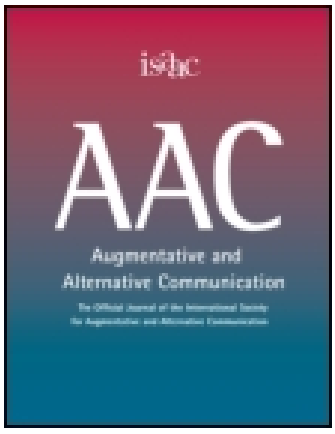


Augmentative and Alternative Communication
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/iaac20

# Huffman and Linear Scanning Methods with Statistical Language Models

Brian Roark[a], Melanie Fried-Oken[b] & Chris Gibbons[c]
[a] Google Research, Portland, Oregon USA
[b] Departments of Neurology, Biomedical Engineering, Pediatrics and Otolaryngology, Oregon Health & Science University, Portland, Oregon USA
[c] AbleNet, Inc., Roseville, Minnesota, USA
Published online: 08 May 2015.

*Augmentative and Alternative Communication*, 2015; 31(1): 37–50

ISSN 0743-4618 print/ISSN 1477-3848 online
DOI: 10.3109/07434618.2014.997890

informa
healthcare



**RESEARCH ARTICLE**

# Huffman and Linear Scanning Methods with Statistical Language Models

BRIAN ROARK[1], MELANIE FRIED-OKEN[2] & CHRIS GIBBONS[3]

[1]*Google Research, Portland, Oregon USA,* [2]*Departments of Neurology, Biomedical Engineering, Pediatrics and Otolaryngology, Oregon Health & Science University, Portland, Oregon USA and* [3]*AbleNet, Inc., Roseville, Minnesota, USA*

**Abstract**

Current scanning access methods for text generation in AAC devices are limited to relatively few options, most notably row/column variations within a matrix. We present Huffman scanning, a new method for applying statistical language models to binary-switch, static-grid typing AAC interfaces, and compare it to other scanning options under a variety of conditions. We present results for 16 adults without disabilities and one 36-year-old man with locked-in syndrome who presents with complex communication needs and uses AAC scanning devices for writing. Huffman scanning with a statistical language model yielded significant typing speedups for the 16 participants without disabilities versus any of the other methods tested, including two row/column scanning methods. A similar pattern of results was found with the individual with locked-in syndrome. Interestingly, faster typing speeds were obtained with Huffman scanning using a more leisurely scan rate than relatively fast individually calibrated scan rates. Overall, the results reported here demonstrate great promise for the usability of Huffman scanning as a faster alternative to row/column scanning.

**Keywords:** *Scanning; Natural language processing; Augmentative and alternative communication (AAC)*

## Introduction

Predictive language models, commonly designed within the field of natural language processing, can assist individuals with severe speech and language disabilities to communicate more effectively with their text entry systems, either by speeding access to an intended message or by reducing the effort required to select that message. For example, word completion systems use predictive language models to identify likely words that begin with already entered letters so that they can be presented to the user and selected with a single keystroke, instead of requiring each remaining letter in the word to be individually typed. Whether or not such word completion systems result in true speedups in text entry depends on many factors, including interface design, quality of the predictive model, and the text entry speed of the individual. Even so, word completion is a ubiquitous feature of speech-generating devices (SGD) and general mobile technologies, and is favorably used by many people with and without disabilities.

Words are not the only linguistic units that can be predicted within an SGD. Utterance-based systems (Alm, Arnott, & Newell, 1992; Todman, Alm, Higginbotham, & File, 2008) attempt to predict whole utterances that are contextually appropriate, which can be particularly useful for beginnings and ends of conversations, or during small talk. For pre-literate SGD users, symbols representing concepts (Gatti, Matteucci, & Sbattella, 2004) or sounds (Trinh, Waller, Vertanen, Kristensson, & Hanson, 2012) can be predicted to allow for communication. Finally, individuals using a single switch for keyboard emulation (e.g., through eye blink or other switches when direct selection is difficult or impossible) can benefit from predictions regarding which stimuli to present or highlight for selection. Sometimes this is achieved via linear scanning, by presenting one symbol at a time, perhaps in order of decreasing likelihood, and sometimes this is achieved by highlighting sets of symbols in a spelling grid according to predictive models. Recently, a new method, called *Huffman scanning*, was introduced that highlights portions of the grid in an optimal way given a predictive model and a Huffman code (Roark, de Villiers, Gibbons, & Fried-Oken, 2010; Roark, Beckley, Gibbons, & Fried-Oken, 2013). Huffman scanning has been shown to require far fewer keystrokes than widely used row/column scanning, and to result in faster text entry and strong user preferences relative

Correspondence: Melanie Fried-Oken, Oregon Health & Science University, CDRC, PO Box 574, Portland, OR 97239, USA. Tel: +1 503 494 7587. E-mail: friedm@ohsu.edu

*(Received 14 November 2013; revised 2 December 2014; accepted 7 December 2014)*

to row/column scanning. The above cited papers, however, present results from simulation and participants without disabilities, as well as results with relatively fast calibrated scan rates, leaving open the question of whether the approach would be found to be useable by individuals with severe motor impairments who make up the target users of such technology. In this paper, we present some trials to examine the importance of scan rate on overall typing speed for a copy task; and we present a case study of the use of this scanning technique alongside linear scanning and row/column scanning for a user with functional locked-in syndrome. We find statistically significant text entry speedups over row/column scanning using Huffman scanning with a contextual language model. These results provide a preliminary indication of the utility of these methods for individuals using scanning for keyboard emulation.

### *Alternative Text Entry*

Literate adults who cannot use standard keyboards for text entry because of physical impairments have a number of alternative text entry methods that permit typing, known as keyboard emulation. A single binary switch is a commonly used interface for alternative text entry, and may be realized with a button press at any consistent and reliable anatomical site, eye-blink or even event related potentials (ERP) such as the P300 detected in EEG signals (Lesher, Moulton, & Higginbotham, 1998). Typing speed is a challenge in such interfaces, yet critically important for usability. A common alternative text entry approach that uses a binary switch is row/column scanning on a matrix of characters, symbols or images (often referred to as a spelling grid). With the fixed spelling grid that appears in Figure 1, the user selects a target symbol by simply indicating yes when the desired row is highlighted, and then indicating yes when the desired cell is highlighted in the columns.

Of the ways in which AAC typing interfaces differ, perhaps most relevant to the current paper is whether the symbol positions are fixed or can move dynamically, because dynamic layouts can facilitate integration of richer language models. For example, if character probabilities are re-calculated after each typed character, then the characters in the grid could be re-arranged so that the most likely are placed in the upper left-hand corner for row/column scanning. Early research into optimizing spelling grids resulted in the Tufts Interactive Communicator (TIC) (Crochetiere, Foulds, & Sterne, 1974; Foulds, Baletsa, & Crochetiere, 1975) and the Anticipatory TIC (ANTIC) (Baletsa, 1977). In contrast to the roughly alphabetic grid shown in Figure 1, the TIC organized the letters in frequency order, so that frequently accessed symbols occurred in the upper left-hand corner of the grid, where less scanning is required to access them. Most currently manufactured AAC devices that present row/column scanning options rely on such an optimized letter configuration for fixed spelling grids. In 1987, Heckathorne and his research team described a device called PACA (Portable Anticipatory Communication Aid) that attempted to reorganize the grid based on probabilities given the previously typed letter, but it was never brought to market (Heckathorne, Voda, & Leibowitz, 1987). Clearly the cognitive load of processing a different grid arrangement after every character would slow down typing more than the speedup due to the improved binary coding (Baletsa, Foulds, & Crochetiere, 1976; Lesher et al., 1998). The GazeTalk system (Hansen, Johansen, Hansen, Itoh, & Mashino, 2003) is an eye gaze system with a dynamically changing $7.62 \times 10.16$ cm grid. In the typical configuration of that system, parts of the grid are fixed, but some cells contain likely word completions or the most likely single character continuations, both based on language model predictions. Hansen et al. (2003) report that users produced more words per minute with a static keyboard than with the predictive grid interface, illustrating the impact of the cognitive overhead that goes along the visual scanning required by dynamic grid reorganization.

