Development and Theoretical Evaluation of Optimized Phonemic Interfaces

Gabriel J. Cler
Boston University
635 Commonwealth Ave
Boston, MA 02215
617-358-1395
gcler@bu.edu

Cara E. Stepp
Boston University
635 Commonwealth Ave
Boston, MA 02215
617-353-7487
cstepp@bu.edu

ABSTRACT
In this paper, optimized communication interfaces in which users select phonemes (sounds) instead of letters or whole words are presented and evaluated. Optimization is based on phoneme transition likelihoods (i.e., the probability of transitioning from one phoneme to another in a particular communication corpus), similar to letter-to-letter transition likelihoods used to optimize orthographic interfaces. However, it is unknown to what extent phoneme transition likelihoods vary by corpus, nor how optimizing based on different corpora affects the final interface efficiency. Here we used computational evaluations to compare phoneme transition likelihoods between various phonemic corpora and optimize phonemic interfaces with each corpus. Each interface's efficiency was evaluated against all the corpora. Phoneme-to-phoneme transitions were highly correlated across corpora (r = 0.7-0.86). Optimization based on phoneme-to-phoneme transition likelihoods improved efficiency by around 20-30% compared to random phonemic layouts, regardless of the corpus used to optimize the interface. Optimizations using different corpora were similar, varying only by 3-5%. We conclude that, if possible, future phonemic interfaces should be optimized via a corpus from the intended user's communication. If this is not possible, however, optimization still improved efficiency using all testing corpora, suggesting that optimizing via any relevant corpus is indicated over other layouts.

CCS Concepts
• Human-centered computing → Human computer interaction (HCI) → Interaction devices → Keyboards • Human-centered computing → Accessibility → Accessibility design and evaluation methods • Human-centered computing → Interaction design → Interaction design process and methods → User interface design

Keywords
Phonemic interface; augmentative and alternative communication; metropolis algorithm

ASSETS '17, October 29-November 1, 2017, Baltimore, MD, USA

ACM 978-1-4503-4926-0/17/10…$15.00
https://doi.org/10.1145/3132525.3132537

1. INTRODUCTION
Some individuals use augmentative and alternative communication (AAC) methods to communicate, including those who have concomitant motor impairments. For these individuals, AAC use requires both an interface from which to select targets and a method by which to select those targets. Alternative access methods for people with motor impairments range from devices that track head movements [e.g., 30, 31], eye movements [e.g., 11, 12, 16], tongue movements [e.g., 13], sip and puff actions [e.g., 12], or brain signals [e.g., 4, 32]. However, all of these access methods remain noisy and effortful for the user. Despite the technological advances achieved in these areas, target selection can remain slow and effortful, particularly in people with minimal movement capabilities [2, 12]. To improve communication speed and flexibility, further efforts are necessary to improve both AAC access methods and AAC communication interfaces.

1.1 Phonemic Interfaces
AAC interfaces typically provide targets consisting of letters, whole words, or symbols. Each presents benefits and drawbacks, typically compromising between speed and flexibility. Some AAC interfaces have been developed that use phonemes (which represent a particular sound in a spoken language) as targets instead. Phoneme selection allows individuals to create any set of sounds in their language, rather than relying on text-to-speech methods. Of particular interest to speakers with motor impairments, common AAC messages have 14-20% fewer phonemes than letters, depending on the set of vocabulary or messages evaluated (e.g., [5]). This may reduce the time needed to produce messages while retaining full flexibility. The drawback to using phonemes as interface targets is that one must learn to translate intended thoughts into a phoneme set rather than into letters. Typically we spend many years as children learning to translate thoughts into letters (i.e., writing). Sequencing phonemes (or syllables), although more similar to typical oral communication, is also likely to require training in order to produce intended messages. However, the speed and flexibility advantages to phonemic input suggest that it may be appropriate for some users, and thus effort should be expended to develop the most efficient phonemic interface possible.

Phonemic interfaces have previously been proposed for use by a variety of user populations, including children and adults with learning disabilities and/or motor impairments [3, 20, 24, 27]. Some systems contain only a small set of phonemes [3], or display a reduced set of phonemes on the screen at one time [24], such that users must make several motor actions to select one phoneme (e.g., selecting one target to indicate that you wish to select a fricative, and then selecting a target on a second screen that appears to select /f/). Other systems use a reduced set of phonemes and then must disambiguate the intended selections based on prior selections [27], somewhat similar to the T9 texting system [e.g., 14]. Finally, some phonemic interfaces display all

possible phonemes, but disable phonemes that are unlikely to be selected next [20]. Unfortunately, these final two methods for increasing efficiency restrict users to selecting only those words contained in the system's dictionary, without allowing for non-words, proper names, or novel utterances.

It has previously been shown that participants without motor impairments could use a noisy AAC access method [8] to produce speech using a phonemic interface in which all phonemes are available to select at all times [5]. This interface had phonemic targets arranged *a priori* based on articulatory features. However, other ways to improve efficiency of phonemic interfaces in which all phonemes are available to select at all times have not yet been explored.

1.2 Efficiency of Orthographic Interfaces

A variety of methods for optimizing orthographic arrangements have been employed, for both physical keyboards (e.g., the Dvorak typewriter keyboard [10]) and onscreen keyboards (e.g., OPTI [17], FITALY [22] or ATOMIK [33]). Many of these were optimized using some combination of trial-and-error and manual incorporation of letter frequency-of-use and letter-to-letter transition likelihoods. Due to the ubiquity of QWERTY keyboards, most users (with and without motor impairments) do not choose a more efficient orthographic keyboard layout. AAC users do sometimes use an alphabetic arrangement or a frequency-based arrangement, particularly if using a very slow scanning method of communication access. However, if users choose to utilize a phoneme-based interface due to its flexibility and the reduced number of selections required, they will not have a previously-learned arrangement of targets (such as QWERTY in an orthographic interface) to produce interference, so learning an optimal target arrangement is likely not appreciably different from learning any other target arrangement.

1.3 Optimizing Interface Efficiency

Direct selection access methods (e.g., finger pointing, head-tracking, eye-tracking) are generally considered to be less cognitively taxing than scanning methods [21], and thus are typically chosen if AAC users have the physical capability to directly select. Optimizing the layout of an interface used in switch scanning typically involves reordering the targets by frequency of use, such that those targets that are likeliest appear in the beginning of the scanning process [15]. There are also methods of optimizing the arrangements of targets on an interface to be used with direct selection; while this efficiency optimization has been implemented for orthographic keyboards (e.g. [33]), it has not been applied to phonemic interfaces.

One way to calculate and then maximize the efficiency of an interface is based on Fitts' law, a fundamental model of directed movements that suggests that the time it takes to select a target is based on the target's distance and size – a nearer target is faster to select because it requires less movement, and a larger target is faster to select because it requires less precise movements. To optimize the efficiency of an interface, one can arrange the targets such that the distance between targets that are often selected sequentially is minimized.

