 key per character. It is difficult to enter characters outside those defined for the keyboard without learning Alt keyboard sequences. Today, the typical computer keyboard has 100+ keys.

Nokia's GSM handsets were the first to support user generated text messages in 1993. These handsets contained hard keys and assigned letters to the keys based on the original telephone exchange character assignments from rotary dials. Entering text required character level disambiguation to identify the desired letter on the key. The multi-tap method was typically used to disambiguate the characters; an ambiguous key was tapped multiple times to identify the character position on the key. Multi-tap allowed text entry at an average of 2.03[19] keys per character.

Also in the 1990's, research to support communication for people with severe mobility impairments was occurring. Kushler, King and Grover developed prototype hardware and software to allow a user to enter text into a computer using eight keys. Seven keys contained either 3-4 characters each and the eighth key allowed a user to enter a space. The user wore an infrared system that could detect which direction a user glanced: up, down, left, right, and the four corners between these positions. A glance in one of the directions entered the key. This system used word level disambiguation instead of character level disambiguation. When the user entered the eighth key, the software would take the sequence of ambiguous character key entries, identify words that contained characters matching the ambiguous key sequence, and show the most likely word containing one character from each of the keys. As there might be other words with the same sequence of keys, the user could use the space key to step through

[19] MacKenzie, Scott, et al, LetterWise: prefix-based disambiguation for mobile text input, Proc. of UIST 2001, ACM Press (2001)

alternate words. This research is the basis for the JustType concepts presented in this paper. This method allowed text entry at 1.01 keys per character.

While this research was promising, the assistive technology marketplace has yet to adopt this approach in any but an almost vanishingly small number of devices. The researchers also identified text messaging on a mobile phone to be a nearly identical problem. Kushler, King and Grover commercialized their research as T9 in 1997. The T9 software and its variants has been installed on over 7 billion devices.

The advent of touchscreen enabled PDA's (followed soon by smartphones) and led to a different form of constrained text input. The touchscreens removed the restriction on the number of keys and replaced it with a full QWERTY keyboard. It is very difficult to tap the desired key accurately due to its small size, so a similar problem for text entry continued. The solution was the same – word level disambiguation. In this case, the ambiguity arose not because each key contained multiple letters, but rather that when the user tapped the 'S' key, for example, did they really mean to get an 'S', or did they intend one of the neighbors 'Q', 'W', 'E', 'D', 'C', 'X', 'Z', or 'A'? Word level disambiguation is used to allow for inaccurate key entry.

In 2002, one method of text entry for people with mobility impairments was using head/eye tracking to identify desired keys on a keyboard. A user would focus on the desired key and after a threshold time, the system would accept the key. Randy Marsden approached Kushler asking if there wasn't a way to remove the dwell time for entering each key, but instead follow the eye gaze for a full word and then identify word candidates. Kushler was intrigued, mulled the problem over and asked Illing to join him in the new endeavor.

In 2002, the simplest device to use for this development was to mimic a head/eye tracking system was a PDA with a touchscreen. Smartphones were not yet available, but PDA's were abundant. The user would place the stylus down on or near the first letter, slide to each of the other letters of the word, and lift the stylus on the last letter. By 2005, the system could take carefully entered paths and identify the intended word, but the algorithm was not tolerant enough for sloppy entry. In 2008, the system was ready. It could handle sloppy paths, misspellings, and poorly planned paths. It was released to the world as Swype, the first continuous touch text entry method. In 2015, Tobii introduce a Swype-like entry method using eye gaze for input control.

In 2010, a number of input methods began using more complicated language models to improve not only input recognition, but to also anticipate words the user may want based on their own patterns of use.