Interfaces that require extensive visual scanning or motor control, or which rely upon complex gestures to speed typing, can make the interface difficult if not impossible for many people who use AAC. Venkatagiri (1999) compared different keyboards and letter presentations and found time and keystroke requirements varied as a function of the layouts and access methods. In this paper we will make use of a static grid, or a single letter linear scanning interface, yet scan in a way that allows for the use of contextual language model probabilities when constructing the binary access code for each symbol.

| | | | | | |
|---|---|---|---|---|---|
| _ | a | b | c | d | e |
| ← | f | g | h | i | j |
| k | l | m | n | o | p |
| q | r | s | t | u | v |
| w | x | y | z | . | , |
| " | - | ' | $ | : | ; |

Figure 1. Spelling grid used for standard row/column scanning.

### *Binary Codes for Typing Interfaces*

For any given scanning method, the use of a binary switch to select from among a set of options (letter, symbols, or images) amounts to the assignment of binary codes to each symbol. There are many ways to assign a binary code to symbols, such as extended ASCII codes, which assign 8 bit codes to symbols, such as the letter "m," which has the ASCII code 01101101. Scanning methods also amount to assigning such binary codes to symbols, different from the ASCII codes above in that they typically are of varying length, with some symbol codes being very short and others longer. The

binary switch for scanning is used to indicate the zeros and ones of the code, which results in the symbol being typed. For example, the standard row/column scanning algorithm works by scanning each row until a selection is made, then scanning each column until a selection is made, and returning the symbol at the selected row and column. For such an access method, the binary code for a symbol could be written as a 0 for every row from the top that does not contain the target symbol, followed by a 1 (for the row that contains it); then a zero for every column from the left that does not contain the target symbol, followed by a 1 (for the column that contains it). Not activating the switch indicates a zero bit; activating the switch indicates a one bit, and in such a way the binary code is input. Using this scanning method with the spelling grid in Figure 1, the binary code for the letter "j" is 01000001; the letter "m" is 001001. Note that this binary code for "m" is 6 bits long, shorter than its 8-bit ASCII code.

Ordering the symbols in the grid so that the most frequently accessed symbols are in the upper left-hand corner of the grid (as in TIC) results in shorter binary codes for those frequent symbols, hence less scanning required for typical symbol access. The overall frequency of letters, however, does not take into account what has just been typed, but rather assigns its codes identically in all contexts. Whether a particular character is likely or not depends to a great extent on the previous character and in fact the whole of the message up to that point. In this paper we examine alternative fixed-grid scanning methods that do take into account such context in the statistical language models used to establish binary codes for keyboard emulation.

### *Language Modeling for Typing Interfaces*

Statistical language models are common components of AAC systems. Most commonly, language models are used for word completion or word prediction, as mentioned earlier for the GazeTalk system. There has been extensive research on methods for integrating word completion or prediction models into AAC systems in such a way that they achieve keystroke reductions (Darragh, Witten, & James, 1990; Li & Hirst, 2005; Trost, Matiasek, & Baroni, 2005; Trnka, Yarrington, McCoy, & Pennington, 2006; Trnka, Yarrington, McCaw, McCoy, & Pennington, 2007; Wandmacher & Antoine, 2007). These keystroke reductions are achieved when a single keystroke suffices to select the rest of a word, rather than requiring keystrokes to select each of the remaining characters of the word. Monitoring a separate region of the interface that contains suggested completions involves some additional cognitive load (much as with dynamic grid reorganization), hence in some instances the actual realized typing speedup can be far less than the keystroke savings might lead one to expect (Anson et al., 2004), and in many cases a net slowdown in text entry can occur (Koester & Levine, 1994a; 1994b). Even so, there is some evidence that, under certain conditions, word completion and prediction speed typing (Trnka et al., 2007), and AAC keyboard emulation software commonly include such components.

Word completion is one way to use language modeling to speed text entry, though word completion is a method to augment an existing text entry system, rather than a stand-alone system. A fully functional text entry system must be open vocabulary in the same way as standard text entry modalities, such as the QWERTY keyboard: any word that the user wants to type should be able to be produced by the keyboard emulation. Word completion and prediction systems make those predictions over a vocabulary of possible words, and if the target words fall outside of that vocabulary (i.e., the word is out-of-vocabulary) then it will not show up as a word completion/prediction option, leaving the user to type the entire target. This is particularly important for typing items that fall outside of typical lexicons, such as proper names, acronyms, abbreviations, or any departures from standard orthography, such as informal text genre found in social media. The language model in the text entry system must be able to repair spelling and word errors by including some editing symbols such as a delete key. In other words, it is important to design language models to speed up a core open-vocabulary typing function by predicting single ASCII and control characters, rather than just multiple character strings. The predictive model must assign a probability over the set of possible next symbols, given what has already been typed. This task is actually very similar to the influential Shannon game (Shannon, 1950), where a given text was guessed one character at a time by people as the means for establishing the statistical characteristics of the language.

Beyond word completion, language models have been used to make letter selection easier. For example, Dasher (Ward, Blackwell, & MacKay, 2002) uses language models to make the interface region allocated to a character larger, thus facilitating hitting the target letter with a gesture. These gestures involve moving eye gaze or a mouse cursor over the physical region allocated to the symbol. As symbols are typed, newly predicted symbols gain probability and their regions grow dynamically across the interface in a way that makes it appear that the point of focus is moving from left-to-right through the set of possible messages. In order to type efficiently with the interface, Dasher requires sufficient motor control and visual attention to continuously move the cursor. For expert users, it can be very effective, but it can be very challenging for individuals with motor and/or cognitive impairments to use.

Most current AAC devices include both letter and word prediction options. Within the NLP research literature, the Sibyl/Sibylle interface (Schadle, 2004; Wandmacher, Antoine, Poirier, & Departe, 2008), similar to our brain–computer interface for spelling, known as RSVP Keyboard™ [1] (Oken et al., 2014; Orhan et al., 2012), involves a letter prediction component with a linear scan of the letters in probability order (based on

a 5-gram character language model) instead of row/column scanning. User feedback indicated that row/column scanning was more tiring than a linear scan in their system (Wandmacher et al., 2008).

In prior work (Roark et al., 2010; Roark et al., 2013), we demonstrated that Huffman scanning requires far fewer keystrokes than widely used row/column scanning, and results in faster text entry and strong user preferences relative to row/column scanning. However, our previous results were derived from either user simulations or study participants without disabilities, and were achieved with relatively fast calibrated scan rates. It is not yet known whether Huffman scanning is viable or preferred by individuals with severe motor impairments who are the target users of such technology. Obtaining results from people who rely on the assistive technology is critical if we are to understand what systems are viable, important and preferred. If we do not obtain patient-centered outcomes or conduct trials for practical use, then development efforts may become obsolete quickly, and our goals may not meet the needs of the very individuals we are addressing. As each assistive technology research paradigm must include trials with users, in this paper we present results from users with and without disabilities.

We will describe three experiments that attempt to determine how to leverage contextually sensitive language models in fixed-grid scanning beyond word completion and prediction; and to quantify the degree to which language models can make the diverse scanning interfaces competitive in terms of typing speed. Before doing so, however, we will first explicitly review our terminology, hopefully to avoid any confusion when describing methods and results.

### *Terminology*

This paper is presenting work at the intersection of several disciplines: AAC, computational linguistics, and human-computer interaction. As a result, what may seem to be the most natural term for a concept will vary across readers. Further complicating this is the occasional lack of field-internal consensus on the appropriate term. To aid interdisciplinary understanding, and avoid misinterpretation of our methods or results, we present terminology and how it relates to prior usage.

*Symbols, Words and Strings.* We use the terms symbol and character interchangeably to describe the smallest items being typed. Typically these are letters, but can include editing or formatting symbols such as backspace, delete or whitespace. A word is a whitespace delimited string of symbols. An n-gram is a string of symbols of length n (e.g., "abcd" is a 4-gram of letters).