Importantly, methods of calculating efficiency for direct selection rely on the frequency of letter-to-letter transitions, or *digraph statistics*, in which "digraph" means letter pairs. For example, if the word "the" appears in a corpus many times, the digraphs T→H and H→E (and space→T and E→space) will have high probabilities. For orthographic text entry, many researchers use digraph statistics from Mayzner and Tresselt [18]. However, some have noted that these traditional digraph likelihoods do not represent AAC usage [29]. Various text and conversational corpora have been used to calculate digraph likelihoods and to train language models for prediction. Results show that while testing and training models on the same corpus leads to the best keystroke savings, some savings can still be found even when training and testing on different text corpora [29]. It is not clear whether this holds true with *diphone* likelihoods (phoneme-to-phoneme transition likelihoods) and resulting phonemic interfaces.

In this paper we present the results of optimizing and then testing the efficiency of phonemic interfaces using a variety of corpora to determine if phonemic interfaces optimized for AAC must be tailored to each user or if one generic keyboard (e.g., QWERTY, ATOMIK orthographic keyboards) is sufficiently efficient for all users.

1.4 Research Questions and Motivation

It is currently unknown to what extent phoneme transition likelihoods vary by corpus, nor how optimizing based on different corpora affects the final interface efficiency. In this study, we evaluate phoneme transition likelihoods between various phonemic corpora and optimize phonemic interfaces with each corpus. Then we evaluate each interface's efficiency by testing against all the corpora. If interface efficiency is highly impacted by the testing corpus, communication interfaces should be optimized per user. If efficiency is stable across testing corpora, we would expect AAC users to show similar performance using arrangements of phonemes based on any number of corpora. For reference, we additionally evaluated the efficiencies of two potential phonemic interfaces that were not explicitly optimized for efficiency, but potentially offer more immediate ease of use: a phonemic interface in which the phonemes are arranged alphabetically by their label ("Alphabetic"; developed for this study) and a phonemic interface in which phonemes are arranged by articulatory features such as manner and place ("Articulatory"; developed previously and described in [5]).

Here we present a theoretical evaluation of seven different phonemic interfaces against five different AAC/speech corpora. Thoroughly testing this many interface and testing set combinations in AAC users is infeasible, particularly as performance typically improves over time (thus necessitating many testing sessions per interface per user) [6], and because access to these individuals is limited. This paper thus focuses on thoroughly detailing the quantitative processes involved in evaluating various corpora, optimizing interfaces, and performing theoretical evaluations as a means to reduce the set of interfaces upon which to perform the necessary empirical evaluations by AAC users.

2. METHODS

2.1 Phoneme Set

The full set of phonemes used in American English is subject to some debate. For simplicity, the set of phonemes used in this study was the reduced set of phonemes used in the Carnegie Mellon Pronouncing Dictionary [23]; this machine-readable dictionary was used to convert text corpora to phonemes, and its set of phonemes is similar to those used in the Buckeye Corpus ([19]; see *2.2 Corpora*). See Table 1 for the set of phonemes used for most of the interfaces. Note that the articulatory interface (see *3.2.3 Articulatory Interface*) collapsed the phonemes /AA/ and /AO/ into one target.

2.2 Corpora

The usage statistics used to optimize an interface impact its arrangement and thus its efficiency for the end-user. However, there is no one ideal corpus of AAC messages. Therefore, we have compared five corpora (see Table 2 for more details): an unabridged vocabulary list of one young adult AAC user [26], a list of conversational phrases suggested by AAC specialists [9], a bank of simulated AAC messages [28], and the Buckeye corpus of conversational speech [19]. Text corpora were converted to phoneme transition likelihoods by converting text to phonemes via the CMU Pronouncing Dictionary [23] with hand-corrections for words not contained in the dictionary (e.g., "aneurysm"). The Buckeye Corpus has two types of phonemic transcriptions: one that matches the dictionary entry for a given orthographic transcription ('phonemic' by their terminology, or 'dictionary' here for clarity) and one with actual phonemes produced by speaker ('phonetic' by their terminology, but 'direct' here). For example, one speaker said the phrase "tomorrow's my dinner"; the dictionary transcription of "tomorrow's" is /T AH M AA R OW Z/, whereas the direct transcription is /T M AA R AH Z/. Two separate sets of transition likelihoods were calculated using the dictionary and direct transcriptions. For both, any transcriptions that included phonemes that were not in our set (e.g., 'AHN' for a nasalized 'AH'; syllabic 'EL') were converted to in-set phonemes (e.g., 'AH'; 'AH – L').

Table 1. Reduced set of phonemes

Arpabet label	IPA label	Example word
AA*	ɑ	father
AE	æ	at
AH	ʌ, ə	hut
AO*	ɔ	ought
AW	aʊ	cow
AY	aɪ	hide
B	b	be
CH	tʃ	cheese
D	d	dee
DH	ð	that
EH	ɛ	red
ER	ɝ	hurt
EY	eɪ	ate
F	f	fee
G	g	green
HH	h	he
IH	ɪ	it
IY	i	eat
JH	dʒ	just
K	k	key
L	l	lay
M	m	man
N	n	no
NG	ŋ	sing
OW	oʊ	oat
OY	ɔɪ	toy
P	p	pay
R	r	read
S	s	sea
SH	ʃ	she
T	t	tier
TH	θ	think
UH	ʊ	hood
UW	u	two
V	v	veer
W	w	we
Y	j	yield
Z	z	zoo
ZH	ʒ	measure

*These two phonemes are combined into one phoneme in the Articulatory interface

2.3 Calculating Interface Efficiency

Fitts' law (Equation 1) suggests that the movement time (MT) necessary to travel between targets *i* and *j* is related to the distance between the centers of target *i* and target *j* (D_{ij}), and the width of the second target (W_j). Exact movement times are determined experimentally based on the pointing device employed; these are then used to derive the constants *a* and *b*.

$$MT = a + b\left[\log_2\left(\frac{D_{ij}}{W_j}+1\right)\right] \quad \text{Eq. (1)}$$

Table 2. Corpora

Corpus	Description	Number of words	Conversion process
AAC user [26] ("Actual AAC")	Unabridged vocabulary list with use statistics from one young adult AAC user	49,718	Converted each **word** to phonemes via CMUDict (thus missing word-to-word transitions)
AAC conversational phrases [9] ("Suggested AAC")	Context-specific message list compiled by AAC specialists	3,941	Converted each **message** to phonemes via CMUDict
Simulated AAC messages [28] ("Simulated AAC")	Mechanical Turk simulated AAC messages	25,182	Converted each **message** to phonemes via CMUDict
Buckeye Corpus - dictionary transcription [19] ("Conversation-dictionary")	40 typical speakers conversing orally with interviewer	284,832	Converted to reduced phoneme set. Calculated transitions by **message** from dictionary phonemic entry of orthographic transcription
Buckeye Corpus - direct transcription [19] ("Conversation - direct")	40 typical speakers conversing orally with interviewer	284,832	Converted to reduced phoneme set. Calculated transitions by **message** from direct phonetic transcription