*Bits, Switches, Scan Steps.* In direct selection systems, the symbol is the smallest unit that can be selected, and typing performance is often measured in terms of keystrokes (i.e., the number of select gestures required to enter the message). In scanning systems, multiple user actions are generally required to type a single symbol, and these actions can be referred to in various ways. Lesher et al. (1998) refer to base actions as *switches*, and these include switch activations (also called keypress or switch hit) or timeout (absence of switch hit). Recently, Mackenzie (2012) referred to switches in the Lesher et al. (1998) sense as *scan steps*, and advocated scan steps per character as a measure of scanning efficiency. Taking the perspective of binary codes as we have done in this paper, each of these switches or scan steps corresponds to a bit, hence bits-per-character is an equivalent measure to switches-per-character or scan-steps per-character. Given the focus on binary coding in this approach, we will discuss bits-per-character when presenting results.

*Optimal Bits per Character, Errors and Long Codes.* The ideal users would activate the switch always and only when required to type their target character. At each position in a test sequence, we can calculate the number of bits required to type the character if there are no errors. We report this as optimal bits per character, and this is the sort of measure reported in scanning simulations, such as those by Lesher et al. (1998) and MacKenzie (2012). Real users, however, make errors, and these errors can either result in the wrong symbol being typed, or in scanning past the target symbol but correctly typing it when the system reaches it again. We term this error as a *long code*, since the binary code that is used to select the symbol is longer than the optimal code for that symbol. A long code can result from either a timing error (failure to activate switch) or selection error (erroneous switch activation). For example, in row/column scanning, the target row may be missed or the wrong row selected, but this does not necessarily result in the wrong symbol being typed. We report both error rate (percentage of typed symbols that were incorrect) and long code rate (percentage of correctly typed symbols that had a longer than optimal code).

*Scan rate, Dwell time, Auto scan, Step scan.* We designed our scanning interfaces so that one of the user actions is to refrain from activating the switch and allow a timeout to trigger. Dwell time refers to the duration of the time interval during which the system waits for switch activation. A short dwell time produces a faster rate of scanning, since the user has less to activate the switch before the system records a timeout and moves on. In auto scanning, a timeout indicates that no selection has been made; in step scanning, a timeout indicates selection.

## Methods

### *Participants*

Participants were 16 literate adults without disabilities and one man with severe speech and physical impairments secondary to brainstem stroke. The participants

without disabilities were students and faculty at the Oregon Health & Science University with reportedly intact cognitive, motor, sensory and language skills. They were not familiar with our keyboard emulation interface prior to participation in the study. GB, the participant with locked-in syndrome, is a 36-year-old male, who had a brainstem stroke at the age of 20 and has been functionally locked-in ever since. He retains some slight head movement and has adequate visual scanning. He uses the Words+ Freedom 2000™ [2] speech generating device with EZ Keys software™ in row/column scanning mode and a Specs™ [3] switch placed on the AbleNet Universal Arm Mounting System™ [4] by his right temple for some of his email communication. For the majority of his expressive language, however, Greg depends on co-construction with a partner-dependent alphabet auditory scan and a familiar communication partner.

GB became quadriplegic after he completed US military service. At the time, he was enrolled in community college and worked as a landscaper. He currently lives with his mother and sister, and relies on the assistance of three paid caregivers during daytime hours for all activities. GB has been an active member of the RSVP Keyboard™ research team since 2006. He attends monthly team meetings at Oregon Health & Science University, where he provides input and suggestions for all research and development issues that arise. He has published his thoughts about BCI for AAC in *Speak Up*, the publication of the United States Society for Augmentative and Alternative Communication (Bieker, Noethe, & Fried-Oken, 2011). GB attended the 2013 International Brain-Computer Interface (BCI) Meeting in Monterey, CA where he presented a workshop on AAC for BCI (Huggins et al., 2015) and participated in the BCI Users' Forum in front of 350 conference participants (Peters et al., 2015). Of note, all presentations were prepared using partner-assisted auditory scanning. He finds technology to be too slow, fatiguing, and missing the engagement and personal attention that defines verbal communication and interaction for him.

GB was first introduced to AAC for writing with the Apple IIe computer[6], Discover: KENX interface[7], and Co:Writer software[8] by Don Johnston, Inc. Over time, he switched to a Words+ Freedom 2000™ [2] system, with which he uses stored phrases and generates new messages with one switch row/column scanning. He is constantly looking for alternatives. He completed a number of trials with eye gaze systems. He found that the eye gaze devices created a physical barrier between him and his communication partners and he prefers to look at the people he is interacting with. GB relies on assistive technology for short written text only. Due to his dependence on others for system set up and maintenance, and the time demands to start using AAC, he reports that he does not like to use AAC for spontaneous conversation. He states that there is a place for AAC, but he prefers to use partner-assisted auditory scans for speaking with others or preparing long text. Even though he has a portable laptop and wheelchair mount, he finds the AAC system intrusive and difficult to take with him. Placement of the switch and switch mount is a constant barrier to device use, and it must be readjusted multiple times during each use. He enjoys the participation of his caregivers and verbal engagement during the message formulation process, and finds that it is much quicker to ask his familiar caregiver to predict word and phrase completions then to rely on the AAC system. GB uses his row/column scanning device for short emails only. Message production rates are slow; he is about 60% accurate with initial scanning selections and must delete or retype letters often. By nature, he reports that he is not a big talker and relies on short messages rather than long stories. He enjoys humor and sports, often connecting with others in short emails about teams or jokes. He is constantly looking for a better switch option, and is currently investigating a laser switch. GB is a BCI evaluator, and tests all system upgrades and treatment protocols that are developed for our new assistive technology. Since the technology is not yet reliable for home use without significant technical support, he does not rely on BCI for composing personal text. GB is very aware of the speed challenges associated with any AAC device. He states that he often fatigues when using AAC technology, and the attention demands for accuracy are high. At this time he is willing to trade the independence of AAC technology for a faster, more personal system that involves one-on-one interaction and constant engagement with a communication partner.

### *Materials*

*Corpora and Character-based Language Modeling.* Character n-gram models were used for this experiment. We follow Roark et al. (2013) in using Witten-Bell smoothed n-gram language models, and refer the reader to that paper for more extensive presentation of these methods. Briefly, each character's probability is conditioned on the previous seven characters, and the probabilities are regularized using widely known methods so that even unobserved sequences receive some probability. Models are trained on newswire text from the New York Times portion of the English Gigaword corpus (LDC2007T07) and individual words taken from the CMU Pronouncing Dictionary (www.speech.cs.cmu.edu/cgi-bin/cmudict), which were appended to the newswire corpus to give better word coverage. The resulting corpus was preprocessed for the current evaluation, as detailed in Roark et al. (2010) and Roark et al. (2013). The pre-processing was carried out so as to yield text that was actually typed – as opposed to, say, pasted signatures, bylines, tabular data, automatically generated text, etc. – hence of some utility for modeling the kind of language that would be produced in English typing applications. Further, the pre-processing reduced text duplication in the corpus.

*Binary Codes.* The language models described above give us the means to assign a probability to each symbol in the symbol set, given what has been typed up to

that point. The symbol set is then sorted in decreasing probability order and a binary coding tree is built over the symbol set, so that the binary code assigned to each symbol differs depending on the context, i.e., what has been typed up to that point. Huffman and linear scanning approaches make use of these binary coding trees, and we refer the reader to Roark et al. (2013) for technical details on this approach.

### *Stimuli*

All participants were presented with the text entry phrase set from MacKenzie and Soukoreff (2003) to evaluate typing performance. Of the 500 phrases in that set, 20 were randomly set aside for testing, and the other 480 were available as stimuli during training and calibration phases. Of the 20 evaluation strings, five were used in this study: I can see the rings on Saturn, Watch out for low flying objects, Neither a borrower nor a lender be, An offer you cannot refuse, and The facts get in the way.

*Scanning Paradigms.* The interface developed for this study presented phrase copy tasks to measure typing performance under six scanning paradigms. The scanning paradigms were: (1–2) row/column scanning, both auto scan (switch activation for character selection) and step scan (absence of switch activation for character selection); (3–4) Huffman scanning using either a unigram language model (no context) or an 8-gram language model to derive the Huffman codes; and (5–6) scanning with a linear scanning code, either on the six-by-six grid like the other scanning approaches, or using RSVP (Rapid Serial Visual Presentation) paradigm, where only one symbol at a time is flashed on the screen. For each paradigm, participants saw a target phrase with instructions to type the phrase exactly as displayed. Any incorrect symbols were deleted by selecting the backspace symbol followed by selection of the correct symbol.