The efficiency of a particular target layout is quantified via Equation 2 [33], which is derived from Fitts' law, and suggests

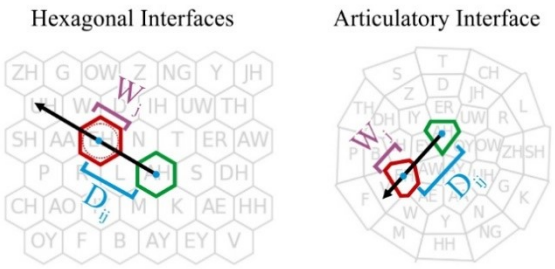

Figure 1. Width (W_j) and distance (D_{ij}) calculations for the interfaces developed and evaluated in this study. Starting phoneme *i* outlined in green, with target phoneme *j* outlined in red. Width is calculated as the distance between the two intersection points of the ideal path from the center of the starting phoneme through the center of the target phoneme (blue dots).

that the average movement time ($\overline{MT}$) of an interface is characterized by the sum of the probability of transitioning between each pair of phonemes (i and j) multiplied by the time it would take to get from phoneme i to phoneme j. Average movement time is converted to words per minute (WPM) as shown in Equation 3 for human readability and comparison with other quantitative orthographic keyboards.

$$\overline{MT} = a + \sum_{i=1}^{39}\sum_{j=1}^{39} Pr_{ij} * b\left[\log_2\left(\frac{D_{ij}}{W_j}+1\right)\right] \quad \text{Eq. (2)}$$

$$\text{Efficiency (words/min)} = \frac{1 \text{ word}}{5 \text{ phonemes}} \times \frac{1 \text{ phoneme}}{\overline{MT} \text{ (sec)}} \times \frac{60 \text{ sec}}{\text{minute}} \quad \text{Eq. (3)}$$

2.3.1 Fitts' constants

The constants *a* and *b* in Equations 1 and 2 are Fitts' constants, which are experimentally derived aspects of the pointing device itself. Any change in these arising from choosing a different access method will affect the estimate of efficiency (i.e., some access methods are slower than others), but will not affect the *comparison* of two efficiencies using the optimizing algorithm. Therefore, we have used constants that apply to a stylus type pointing device, in which a is assumed to be 0, b = 1 / 4.9, and a = .127s when *i* = *j* in order to compare to other literature [17, 33]. Thus, efficiencies calculated will be considerably higher than those generated by AAC users with noisy access methods, across all interfaces.

2.3.2 Distance and widths

Distances between each pair of targets are calculated as the Euclidean distance between the centers of the targets *i* and *j*. Widths are calculated as the width of the second target *j* along the ideal path from the center of the starting phoneme through the center of the target phoneme. For all of the interfaces developed in this study, the target width is consistent, whereas for the articulatory interface (previously developed and discussed in [5]), the target width varies (see Figure 1).

2.3.3 Transition likelihoods

Transition likelihoods were calculated from each corpus in Table 2 separately. Any text content was translated to phonemes via the CMU Pronouncing Dictionary [23] and hand-corrected. For the AAC user's vocabulary list, each word's transitions were counted and multiplied by the number of times it was used. For the remaining corpora utilizing messages, all phonemes from the message were concatenated, and each transition was counted.

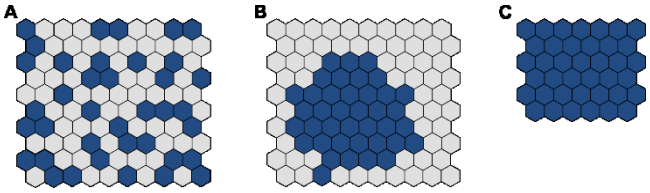

Figure 2. Different configurations of 39 targets; (A) shows a 10x10 interface before the Metropolis algorithm has run, in which 39 hexes have been randomly assigned a phoneme (blue) and the rest are unassigned (grey) (B) shows an interface with high efficiency after running the Metropolis algorithm, which has tightly clustered the targets (max efficiency noted: 39.655 WPM via Suggested AAC corpus). (C) has the 39 targets arranged in a consistent layout both before and after the optimization (max efficiency noted: 39.608 WPM via Suggested AAC corpus).

Then each set of counts was divided by the total number of transitions in the corpus, such that the sum of the probabilities was 1.

Note that the phoneme set in the articulatory interface was slightly different than all others (see section 3.2.3 *Articulatory Interface*). Therefore, to test this interface, separate transition likelihoods for each corpus were recalculated to collapse all X→/AA/, /AA/→X, X→/AO/, /AO/→X, /AA/→/AO/, and /AO/→/AA/ likelihoods as appropriate.

2.3.4 Words per minute

Equation 3 includes a standard assumption used for orthographic keyboards, in which the average word is said to require five selections per word (four characters plus the space key). Theoretically, the average number of selections per word for a phonemic interface should be nearer 3, as no space key is necessary or provided, and as there are 14-20% fewer phonemes than letters per word depending on the corpus tested. Regardless, this is left at the orthographic standard of 5 selections/word; values in WPM are presented here only for human readability and could be recalculated at need to represent a "truer" estimate of the phonemic WPM (see section *4.2 Other Efficiency Calculations*).

2.4 Metropolis Optimization Algorithm

Once an interface's efficiency can be quantified, any number of optimization algorithms can be used. One such optimization algorithm is the Metropolis algorithm (see [1] for a review), which is a Markov chain Monte Carlo algorithm with a variety of applications; of particular interest here, it has previously been used to optimize orthographic keyboards.

In this case, the Metropolis algorithm is used as such: one interface of 39 phonemes is randomly generated, and its efficiency is calculated. The interface layout is then optimized via a random walk: two phonemes are randomly swapped and the keyboard efficiency is recalculated. If the new arrangement is more efficient than the current layout, then it is kept and the random walk continues. If the new arrangement is less efficient than the current layout, it may still be kept, according to Equation 4, in which the probability of keeping a less efficient arrangement is quantified by the difference in efficiency (ΔE) and the system temperature (T) multiplied by a scalar (k). The system temperature cycles over time in a process called annealing; this enables the system to break out of local $\overline{MT}$ minima [33].

$$Pr(keep_{new}) = 1, \quad \text{if } \overline{MT}_{new} < \overline{MT}_{old} \quad \text{Eq. (4)}$$

$$= e^{-\Delta E/(kT)}, \text{ if } \overline{MT}_{new} \geq \overline{MT}_{old}$$

2.4.1 Interface Shape

The most efficient interface is one in which the targets are tightly clustered, reducing the distances required to move the cursor. Thus target arrangements with hexagonal targets are more efficient than those with rectangular targets arranged in a grid.