The same optimized letter matrix is used in all grid scanning conditions, based on letter frequency.

*Row/column Scanning.* The typing interface for row/Column scanning is shown in Figure 2(a). The target string to be copied is presented to the participant at the top of the window, and immediately below this is the buffer showing what has already been typed. Whitespace between words (represented here by the underscore character, shown in the grid's upper left-hand corner) also must be typed correctly by the participant. When an incorrect character is selected, the error is shown in red with a backspace symbol immediately following – as shown in Figure 2(b) – to assist users in recognizing that an error has been made and must be corrected.

For row/column scanning, the cursor returns to the top row after passing the bottom row without a selection. Once a row has been selected, the cursor wraps around the letters three times. If no cell is selected, then scanning continues from the following row. This provides a recovery mechanism if an error in row selection is made.

*Huffman Scanning.* When configured for Huffman scanning (Roark et al., 2010; Roark et al., 2013), the grid remains the same but the highlighting differs, as shown in Figure 3. In this mode, a Huffman code divides the symbols into two sets (0 and 1). Highlighting of a cell indicates that the symbol is in Set 1. Because the division into two sets occurs in order to reduce the number of selections required to reach the desired symbol, the highlighted cells cannot generally be contiguous. It is the relaxation of requiring highlighting to be contiguous that allows the use of the language model to determine the code. Non-contiguous highlighting in scanning is rare, but has been discussed (Demasco, 1994). As far as we know, however, this is the first time non-contiguous highlighting has been used for dynamic, contextually-sensitive coding. Huffman coding has been applied previously to a scanning paradigm (Baljko & Tam, 2006), though with a unigram model (i.e., no context) in a way that ensured contiguity of highlighted regions.

To understand how Huffman scanning works, we provide the following example, which calculates the probability of letter selection from all possible letters

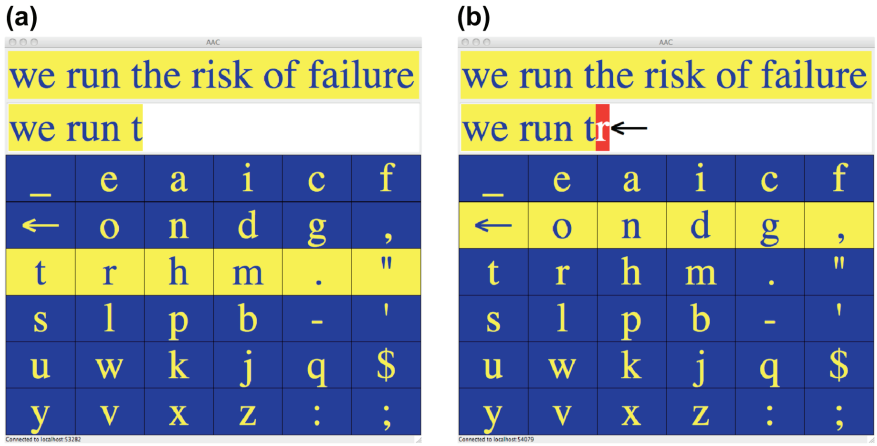


Figure 2. The typing interface presented to participants with and without disabilities (a) for row/column scanning and (b) after an error was entered during the copy task.


Figure 3. The typing interface presented to participants with and without disabilities during the Huffman scanning paradigm.

(Huffman, 1952). Suppose the partial phrase "His name is Tom Rob" has been typed, and we are using Huffman scanning to type the next letter. The current intended name might continue with an *i* (e.g., Robinson), an *e* (Roberts), a *b* (Robbins), and so on. Or perhaps the previously typed *b* was an error, hence the current target is the delete symbol. For the sake of illustration, suppose that the probabilities for the most likely symbols are as follows: prob(*i*) = 0.3; prob(*e*) = 0.3; prob(*b*) = 0.2; and prob(delete) = 0.1. The rest of the letters share the remaining 0.1 of probability, summing properly to 1.

The symbols are divided into two sets, so that each set has as close to half of the probability mass as is possible. (Recall that a set's probability is the sum of the probabilities of its members.) Note that, given the probabilities above, the letters *i* and *e* cannot end up in the same set, since together they have total probability 0.6, which would lead to the other set having probability 0.4, less than the optimal 0.5. If, instead, we put *i* and *b* in one set, and everything else in the other set, both sets will have the desired probability of 0.5. The smaller set (*i* and *b*) is then highlighted and the system observes whether there is switch activation or not within the specified dwell time. Note that the letter *e*, which is tied for the highest probability, is not highlighted. Highlighting of a cell in the grid does not indicate that a cell is particularly likely or not; rather it simply indicates whether, after dividing the symbols up, it is in the highlighted set.

In this novel Huffman scanning approach, the highlighted set of symbols is selected by switch activation. If a selected set is just a single character, that character is typed. If there is more than one symbol in the selected set, then the binary code is recalculated for the next presentation of highlighting. This is done in such a way that, until a symbol is typed, no symbol is ruled out. If the switch is activated, so that the highlighted set (*i* and *b*) is selected, there is some small probability (0.1) that the selection was unintended (i.e., the target symbol is in the other set). Based on this, we recalculate all the symbol probabilities, and the new probability of letter *i* works out to be just over 0.5, so it will be placed in its own set for the next step of the scanning. Scanning continues like this until a single character is selected. After every scan step, the Huffman or linear scanning codes are recalculated, taking into account the probability of an error, in such a way that non-selected letters retain some probability of being selected in the system. In other words, even if the wrong character is selected, thus reducing the probability of the target symbol, subsequent correct selections will gradually increase the probability of the target symbol, eventually leading to it being typed. The recovery procedure with Huffman scanning does not resemble the recovery loop used for row column scanning. We refer the reader to Roark et al. (2010) and Roark et al. (2013) for further technical discussions on how these codes are recalculated taking into account an error rate parameter.

*Linear Scanning.* By highlighting a single cell at a time, in descending probability order, linear scanning occurs on the grids shown in Figures 2 and 3. Alternatively, linear scanning occurs with an RSVP interface, shown in Figure 4, which presents single characters serially.

### *Procedure*

There are three stages to the experimental paradigm: training a participant to understand the different scanning tasks, calibrating the scanning rate for each task, and completing the experimental scanning tasks. Participants completed phrase copy tasks with the aforementioned six different scanning paradigms. Participants without disabilities engaged in all six scanning paradigms; GB participated in four of these conditions. Participants were seated at a table in front of a laptop that presented our keyboard emulation software. The participants without disabilities selected targeted symbols by using their dominant hand to hit a Jelly Bean Switch™ [5] on the tabletop; GB used his chin to depress a Specs switch attached to the Universal Switch Mounting System on his wheelchair.

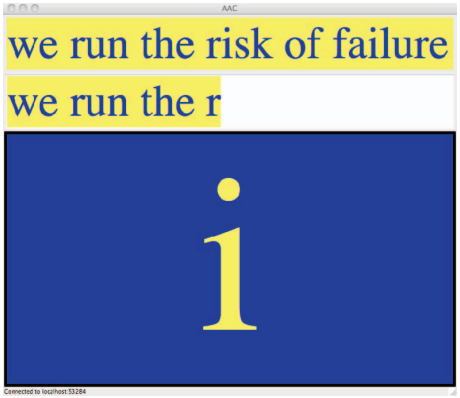


Figure 4. The typing interface presented to participants with and without disabilities during the RSVP paradigm.

*Training.* Participants were presented with a short demonstration of the interface by an author, which involved typing a short string ("a dog") using row/column auto scan, row/column step scan, and Huffman scanning. The demonstration included target selection, typing the wrong symbol, deleting a symbol, typing a space, and scanning past the target symbol. Following this simple demonstration, the participants were presented graphically with overall instructions for the task. A self-paced instruction module explained details about the task to be completed. For each of the typing conditions, condition-specific instructions were presented. These instructions explained how keyboard emulation functions for each condition and reiterated important overall task instructions. The scan rate for each condition, specifying how long to wait for a button press before advancing, was set for each individual during the training/calibration session.