The Metropolis algorithm can optimize any interface shape. Initial simulations were done with a large target space (e.g., 10 rows x 10 columns of target locations for the 39 phonemes; Figure 2A) to seek the most efficient layout. After running the algorithm, the most efficient arrangements had targets clustered together (Figure 2B). The tightly clustered targets were in a roughly circular arrangement, which maximizes efficiency (i.e., with phonemes assigned in a particular order, Figure 2B had an efficiency of 39.655 WPM, the maximum output of the algorithm through many iterations), but is less efficient in terms of screen space for end-users, who will need to have other programs on the screen. Therefore, the pre-set target arrangement in Figure 2C was chosen to maximize end-user usability and aesthetics while only slightly reducing efficiency (i.e., highest efficiency with this shape was 39.608 WPM).

2.4.2 Determining Constants

The width of each target was set at 10 to represent the circle circumscribed by the hexagon of the target. The distance between each button was calculated via the Euclidean distance between the centers of each target. As Equations 1 and 2 show, distance is divided by width, making the units arbitrary as long as they are consistent.

The probability of selecting an arrangement that is less efficient (Equation 4) is dependent on the difference in efficiency and two scaling variables, k (scalar) and T (system temperature, systematically varied over time). Zhai, Hunter, and Smith [33] present this equation but do not suggest ranges for either k or T.

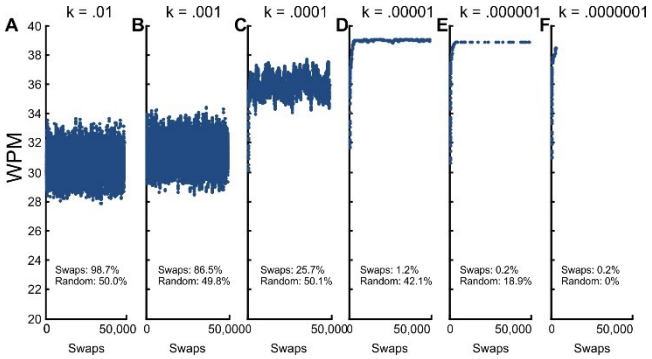

Figure 3. Each panel shows the random walk through the interface space via the Metropolis algorithm with a different value for scalar k with T (arbitrarily) at 10. Each data point represents the WPM for an interface arrangement with a higher efficiency than the "current" arrangement.

Figure 3 shows the results of example iterations of the Metropolis algorithm for different values of k, with a static T set at 10. Note that panels A and B stay near the mean random arrangement efficiency ~30 WPM); with this high of a k, many sub-optimal arrangements are kept, and thus the system never approaches an optimal value. Alternately, panels E and F have very few "kept" arrangements; these then are highly dependent on the starting arrangement and do not have the opportunity to get out of local minima. Panels C and D, however, show behavior closer to the goal. Panel C shows a WPM that hovers above the mean. Panel D shows similar behavior to E and F, but with some minor dips in efficiency that eventually lead to higher WPM. Optimal behavior

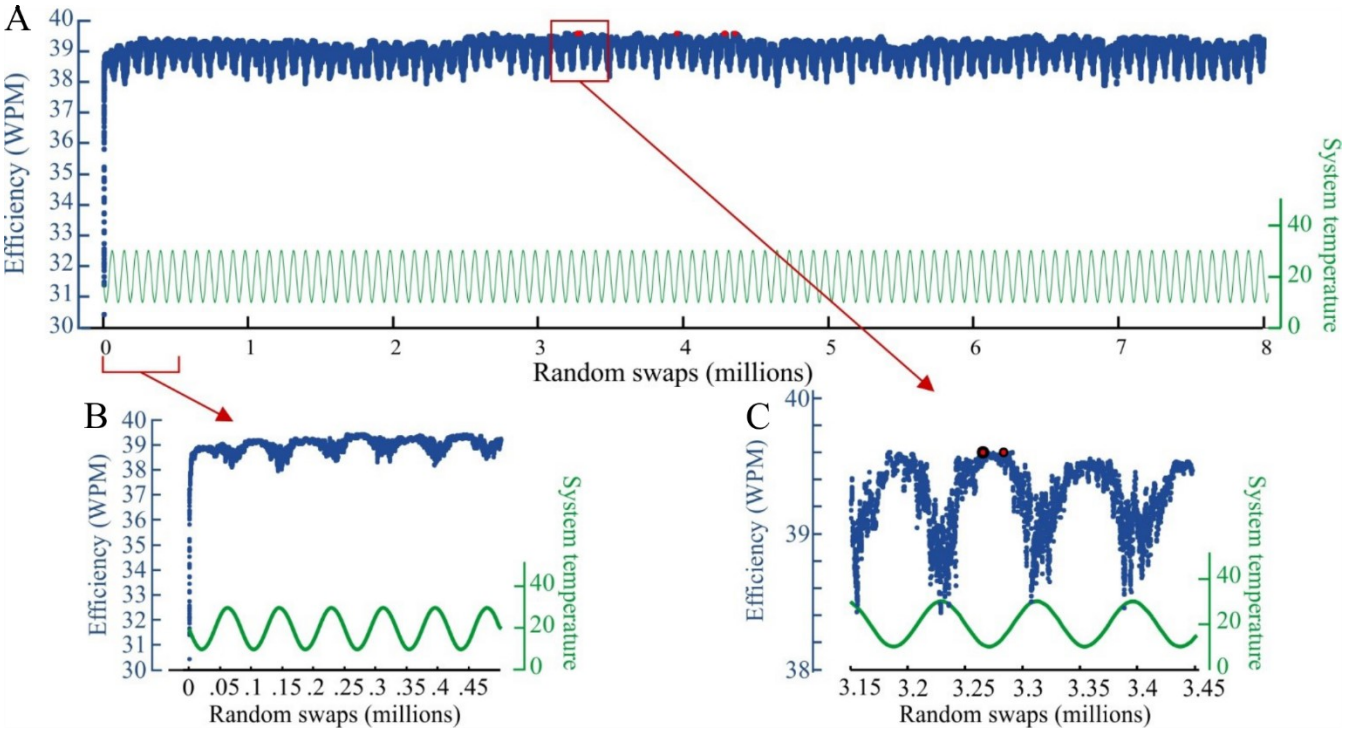

Figure 4. Metropolis algorithm. Top panel shows one iteration of the algorithm, consisting of 8 million random swaps and annealing system temperature. Panel B shows the typical process at the beginning of the algorithm, in which the efficiency quickly rises to the neighborhood of the final "optimized" version. Panel C shows how the annealing process (system temperature in green raising and lowering over time) allows the system to come out of local maxima in order to then approach the optimal solution (red dots).

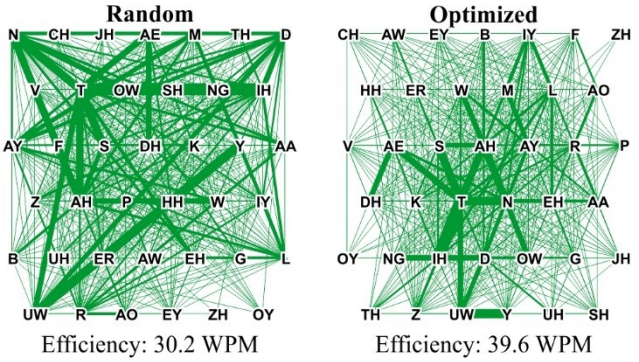

Figure 5. Visual comparison of results of Metropolis algorithm. Left shows a random organization of phonemes; right shows an optimized interface. Width of lines between two targets represents the likelihood of transitioning between them in the AAC conversational phrases corpus.

is likely in between these two numbers, as the algorithm optimizes by periodically choosing a less-optimal solution. Therefore, k was set at .00001 (Panel D), and system temperature was set to vary between 10 and 35, such that the final behavior of the algorithm was between Panels C and D of Figure 3.

For all interfaces, system temperature was varied sinusoidally between 10 and 35 at twelve cycles per million random swaps. The optimization ran for 8 million random swaps for each interface. An example optimization is shown in Figure 4.

2.4.3 Verification of Optimization Outcome

Figure 4 shows the results of one iteration of the optimization algorithm, which consisted of 8 million random swaps. The efficiency first increases rapidly from an average random efficiency (~30 WPM for the suggested AAC messages corpus) to near the final optimum efficiency (~39 WPM; Figure 4B). Next the annealing process (in green; Figure 4C especially) systematically increases and decreases the system temperature, thus increasing and decreasing the likelihood that the algorithm will accept an arrangement with a worse efficiency, as in Equation 4. This allows the system to come out of local maxima in order to then approach the "optimal" solution.

In addition to the efficiency calculations performed at each step by the Metropolis algorithm, representations of the results of the optimization process were evaluated visually to verify the function of the algorithm. Figure 5 shows two different arrangements of 39 phonemes. The left is a random organization, whereas the right shows an optimized arrangement based on the Suggested AAC corpus. The width of the lines represent the transition likelihood between each pair of phonemes. Note that while thick lines occur throughout the random interface (left), the length of thick lines are minimized in the optimized interface; this suggests that when users try to produce the words and phrases common in the corpus (e.g., "you" or /Y-UW/, and "don't" or /D-OW-N-T/), they will not need to move as far to select the required targets. The efficiency of the random interface shown here (30.2) is also the mean efficiency of 100,000 random phoneme layouts using the Suggested AAC corpus to evaluate efficiency.

2.5 Evaluation

Seven interfaces (Figure 6) were evaluated against five corpora (see Table 2). Interfaces A-E were optimized as stated in section *2.4 Metropolis Optimization Algorithm* using diphone probability statistics from each of the five respective corpora. Interface F (Alphabetic) was generated to be the same shape as Interfaces A-E, but with the phonemes arranged alphabetically according to their labels. The final interface (Figure 6G; Articulatory) was included to compare to previous studies [6, 7]. Interfaces F and G are included primarily as non-optimized controls in order to evaluate the efficacy of the optimization. Further, while we hypothesized that the Alphabetic and Articulatory interfaces would not provide optimal efficiencies, these interfaces are organized by rules that people can learn, and thus may help users quickly learn where different targets are on the interface. Although the interfaces were not evaluated empirically in this study, future studies may wish to do so. Additionally, if the differences in efficiency are small, users may choose to use the Articulatory or Alphabetic interfaces; whereas if the differences in efficiency are large, they may choose to use an optimized interface instead.

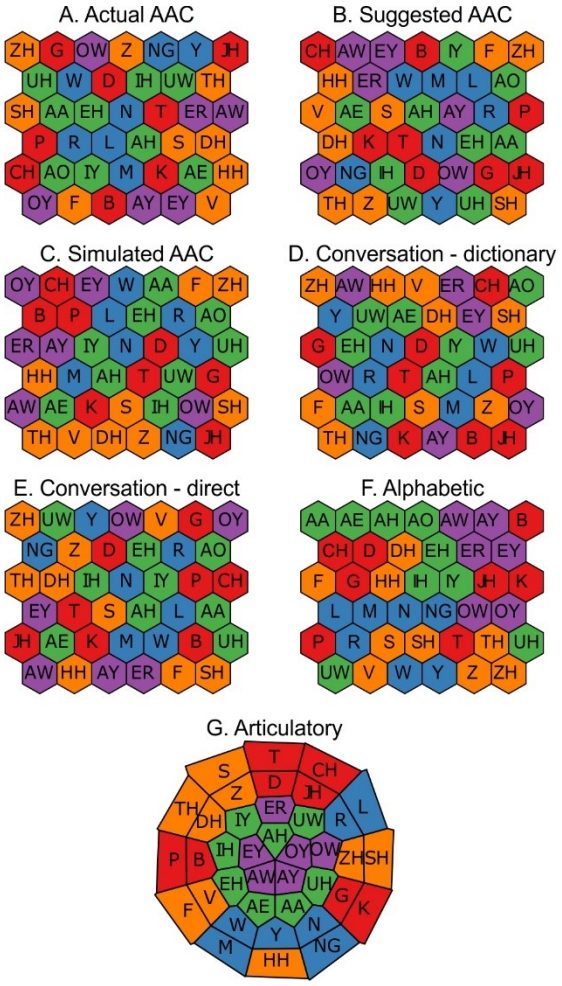

Figure 6. Interfaces developed and evaluated in this study. Target colors show rough groupings of phonemes: simple vowels in green, complex vowels (diphthongs, r-colored vowels) in purple, fricatives and affricates in orange, stops in red, and liquids, nasals, and semivowels in blue.