*Individual Scan Rate Calibrations.* Each participant's scan rate must be calibrated before the experimental task is presented. We will briefly review the calibration procedure here and refer the reader to Roark et al. (2013) for full details. Because the Huffman and linear scanning approaches require the individual to react to what is presented on the screen, the scan rate is constrained by reaction time. In contrast, row/column scanning allows for anticipation and planning, since the current configuration of the highlighting determines the subsequent configuration of the highlighting. One might expect, then, that faster scan rates would be more suited to row/scanning than compared to the other approaches being investigated. Auto scan and step scan, however, are substantially different in terms of the control required to advance scanning versus selection, hence, one might expect these scan rates to also differ. Calibration was thus conducted for three scenarios: row/column step scan, row/column auto scan, and Huffman scanning with a unigram language model. This latter rate was then used for all non-row/column scanning conditions, all of which are constrained by reaction time.

Calibration is run in two phases. The first phase begins with a relatively slow scan rate and speeds up by 200 ms after each correctly typed target string until it reaches a scan rate at which the participant is unable to successfully type the target. The second phase for each condition (held after a period during which other methods are used) begins with a scan rate 500 ms slower than the fastest scan rate at which the participant failed in the prior phase, then speeds up by 100 ms after each successful target. After reaching a scan rate that is too fast for the participant to successfully type the target, the scan rate decreases by 50 ms until the subject can successfully type the target, and this final scan rate is taken as the final calibrated rate for that individual.

*Experimental Testing.* Participants typed the same five phrases from the test set in six distinct conditions, the ordering of which was randomized for each participant. There were two row/column scanning conditions, auto and step scan; two Huffman scanning conditions, with codes derived from either the unigram language model or the 8-gram language model; and two linear scanning conditions, either scanning on the 6-by-6 spelling grid or RSVP single letter presentation. Both linear scanning conditions used codes derived from the 8-gram language model. Identical instructions to those given during the calibration phase were provided in each condition, and a practice phase preceded the typing of phrases from the test set. Once the participant met the error rate criterion performance (10% error rate or lower) for that condition, he or she advanced from practice phase and was given the test phrases to type.

## Results

### *Experiment 1: Calibrated Scanning by Participants Without Disabilities*

The 16 native English speakers without disabilities calibrated the scan rate for each scanning paradigm at the start of the experimental session, and the results are presented in Table I. The raw results presented for this experiment are also reported in Roark et al. (2013), though we provide a statistical analysis of the results and subsequent discussion here for the first time.

This is a copy task where participants are instructed to correct all errors and copy the presented test string exactly. To avoid scenarios where the participant was unable to correct all errors and successfully complete typing a particular stimulus (due, perhaps, to the participant not recognizing that an error has been made or being unable to recover from a rapid sequence of many errors), a trial was interrupted if the total number of errors on the stimulus reaches a threshold. After interruption, the participant was prompted to type the same target phrase again, starting from the beginning. The total time and number of keystrokes both before and after interruption were counted when calculating performance. For the current experiment, the threshold was set to 20 errors, and just two participants reached this threshold, for one phrase each in a row/column scanning condition (see Table II).

Table II (first presented in Roark et al., 2013) reports typing speed and related measures (means and standard deviations) for the participants. Typing speed is presented as characters per minute, and the number of keypress or non-keypress (timeout) events used to type the characters is presented as bits per character. (See the earlier Terminology section for a detailed discussion

Table I. Calibrated Scan Rates.

| Scanning condition | Scan rate (ms) *M* (*SD*) |
|---|---|
| Row/column step scan | 419 (95) |
| Row/column auto scan | 328 (77) |
| Huffman and linear scan | 500 (89) |

Table II. Typing Results for 16 Users on Five Test Strings (Total 31 Words, 145 Characters), Under Six Conditions, with Calibrated Scan Rates.

| Scanning condition | Speed (cpm) *M* (*SD*) | Bits per character *M* (*SD*) | Bits per character opt | Error rate *M* (*SD*) | Long code rate *M* (*SD*) |
|---|---|---|---|---|---|
| Row/column step scan | 21.2 (3.8) | 8.4 (2.5) | 4.5 | 6.2 (4.4) | 30.4 (17.9) |
| Row/column auto scan | 19.6 (3.2) | 8.0 (1.7) | 4.5 | 4.6 (3.0) | 28.9 (13.3) |
| Huffman unigram | 12.8 (2.3) | 7.9 (1.7) | 4.4 | 4.4 (2.3) | 35.1 (13.0) |
| Huffman 8-gram | 24.5 (3.9) | 4.0 (0.9) | 2.6 | 4.2 (2.2) | 15.4 (12.5) |
| Linear grid 8-gram | 23.2 (2.4) | 4.1 (0.6) | 3.4 | 2.4 (1.7) | 4.3 (3.4) |
| RSVP 8-gram | 20.9 (4.4) | 5.5 (2.2) | 3.4 | 7.1 (4.8) | 4.2 (3.7) |



of these terms.) Bits per character does not correlate perfectly with typing speed, since a timeout by definition consumes the fully allotted trial time, and a key press is typically much shorter. We also present rates of two kinds of errors. Note that, to successfully complete the trial, the stimulus must be typed correctly. One kind of error involves typing the wrong symbol, then deleting and retyping the correct symbol. We present this sort of mistake as error rate – the number of incorrect symbols typed divided by the total symbols typed. Another kind of mistake in typing a symbol is to scan past the symbol without typing it, but eventually return to that symbol and type it correctly. This mistake results in extra keystrokes, which means that the binary code used to select the character was longer than what it would have been optimally. We present this sort of mistake as the long code rate – the percentage of correctly typed symbols that required a longer than optimal code. To see how a perfect user would have performed, we present the optimal bits per character that could be achieved with each method.

We used a paired-sample t-test to compare the typing speed of the subjects across the six conditions, with Bonferroni correction based on 15 comparisons. The Huffman unigram condition was significantly slower, $p < 0.001$, than each of the other conditions, approximately twice as slow as Huffman 8-gram. Both the Huffman 8-gram, $p < 0.01$, and Linear grid 8-gram, $p < 0.05$, conditions were significantly faster than row/column auto scan. The Huffman 8-gram condition was also significantly faster than the RSVP 8-gram condition, $p < 0.05$. None of the other differences were statistically significant.

Table III (originally from Roark et al., 2013) presents mean scores for four questions: I was fatigued by the end of the trial. I was stressed by the end of the trial. I liked this trial. I was frustrated by this trial. The responses showed a consistent preference for Huffman and linear grid conditions with an 8-gram language model over the other conditions (see results in Table III).

### *Experiment 2: Fixed Rate Scanning by Participants without Disabilities*

Having a calibration session, and setting scan rates at the fastest scan rate at which participants can perform the task within stipulated error rate bounds, ensures that methods that allow for anticipation, such as row/column scanning, get some advantage in terms of characters per minute that might be produced. However, the fastest scan rate at which the task can be accomplished is not the same as the scan rate that results in the most characters per minute. In fact, the relatively high error rates and long code rates that were observed in some of these conditions may lead one to believe that these scan rates were not optimal in this respect and that a more leisurely scan rate which resulted in fewer errors and missed targets may in fact yield faster typing. From these considerations, we ran a follow-up experiment using a fixed scan rate of 600 ms across all of the conditions. Note that there are many ways in which a scan rate can be chosen for a particular individual, through the use of more complex calibration methods or with the help of a speech-language pathologist. Rather than individualizing, we chose a rate that was comfortable across all participants and conditions, but fast enough to avoid frustration. The same participants were used for this experiment as for Experiment 1, and the testing protocol was identical to that experiment. Participants had at least one day between Experiment 1 and Experiment 2. Results are presented in Table IV.