3. RESULTS

3.1 Corpora similarity

Figure 7 shows the correlation of diphone likelihoods between the different corpora tested here. Correlations were high, ranging from .70 to .86. Correlations between text resources (Actual AAC, suggested AAC, and simulated AAC) and conversational

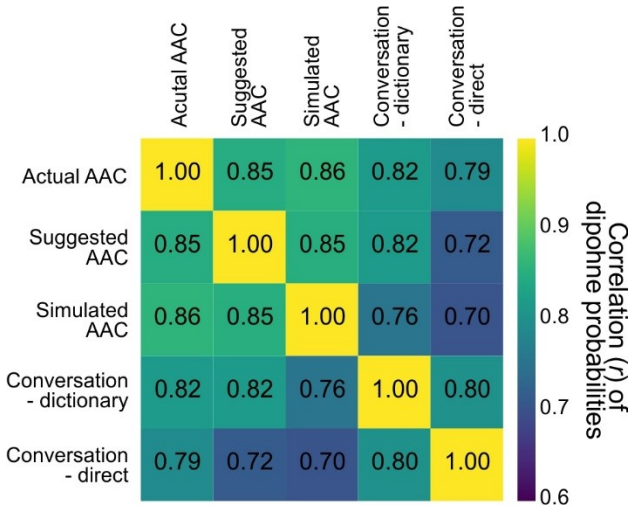

Figure 7. Similarity between phoneme-to-phoneme transition probabilities from different corpora (color represents r-value of Pearson's correlation between all diphone probabilities)

resources (Conversation-dictionary and Conversation-direct) were on the low-to-mid end of the range of correlations, ranging from .70 to .82, while those within text resources were the highest, from .85 to .86. Interestingly, the correlation between the Conversation-dictionary and Conversation-direct probabilities was only .80, despite both being derived from the same conversational source.

3.2 Interfaces

Figure 6 shows the different interfaces developed for this project (A-F) as well as the articulatory interface previously developed (G).

3.2.1 Optimized Interfaces

The first five interfaces (Figure 6A-E) were generated with the Metropolis algorithm as in *Section 2.4 Metropolis Optimization* Algorithm using the diphone probabilities from each respective corpus.

3.2.2 Alphabetic Interface

The alphabetic interface (Figure 6F) used the same layout and phonemes as the optimized interfaces, but arranged the phonemes in alphabetical order based on their label (see Figure 6).

3.2.3 Articulatory Interface

The articulatory interface (Figure 6G) has previously been described in [5]. Briefly, the targets on the interface were arranged manually in a circular layout, such that phonemes were organized based roughly on articulatory features (manner and place of articulation). Phonemes that are differentiated only by voicing (e.g., /TH/ and /DH/) are located at the same angle but different radii. Only 38 phonemes were used for this interface instead of the set of 39 used in the other interfaces in this study; as noted in Table 1, the phonemes /AA/ (*father*) and /AO/ (*ought*) were collapsed into one phoneme. Interface targets were allowed to be directly adjacent (Figure 6G) rather than leaving gaps in between targets as in [5]; this was so that widths were as large as possible in relation to distance between targets, as those in the other hexagonal interfaces are tightly packed.

3.3 Interface efficiency

Efficiency was calculated for each of the seven interfaces against each of the five testing corpora. Results are shown in Figure 8 in

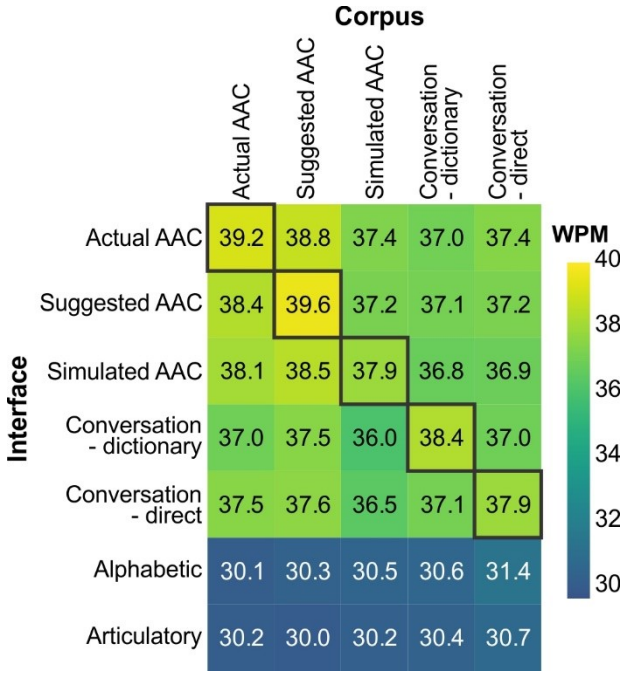

Figure 8. Efficiency in words per minute (WPM) for each interface tested with each corpus. Highlights on the diagonal show when the interface is being tested against the same corpus used to optimize its layout.

terms of WPM. The optimized interfaces had relatively high efficiencies across all testing sets, from 36.5 to 39.6 WPM. Generally efficiencies were highest when the testing corpus was the same as the corpus used to generate the diphone probabilities (shown in Figure 8 on the diagonal, highlighted in black). Variability in WPM for each interface across the different corpora was low, with ideal efficiency to lowest efficiency differing by only 1-3WPM. Efficiencies were lower with both interfaces not optimized by the Metropolis algorithm (30.0-31.4); these were even more consistent across testing corpora, with the Alphabetic only varying by 1.3 WPM across testing corpora, and the Articulatory only varying by 0.7 WPM depending on which corpus was used to evaluate its efficiency.

4. DISCUSSION

Interfaces not optimized for efficiency (Alphabetic; Articulatory) show efficiencies around 30WPM, which is near the mean random interface efficiency (30.2 WPM). Optimizing an interface using the Metropolis algorithm yields an improvement of around 9 WPM over a random arrangement and the Alphabetic and Articulatory interfaces. However, optimizing by one corpus and testing against another yields differences around 1-3 WPM.

4.1 Benefits of Alp habetic and Articulatory Interfaces

The Alphabetic and Articulatory interfaces may show advantages that are not captured in the efficiency calculation. The organization of the Alphabetic interface gives users some *a priori* information about where targets are located on the interface, as they are arranged via phonemic label. Figure 6F shows that targets arranged by label are also thus somewhat arranged by type of phoneme (note that all nasals are together; vowels are largely grouped). This would likely improve early communication rates via reducing visual search time when users are first learning to use the interface. The Articulatory interface similarly shows

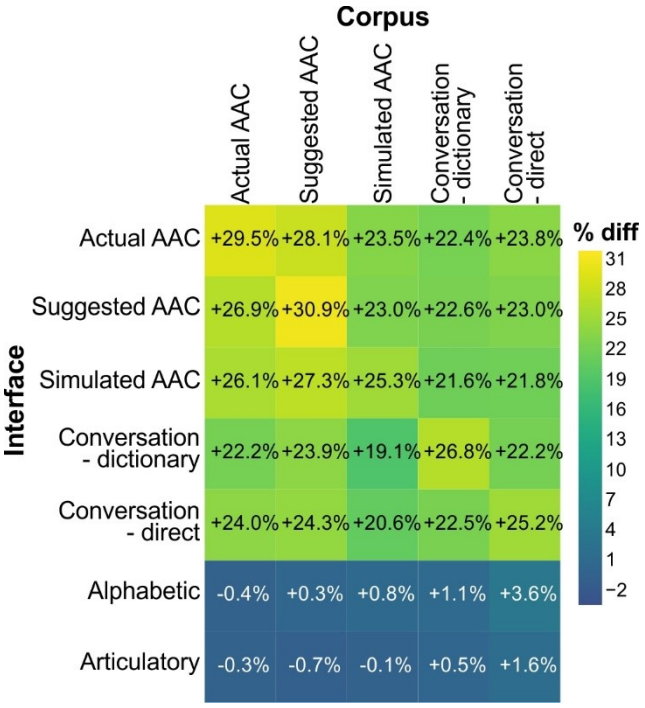

Figure 9. Percent difference in efficiency compared to a random arrangement of phonemes.