Table III. Mean Likert Scores to Experiment 1 Survey Questions (5 = Strongly Agree; 1 = Strongly Disagree).

| Question | Row/Column step | Row/Column auto | Huffman 1-grm | Huffman 8-grm | Linear grid | Linear RSVP |
|---|---|---|---|---|---|---|
| I was fatigued by the end of the trial | 3.1 | 1.9 | 3.3 | 1.9 | 1.9 | 2.8 |
| I was stressed by the end of the trial | 2.4 | 2.0 | 2.5 | 1.7 | 1.6 | 2.6 |
| I liked this trial | 2.1 | 3.5 | 2.6 | 3.9 | 3.8 | 2.9 |
| I was frustrated by this trial | 3.4 | 1.8 | 3.1 | 1.7 | 1.7 | 2.9 |

Table IV. Typing Results for 16 Users on Five Test Strings (Total 31 Words, 145 Characters), Under Six Conditions, with Fixed 600-ms Scan Rate.

| Scanning condition | Speed (cpm) *M* (*SD*) | Bits per character *M* (*SD*) | Bits per character opt. | Error rate *M* (*SD*) | Long code rate *M* (*SD*) |
|---|---|---|---|---|---|
| Row/column step scan | 22.3 (2.3) | 6.2 (0.9) | 4.5 | 2.2 (1.6) | 17.5 (9.1) |
| Row/column auto scan | 18.1 (2.2) | 5.6 (0.7) | 4.5 | 3.3 (2.7) | 10.2 (5.8) |
| Huffman unigram | 14.8 (2.0) | 6.0 (0.8) | 4.4 | 2.6 (2.1) | 18.2 (7.0) |
| Huffman 8-gram | 27.3 (3.4) | 3.1 (0.4) | 2.6 | 2.1 (2.3) | 5.8 (4.6) |
| Linear grid 8-gram | 22.6 (1.1) | 3.7 (0.2) | 3.4 | 2.0 (1.6) | 0.8 (0.9) |
| RSVP 8-gram | 23.7 (2.3) | 3.9 (0.5) | 3.4 | 2.8 (2.4) | 1.0 (1.0) |



In only two of the conditions (auto row/column scanning and linear grid scanning) were the typing speeds in characters per minute slower than in the calibrated condition, and neither slowdown reached statistical significance. In most conditions, the reduction in error and long code rates offset the slowdown in scan rate, resulting in significantly faster typing for both Huffman scanning conditions and the RSVP 8-gram condition. Within this experiment, we again find that the Huffman unigram condition was significantly slower than each of the other conditions, $p < 0.001$, again using paired-sample t-test with Bonferroni correction based on 15 comparisons. The nearly factor of two slowdown in this condition relative to the other Huffman scanning condition was also preserved. We also find that the Huffman 8-gram condition is significantly faster than every other condition, $p < 0.001$. Finally, the row/column step scan, linear grid 8-gram and RSVP 8-gram conditions were all significantly faster than row/column auto scan, $p < 0.001$. None of the other differences were statistically significant.

### *Experiment 3: Scanning by Participant with Severe Speech and Physical Impairments*

After examining performance using these new scanning methods for participants without disabilities, we turned to performance of a man who uses AAC. GB participated in four of the original six conditions. Step scanning was omitted because it requires a greater number of key presses than auto scanning methods, and was too difficult for GB. Based on the prior results, we see that the Huffman unigram method is much more difficult for the users, hence this was also omitted. The remaining four conditions were preserved: auto scan row/column, Huffman scanning with an 8-gram model, linear scanning on a grid, and linear scanning with RSVP.

Rather than using an automatic scan rate calibration method, we provided the user with a training session, in which a participating speech-language pathologist established the correct position of the switch and manually calibrated the scan rate of various conditions to be comfortable for the user. A scan rate of 1 s was established for the row/column scanning condition and 1.5 s for the other conditions.

Table V presents GB's typing speed and other statistics for the four conditions, with the same test stimuli as the previously presented results. To no surprise, typing speed was slower for all conditions when using GB's 1 s versus 600 ms scan rates in the prior experiment. Error rates and long code rates were also increased, except for the linear grid 8-gram rate where GB's rate matched the rate of the other participants. For GB, Huffman 8-gram yielded a 60% speedup over row/column auto scanning, more than the 50% speedup observed in Experiment 2. Table VI presents GB's responses to the user survey, showing a preference for the row/column scanning and Huffman scanning versus the linear scanning conditions, presumably due to frustration with positions requiring a long linear scan through many options. In addition, the RSVP condition was the one judged to be fatiguing and stressful. GB, as disability consultant to our RSVP BCI project, also provides regular feedback on the focused attention that is needed for the RSVP Keyboard, commenting often that it would be difficult to use this paradigm for message generation (see Table VI).

## Discussion

The experimental examination of language-model effects within static scanning grids is a timely endeavor, as AAC is becoming more available through SGDs and mobile technologies to literate individuals with severe

Table V. Typing results for One AAC User on Five Test Strings (Total 31 Words, 145 Characters), Under Four Conditions, with Fixed Scan Rates (1 s for Row/Column Scanning; 1.5 s Others).

| Scanning condition | Speed (cpm) | Bits per character Subject | Bits per character Opt. | Error rate | Long code rate |
|---|---|---|---|---|---|
| Row/column auto scan | 6.0 | 8.5 | 4.5 | 4.4 | 32.2 |
| Huffman 8-gram | 9.7 | 3.8 | 2.6 | 3.8 | 14.6 |
| Linear grid 8-gram | 9.5 | 3.9 | 3.4 | 2.0 | 5.4 |
| RSVP 8-gram | 7.5 | 5.2 | 3.4 | 5.5 | 5.2 |

Table VI. Likert Scores for AAC user Survey Questions (5 = Strongly Agree; 1 = Strongly Disagree).

| Question | Row/Col Auto | Huffman 8-grm | Linear grid | Linear RSVP |
|---|---|---|---|---|
| I was fatigued by the end of the trial | 1 | 1 | 1 | 3 |
| I was stressed by the end of the trial | 1 | 1 | 1 | 5 |
| I liked this trial | 5 | 5 | 1 | 1 |
| I was frustrated by this trial | 1 | 1 | 5 | 5 |



speech and physical impairments. Adults with acquired impairments are coming to the task of single-switch spelling with expertise in letter prediction and word prediction gained from using smart phones and word processors. They are asking sophisticated questions about how to increase typing speed for spontaneous conversation, given their past experiences with mobile technology and computer access. The field of AAC has not kept pace with the rapid computational language and machine learning advances that are occurring for general technology. With renewed interest from both computational linguists and AAC experts in language enhancement for functional communication, we must begin to analyze binary switch scanning performance with language, cognitive, sensory/motor and environmental variables. The field of natural language processing can offer new avenues of research for the AAC field to make assistive technology for communication more effective and efficient (Darragh & Witten, 1992; Higginbotham, Moulton, Lesher, & Roark, 2012). The introduction of Huffman coding as a plausible scanning technique is one such endeavor.

Despite the small study size (17 study participants, including just one literate adult who used AAC), several key results are apparent. First, Huffman scanning requires a strong language model to be a viable alternative to row/column scanning. When just using a unigram model, Huffman scanning was significantly slower than either of the row/column scanning approaches. For the three methods that do not make use of the contextual language model (unigram Huffman scanning and both row/column scanning approaches), we observed a relatively high long code rate, leading to a near doubling of the length of codes required for each character in the absence of errors and a slow scan rate. The slow scan rate reduces the error rates, but does not yield improved speeds for any of these methods.

Second, the contextual model significantly improves Huffman scanning. The 8-gram language model reduces the optimal bits per character for Huffman scanning over the unigram model, and reduces the difference between the actual bits per character achieved by the participants and the optimal. This results in a near doubling of typing speed over Huffman scanning with the unigram model. With the faster calibrated scan rates, linear scanning on the grid with the 8-gram model yields nearly the same bits per character as Huffman scanning. This effect results from a decrease in the error rate and long code rate with Huffman scanning. We believe the reduced error rate is influenced by easier visual scanning, resulting from just one cell being highlighted at each step, drawing the eye to the target cell more quickly. The long code rate remains substantially lower for linear scanning than for Huffman scanning, even at the more leisurely fixed scan rate of 600 ms. This same result was observed for GB, who noted his preference for linear scanning.

Finally, scanning with the RSVP interface is slower than Huffman scanning with an 8-gram model or linear scanning on the grid, due to an increase in error rate. Even so, it yields speeds commensurate with row/column scanning. Finding a symbol on the grid and waiting for its cell to be highlighted seems to be a slightly easier task than recognizing the symbol that appears in the single RSVP screen and reacting, even for literate adults with no visual attention challenges. The fact that this task became easier with the slower fixed scanning rate (error rates decreased dramatically) demonstrates the extra time demands of recognition and reaction of this condition versus just watching for a cell in the grid to light up.

The results from different scanning paradigms speak to a pressing need to evaluate the visual processing demands and working memory requirements of all alphabetic scanning systems. While our work highlights the contribution of language models to scanning, there are many unexplored variables that affect performance. Wilkinson, Hennig, Soto, and Zangari (2009) emphasized the multiple and unique cognitive processing demands of AAC technologies, without discussing alphabetic scanning paradigms for literate users. Thistle and Wilkinson (2012) discuss the attention demands of dynamic screens and grid layouts with symbols for children. Visual attention patterns of adult AAC users with aphasia have been explored, again in dynamic screens rather than scanning matrices (Thiessen, Beukelman, Ullman, & Longenecker, 2014). The rapid changing demands of letter-based scanning paradigms are another visual presentation that deserves our attention, and can account for significant variation in learning and use by literate individuals with and without disabilities. Additional variables, including the specific method of error recovery in a particular scanning approach, the possibility of imperfect scan rate calibration, and the demands of the copy task itself could have an impact on typing speed. In previous work, we controlled the level of difficulty of phrases in the copy task and contributions of the language model to letter selection, and found that both variables affect alternative access spelling, in this case with a brain–computer interface (Orhan et al., 2012). It is difficult, with user studies, to control all influencing factors. When simulations are conducted, as in Lesher et al. (1998) or MacKenzie (2012), all variables are controlled to examine a full range of factors that affect optimal configurations of typing systems. The current results cannot be compared easily with the outcomes from simulations; or, for that matter, to results with an ambiguous keyboard such as

Hansen et al. (2003) or MacKenzie & Felzer (2010). The conditions under which we collected data (two user studies for individuals with and without disabilities) provided substantially different testing situations than MacKenzie (2012), which included simulations and word completion. Reviewing different stimulus conditions (simulations versus user studies), language model contributions (word completion versus no word completion), and keyboards (ambiguous versus unambiguous) will lead to important and interesting questions that this paper has left unresolved.

Results for the 16 participants without disabilities and for GB, the participant with locked-in syndrome for 20 years, were similar for four conditions. For both non-disabled and disabled participants, the Huffman scanning is the fastest of the options, but linear scanning on a grid is competitive due to a lower long code rate. Note that for the linear scanning on the grid, GB's bits per character almost mirror the mean of the participants without disabilities (Table IV), so that the differences in typing speed can be attributed to differences in scan rate. This provides evidence that people who use AAC and present with only physical impairments perform within normal limits for age-matched peers. Even though GB prefers partner-assisted auditory alphabet scanning, his technology use resembles that of his non-disabled peers. He demonstrates that spelling-based keyboard interfaces with letter and word prediction for individuals who are locked-in are viable clinical options, if the individuals retain cognitive and language skills for functional expression.

GB's participation here highlights the unique role that people with severe disabilities should play in all AAC research. While disability advocates remind us of "nothing about me without me" (Delbanco et al., 2001), this concept is not always practiced during technology research and development. Often we read about experiments with symbol sets or device layouts that have been presented to children without disabilities, or engineering efforts that simulate conditions without including potential users. In fact, the challenges that are posed to people with severe speech and physical impairments are the very barriers that we must address in research. These barriers may affect attitudes toward adoption or abandonment, the availability of customization options and preferences for device features, and the ultimate challenge of real time implementation. GB and other individuals with disabilities who are research consultants or disability advocates are the experts who we must learn from. Their opinions, equipment testing, feedback and experimental results validate research and development efforts. GB tells us that he participates in research so that he can help others who are less fortunate than he is. By examining his behavior, we are given an opportunity to scientifically answer the real questions that should be asked. Inclusion of GB and others with complex communication needs provides a level of AAC validity that leads to measurement and interpretation of the relevant constructs or ideas.

We have demonstrated the importance of contextual language modeling for the usability of Huffman scanning. An extension of this work would be the investigation of learning effects on message generation with scanning paradigms. The order of paradigm presentation or even the repetitive text stimuli might have affected results. We report results from novice users of scanning interfaces, with the exception of our participant who uses AAC. Some of the differences between the calibrated experiment and the fixed rate scanning experiment were likely due to novice users having high error rates with very fast scan rates, a skill that should improve with training. Further, the frequency ordered grid more heavily impacts speed for novices than expert users, since expert users are familiar with the order of letters on the grid. The effect of training would determine whether expertise speeds up Huffman and linear scanning as much as row/column scanning, a research question that should be explored. These could be addressed easily through simulations but would not provide the human–computer interface results that we explored.

Additional investigations are warranted to optimize performance for users based on their individual preferences and behaviors. Parameters can be adjusted, and are customizable in currently available AAC scanning systems. It is difficult to control for individual variability, and based on our sample size, we cannot conclude that Huffman scanning is faster than various kinds of row/column scanning. We asked whether scanning based on Huffman codes – which are guaranteed to yield the best optimal bits per character – would be found to be sufficiently useable by individuals as a viable alternative to available scanning methods. Based on the results presented here, it is clear that the Huffman scanning was found to be sufficiently useable to capture much of the gain promised by the optimal Huffman codes. In fact, GB, the participant who uses AAC, typed faster with Huffman scanning than with row/column scanning. Within AAC, there are currently too few options to speed up text entry for the literate person who relies on scanning access for message generation. Huffman scanning may provide a potentially useful alternative.

## Acknowledgements

We thank GB for taking the time to participate in this series of scanning experiments.

## Notes

1. The RSVP Keyboard™ is a P300 brain–computer interface spelling system that is being developed at Oregon Health & Science University under an NIH grant (5R01DC009834) to Dr Fried-Oken, principal investigator.
2. The Words+ Freedom 2000™ speech generating device and EZ Keys™ software were manufactured and sold by Words+, Inc. of Lancaster, CA, USA.

3. The Specs™ Switch is manufactured and sold by AbleNet, Inc. of Roseville, MN, USA.
4. The Universal Switch Mounting System™ is manufactured and sold by AbleNet, Inc. of Roseville, MN, USA.
5. The Jelly Bean Switch™ is manufactured and sold by AbleNet, Inc. of Roseville, MN, USA.
6. Apple IIe computer is the third model Apple II series of personal computers produced by Apple Computer of Cupertino, CA, USA.
7. Discover: KENX is a keyboard interface designed to access a computer using a choice of methods other than a standard keyboard that was manufactured by Don Johnston Inc. of Volo, IL, USA, and is a trademark of Madentec Limited, of Edmonton, Alberta, Canada.
8. Co:Writer is word prediction software manufactured by Don Johnston Inc. of Volo, IL, USA.

***Declaration of interest:*** The authors report no conflicts of interest. The authors are responsible for the content and writing of the paper.

We acknowledge support from NIH/NIDCD grant 1R01DC009834.



## References


Alm, N., Arnott, J. L., & Newell, A. F. (1992). Prediction and conversational momentum in an augmentative communication system. *Communications of the ACM*, *35*, 46–57.

Anson, D., Moist, P., Przywars, M., Wells, H., Saylor, H., & Maxime, H. (2004). The effects of word completion and word prediction on typing rates using on-screen keyboards. *Assistive Technology*, *18*, 146–154.

Baletsa, G. S. (1977). *Anticipatory communication*. (Doctoral dissertation, Tufts University).

Baletsa, G., Foulds, R., & Crochetiere, W. (1976). Design parameters of an intelligent communication device. *Proceedings of the 29th Annual Conference on Engineering in Medicine & Biology* (p. 371).

Baljko, M., & Tam, A. (2006). Indirect text entry using one or two keys. *Proceedings of the Eighth International ACM Conference on Assistive Technologies (ASSETS)* (pp. 18–25).

Bieker, G., Noethe, G., & Fried-Oken, M. (2011). Locked-in and reaching new heights. *Speak Up*, 3–6.