organization (Figure 6G), such that all vowels are in the center of the interface, with consonants surrounding them. In addition, the Articulatory interface pairs phonemes that are similar by manner and place of articulation (e.g., /f/ and /v/ are neighbors), which may have initial and ongoing benefits. First, this organization may allow faster learning of the target locations, similar to the Alphabetic interface. In addition, however, this leads to some error tolerance that is not seen in orthographic interfaces or the other phonemic interfaces. If a user overshoots the target and accidentally selects the neighboring pair, the output of may still retain intelligibility that it would not with other interfaces. For example, if a user intends to select /V-OI-S/ ("voice") and instead selects /F-OI-S/ or /V-OI-Z/, a listener would still likely understand that production in context.

4.2 Other Efficiency Calculations

Calculating efficiency in words per minute here relied on two main assumptions that may not hold for phonemic interfaces used by individuals employing alternate access methods.

First, the WPM calculation assumes five letter selections per word, which includes four characters and a space; phonemic interfaces do not require a space, and have fewer characters per word; taking these both into account improves the efficiency calculation for phonemic interfaces.

In addition, alternate access methods are noisier than typical access methods, such as a stylus or physical keyboard. The calculations here used Fitts' constants from a typical stylus to compare to previous work (see section *2.3.1 Fitts' constants*). If we used Fitts' constants derived from an individual with a spinal cord injury using electromyography (EMG) to control a cursor [30], the absolute difference between the interfaces changes. When recalculating Figure 6 using Fitts' constant b=1/2.6, (derived from line of best fit from [30], Figure 5b), the number of estimated words per minute reduces from 30-39 to 8-9 WPM. However, these scale linearly (when using the common assumption that Fitts' a=0). Figure 9 thus shows the improvement in efficiency in percent difference rather than in WPM, as WPM varies by input method, number of characters per word, and by individual skill.

A final way to consider the improvement of the optimization is not in words per minute, but rather as the time it would take to accomplish a task. Table 3 shows time estimates for producing a list of the 1004 suggested AAC messages (the Suggested AAC corpus) with different phonemic and orthographic on-screen interfaces, and using either a typical stylus movement time estimate (in which b=1/4.9) or the EMG cursor movement time estimate (in which b=1/2.6). For the phonemic interfaces, no spaces or punctuation were included. For the orthographic interfaces, no punctuation was included. Fitts' constant remained a=0 for all calculations, as did the special case of movement time when tapping the same target twice (.127s). In these calculations, then, no assumptions were made as to the number of phonemes or letters per word.

The times in Table 3 represent long-term interface usage (i.e., combined time to produce 1004 utterances) using different phonemic (Suggested AAC and Alphabetic, from this study) and orthographic interfaces (orthographic Metropolis interface from [33]; QWERTY keyboard in common use). The optimized phonemic interface shows substantial improvements over orthographic input methods. These improvements increase when the input method is noisy, such as the EMG cursor [30]. Thus, while users with access to typical stylus use may not choose to switch to a phonemic interface, those for whom access is more time-consuming and difficult may find the initial costs of learning a phonemic interface worthwhile (e.g., decreasing time to produce messages from 210 hrs to 102 hrs – roughly 50%). Note, however, that these do not include any prediction, which improves both phonemic and orthographic input rates substantially [25, 27].

Table 3. Time Estimates to Produce Suggested AAC Corpus

Interface	**Access Method**	
	Stylus (b = 1/4.9)	**EMG cursor by person with spinal cord injury** (b = 1/2.6)
Suggested AAC (phonemic)	54 hrs	102 hrs
Alphabetic (phonemic)	70 hrs	133 hrs
METROPOLIS (orthographic) [33]	78 hrs	147 hrs
QWERTY (orthographic; square targets)	111 hrs	210 hrs

4.3 Clinically Meaningful Speed Improvements

It is not yet clear what degree of improvement in efficiency is clinically meaningful, particularly as AAC users have many different access methods and preferences. Therefore it is also unclear whether the 5-8% improvements due to optimizing per corpus are worthwhile. While producing a new optimized interface once a corpus is obtained is not particularly difficult, it can be somewhat time consuming (converting a given corpus to phonemes often involves some level of hand-correcting for out-of-dictionary terms; running the actual optimization process as described in *2.4 Metropolis Optimization Algorithm* takes approximately twelve hours of computing on a shared computing cluster). Further, it can be difficult to obtain an appropriate

corpus. If a user already has an AAC device, they may be willing to allow their AAC specialist to record their usage for some length of time. Although this would be an ideal situation for optimizing an individual AAC interface, only some AAC devices have this recording capability, and recording a person's communication output has privacy concerns.

4.4 Future Directions

4.4.1 Empirical evaluations

An empirical evaluation of these phonemic interfaces is a necessary next step. Although it was not possible to thoroughly evaluate all of the interfaces presented here in AAC users, an empirical evaluation of a small number of interfaces using only one testing set can now be completed to validate the theoretical results. This evaluation should be carried out by users with a variety of motor impairments and access methods. Empirical evaluations should be done to compare the interfaces developed in this study to other existing phonemic interfaces [3, 20, 24, 27].

4.4.2 Individualized optimizations

This paper includes the technical details of the varied quantitative processes that were involved in generating and evaluating the interfaces. These are included specifically so that others can recreate them and produce interfaces optimized based on any given corpus or with weighting factors other than just Fitts'-based efficiency. For example, if a particular user finds left-and-right movements less fatiguing than up-and-down movements, an additional weighting factor could be added for reduced efficiency when the next target is above or below the current target rather than on the same row. Alternate efficiency formulae and weighting could also allow these same algorithms to optimize interfaces intended for use with scanning rather than direct selection methods. In that case, efficiency would be calculated based on time from the onset of scanning to the target's selection, rather than the Euclidean distance between two targets.

4.4.3 Prediction

Finally, another vital way to optimize communication rates is to incorporate online prediction. Studies of existing phonemic interfaces have focused on prediction as the primary way of increasing communication rates [20, 24, 27], without the offline optimizations shown here, and predictive language models can increase phonemic interface communication rates by as much as 100% [24]. Adding prediction to the interfaces presented here is needed in order to compare to other phonemic or orthographic interfaces and to maximize communication rates.