Crochetiere, W., Foulds, R., & Sterne, R. (1974). Computer aided motor communication. *Proceedings of the 1974 Conference on Engineering Devices in Rehabilitation* (pp. 1–8).

Darragh, J. J., & Witten, I. H. (1992). *The reactive keyboard*. Cambridge: Cambridge University Press.

Darragh, J. J., Witten, I. H., & James, M. L. (1990). The reactive keyboard: A predictive typing aid. *Computer*, *23*, 41–49.

Delbanco, T., Berwick, D. M., Boufford, J. I., Ollenschläger, G., Plamping, D., & Rockefeller, R. G. (2001). Healthcare in a land called PeoplePower: Nothing about me without me. *Health Expectations*, *4*, 144–150.

Demasco, P. (1994). Human factors considerations in the design of language interfaces in AAC. *Assistive Technology*, *6*, 10–25.

Foulds, R., Baletsa, G., & Crochetiere, W. (1975). The effectiveness of language redundancy in non-verbal communication. *Proceedings of the Conference on Devices and Systems for the Disabled* (pp. 82–86). Philadelphia, PA.

Gatti, N., Matteucci, M., & Sbattella, L. (2004). An adaptive and predictive environment to support augmentative and alternative communication. In K. Miesenberger, A. Karshmer, P. Penaz, & W. Zagler (Eds.), *Computers Helping People with Special Needs* (pp. 983–990). Berlin, Heidelberg: Springer.

Hansen, J. P., Johansen, A. S., Hansen, D. W., Itoh, K., & Mashino, S. (2003). Language technology in a predictive, restricted on-screen keyboard with ambiguous layout for severely disabled people. *Proceedings of EACL Workshop on Language Modeling for Text Entry Methods*. Seattle, WA.

Heckathorne, C. W., Voda, J. A., & Leibowitz, L. J. (1987). Design rationale and evaluation of the Portable Anticipatory Communication Aid – PACA. *Augmentative and Alternative Communication*, *3*, 170–180.

Higginbotham, J., Moulton, B., Lesher, G., & Roark, B. (2012). The application of natural language processing to augmentative and alternative communication. *Assistive Technology, 24*, 14–24.

Huffman, D.A. (1952). A method for the construction of minimum redundancy codes. *Proceedings of the IRE*, *40*, 1098–1101.

Huggins, J. E., Guger, C., Allison, B., Anderson, C., Batista, A., Brouwer, . . . & Thompson, D. (2014). Workshops of the fifth international brain-computer interface meeting: defining the future. *Brain-Computer Interfaces, 1*, 27–49.

Koester, H., & Levine, S.P. (1994a). Modeling the speed of text entry with a word prediction interface. *IEEE Transactions on Rehabilitation Engineering*, *2*, 177–187.

Koester, H., & Levine, S. P. (1994b). Learning and performance of able-bodied individuals using scanning systems with and without word prediction. *Assistive Technology, 6*, 42–54.

Lesher, G.W., Moulton, B. J., & Higginbotham, D. J. (1998). Techniques for augmenting scanning communication. *Augmentative and Alternative Communication*, *14*, 81–101.

Li, J., & Hirst, G. (2005). Semantic knowledge in word completion. *Proceedings of the 7th International Conference on Computers & Accessibility*. Baltimore, MD.

MacKenzie, I.S. (2012). Modeling text input for single-switch scanning. In K. Miesenberger, A. Karshmer, P. Penaz, & W. Zagler (Eds.), *Computers helping people with special needs* (pp. 423–430). Berlin: Springer.

MacKenzie, I. S., & Felzer, T. (2010). SAK: Scanning ambiguous keyboard for efficient one-key text entry, *ACM Transactions on Computer-Human Interaction*, 17(3), 1–39.

MacKenzie, I. S., & Soukoreff, R. W. (2003). Phrase sets for evaluating text entry techniques. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)* (pp. 754–755). ACM: New York.

Oken, B., Orhan, U., Roark, B., Erdogmus, D., Fowler, A., Mooney, A. & Fried-Oken, M. (2014). Brain-computer interface with language model-electroencephalography fusion in locked-in syndrome. *Neurorehabilitation and Neural Repair, 28*, 387–394.

Orhan, U., Hild II, K., Erdogmus, D., Roark, B., Oken, B., & Fried-Oken, M. (2012). RSVP keyboard™: An EEG based typing interface. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Kyoto, Japan.

Peters, B., Bieker, G., Heckman, S., Huggins, J., Wolf, C., Zeitlin, D., & Fried-Oken, M. (2015). Brain-computer interface users speak up: The virtual users' forum at the 2013 international BCI meeting. *Archives of Physical Medicine and Rehabilitation*. In press.

Roark, B., Beckley, R., Gibbons, C., & Fried-Oken, M. (2013). Huffman scanning: Using language models within fixed-grid keyboard emulation. *Computer Speech & Language, 27*, 1212–1234. http://dx.doi.org/10.1016/j.csl.2012.10.006.

Roark, B., de Villiers, J., Gibbons, C., & Fried-Oken, M. (2010). Scanning methods and language modeling for binary switch typing. *Proceedings of the NAACL-HLT Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)* (pp. 28–36). Los Angeles, CA.

Schadle, I. (2004). Sibyl: AAC system using NLP techniques. In *Proceedings of the 9th International Conference on Computers Helping People with Special needs (ICCHP)* (pp. 1109–1015). Paris, France.

Shannon, C.E. (1950). Prediction and entropy of printed English. *Bell System Technical Journal*, *30*, 50–64.

Thiessen, A., Beukelman, D., Ullman, C., & Longenecker, M. (2014). Measurement of the visual attention patterns of people with aphasia: A preliminary investigation of two types of human engagement in photographic images. *Augmentative and Alternative Communication*, *30*, 120–129. doi:10.3109/07434618.2014.905798

Thistle, J. J., & Wilkinson, K. M. (2012). What are the attention demands of aided AAC? *Perspectives on Augmentative and Alternative Communication, 21*, 17–22.

Todman, J., Alm, N., Higginbotham, D. J., & File, P. (2008). Whole utterance approaches in AAC. *Augmentative and Alternative Communication*, *24*, 235–254.

Trinh, H., Waller, A., Vertanen, K., Kristensson, P. O., & Hanson, V. L. (2012). Applying prediction techniques to phoneme-based AAC systems. *Proceedings of the Third Workshop on Speech and Language Processing for Assistive Technologies (SLPAT*) (pp. 19–27). Montreal, Canada.

Trnka, K., Yarrington, D., McCoy, K. F., & Pennington, C. (2006). Topic modeling in fringe word prediction for AAC. *Proceedings of the International Conference on Intelligent User Interfaces* (pp. 276–278). Sydney, Australia.

Trnka, K., Yarrington, D., McCaw, J., McCoy, K. F., & Pennington, C. (2007). The effects of word prediction on communication rate for AAC. *Proceedings of HLT-NAACL; Companion Volume, Short Papers* (pp. 173–176). Rochester, NY.

Trost, H., Matiasek, J., & Baroni, M. (2005). The language component of the FASTY text prediction system. *Applied Artificial Intelligence*, *19*, 743–781.

Venkatagiri, H. S. (1999). Efficient keyboard layouts for sequential access in augmentative and alternative communication. *Augmentative and Alternative Communication*, *15*, 126–134.

Wandmacher, T., & Antoine, J. Y. (2007). Methods to integrate a language model with semantic information for a word prediction component. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 506–513). Prague, Czech Republic.

Wandmacher, T., Antoine, J. Y., Poirier, F., & Departe, J. P. (2008). Sibylle, an assistive communication system adapting to the context and its user. *ACM Transactions on Accessible Computing (TACCESS)*, *1*, 1–30.

Ward, D. J., Blackwell, A. F., & MacKay, D. J. C. (2002). DASHER – A data entry interface using continuous gestures & language models. *Human-Computer Interaction*, *17*, 199–228.

Wilkinson, K. M., Hennig, S. C.. (2009). Considerations of cognitive, attentional, and motivational demands in the construction and use of aided AAC systems. In Soto, G. & Zangari, C. (Eds.). *Practically speaking: Language, literacy & academic development for students with AAC needs* (pp. 313–334). Baltimore, MD: Paul H. Brookes Publishing.