5. CONCLUSIONS

Phoneme-to-phoneme transition likelihoods are highly correlated across corpora, particularly corpora generated from text or AAC use instead of from oral conversation. Optimization based on any corpus increased efficiency from random layouts or from the Alphabetic and Articulatory interfaces by 19-31%. Optimizing and testing on the same corpus led to efficiency improvements of 5-8%, compared to testing on other corpora. Therefore, if possible, future phonemic interfaces should be optimized via a corpus from the intended user's speech. If this is not possible, however, optimization still improved efficiency using all testing corpora, suggesting that choosing an optimized corpus is indicated over other layouts. Future directions include empirical testing and adding prediction to further increase communication rates.

6. ACKNOWLEDGMENTS

This research is supported by NIH grant F31 DC014872 and NSF grant 1452169. Our thanks to Alfonso Nieto Castañón and Frank Guenther for access to their phonemic interface, which was developed under NIH grant R01DC002852.

7. REFERENCES

[1] Beichl, I. and Sullivan, F. 2000. The Metropolis algorithm. *Computing in Science & Engineering*. 2, 1 (2000), 65–69.

[2] Beukelman, D.R. et al. 2007. AAC for adults with acquired neurological conditions: a review. *Augmentative and Alternative Communication*. 23, 3 (2007), 230–242.

[3] Black, R. et al. Introducing the PhonicStick: Preliminary evaluation with seven children.

[4] Brumberg, J.S. et al. 2010. Brain-computer interfaces for speech communication. *Speech Communication*. 52, 4 (2010), 367–379.

[5] Cler, M.J. et al. 2016. Surface electromyographic control of a novel phonemic interface for speech synthesis. *Augmentative and Alternative Communication*. 32, 2 (2016), 120–130.

[6] Cler, M.J. et al. 2016. Surface electromyographic control of a novel phonemic interface for speech synthesis. *AAC: Augmentative and Alternative Communication*. 32, 2 (2016).

[7] Cler, M.J. et al. 2014. Surface electromyographic control of speech synthesis. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference*. 2014, (2014).

[8] Cler, M.J. and Stepp, C.E. 2015. Discrete vs. continuous mapping of facial electromyography for human-machine-interface control: Performance and training effects. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*. 23, 4 (2015), 572–580.

[9] Context Specific Messages (Suggested by AAC Specialists): *http://cehs.unl.edu/aac/aac-messaging-and-vocabulary/*. Accessed: 2017-01-01.

[10] Dvorak, A. and Dealey, W.L. 1932. US2040248. US2040248 A. 1932.

[11] Frey, L.A. et al. 1990. Eye-Gaze Word Processing. *IEEE Transactions on Systems, Man and Cybernetics*. 20, 4 (1990), 944–950.

[12] Higginbotham, D.J. et al. 2007. Access to AAC: present, past, and future. *Augmentative and alternative communication (Baltimore, Md. : 1985)*. 23, 3 (2007), 243–257.

[13] Huo, X. et al. 2008. Introduction and preliminary evaluation of the Tongue Drive System: Wireless tongue-operated assistive technology for people with little or no upper-limb function. *Journal of Rehabilitation Research & Development*. 45, 6 (2008), 921–930.

[14] Kushler, C. 1998. AAC: Using a Reduced Keyboard. *CSUN Conference on Technology for persons with disabilities* (California State University, Nortridge CA, 1998).

[15] Lesher, G. et al. 1998. Techniques for augmenting scanning communication. *Augmentative and Alternative Communication*. 14, June (1998), 81–101.

[16] Lesher, G.W. et al. 1998. Optimal character arrangements for ambiguous keyboards. *IEEE Transactions on Rehabilitation Engineering*. 6, 4 (1998), 415–423.

[17] MacKenzie, I.S. and Zhang, S.X. 1999. The design and

evaluation of a high-performance soft keyboard. *CHI 99 Conference on Human Factors in Computing Systems*.

[18] Mayzner, M.S. and Tresselt, M.E. 1965. Tables of single-letter and digram frequency counts for various word-length and letter-position combinations. *Psychonomic Monograph Supplements*. Vol 1(2), (1965), 13–32.

[19] Pitt, M.A. et al. 2005. The Buckeye corpus of conversational speech: labeling conventions and a test of transcriber reliability. *Speech Communication*. 45, (2005), 89–95.

[20] Schroeder, J.E. 2005. Improved spelling for persons with learning disabilities. *20th Annual International Conference on Technology and Persons with Disabilities* (Northridge, CA, 2005).

[21] Sevcik, R.A. and Romski, M.B.T.-A.L. 2000. AAC: More Than Three Decades of Growth and Development. 5, 19 (Jul. 2000), 5.

[22] Textware Solutions 1998. The Fitaly one-finger keyboard.

[23] The Carnegie Mellon Pronouncing Dictionary [cmudict. 0.6]: 2005. *http://www.speech.cs.cmu.edu/cgi-bin/cmudict*.

[24] Trinh, H. et al. 2012. iSCAN: A Phoneme-based Predictive Communication Aid for Nonspeaking Individuals. *ASSETS'12*.

[25] Trnka, K. and McCoy, K.F. 2007. Corpus studies in word prediction. *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility - Assets '07*. (2007), 195.

[26] Unabridged Vocabulary Lists with Use Statistics - AAC User: *http://cehs.unl.edu/aac/aac-messaging-and-vocabulary/*.

[27] Vertanen, K. et al. 2012. Applying prediction techniques to phoneme-based AAC systems. *NAACL-HLT 2012 Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)*.

[28] Vertanen, K. and Kristensson, P.O. 2011. The imagination of crowds: conversational AAC language modeling using crowdsourcing and large data sources. *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (2011), 700–711.

[29] Wandmacher, T. and Antoine, J.-Y. 2006. Training language models without appropriate language resources: Experiments with an AAC system for disabled people. *Proceedings of LREC* (2006).

[30] Williams, M.R. and Kirsch, R.F. 2016. Case study: Head orientation and neck electromyography for cursor control in persons with high cervical tetraplegia. *Journal of Rehabilitation Research and Development*. 53, 4 (2016), 519–530.

[31] Williams, M.R. and Kirsch, R.F. 2008. Evaluation of head orientation and neck muscle EMG signals as command inputs to a human-computer interface for individuals with high tetraplegia. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*. 16, 5 (2008), 485–496.

[32] Wolpaw, J.R. et al. 2002. Brain-computer interfaces for communication and control. *Clinical Neurophysiology*. 113, 6 (2002), 767–791.

[33] Zhai, S.M. et al. 2002. Performance optimization of virtual keyboards. *Human-Computer Interaction*. 17, 2–3 (2002), 229–269.