HHS Public Access

Author manuscript

J Neural Eng. Author manuscript; available in PMC 2017 July 03.

Published in final edited form as:

J Neural Eng. 2016 June ; 13(3): 031002. doi:10.1088/1741-2560/13/3/031002.

Integrating Language Models into Classifiers for BCI Communication: A Review

W Speier[1,2], **C Arnold**[2,3], and **N Pouratian**[1,3,4,5]

[1]Department of Neurosurgery, University of California, Los Angeles, CA 90095, USA

[2]Medical Imaging Informatics Group, University of California, Los Angeles, CA 90095, USA

[3]Bioengineering Department, University of California, Los Angeles, CA 90095, USA

[4]Neuroscience Interdepartmental Program, University of California, Los Angeles, CA 90095, USA

[5]Brain Research Institute, University of California, Los Angeles, CA 90095, USA

Abstract

The present review systematically examines the integration of language models to improve classifier performance in brain-computer interface (BCI) communication systems. The domain of natural language has been studied extensively in linguistics and has been used in the natural language processing (NLP) field in applications including information extraction, machine translation, and speech recognition. While these methods have been used for years in traditional augmentative and assistive communication (AAC) devices, information about the output domain has largely been ignored in BCI communication systems. Over the last few years, BCI communication systems have started to leverage this information through the inclusion of language models. Although this movement began only recently, studies have already shown the potential of language integration in BCI communication and it has become a growing field in BCI research. BCI communication systems using language models in their classifiers have progressed down several parallel paths, including: word completion; signal classification; integration of process models; dynamic stopping; unsupervised learning; error correction; and evaluation. Each of these methods have shown significant progress, but have largely been addressed separately. Combining these methods could use the full potential of language model, yielding further performance improvements. This integration should be a priority as the field works to create a BCI system that meets the needs of the ALS population.

1. Introduction

Motor neuron disorders such as amyotrophic lateral sclerosis (ALS) and brainstem injuries can disrupt the neural transmission, resulting in the reduction of muscle control and impairing a patient's ability to communicate. Assistive communication technologies exist that can help such patients by providing indirect communication methods based on alternative muscle movements such as eye tracking [1]. These technologies, however, can be difficult to implement and maintain in patients due to disease progression [2]. Brain computer interfaces (BCI) can restore these 'locked-in' patients' ability to communicate by detecting their intent from electroencephalogram (EEG) signals and translating them into

computer commands, possibly providing a more robust solution due to the relative preservation of cortical physiology [3].

1.1 BCI communication

Several different BCI systems have been developed to enable communication for 'locked-in' patients by translating EEG signals into simulated keyboard input. The most common BCI communication system is the P300 speller which presents the user with a grid of characters on a computer monitor [4]. The user is instructed to focus on a target character while groups of characters are illuminated (i.e., "flashed") in a pseudo-random manner. When the target character is flashed, a response known as the P300 signal is evoked and detected by EEG. A classifier detects these responses and, after combining the responses from several trials, determines the character that was most likely the subject's target.

A similar BCI communication system is the rapid serial visual presentation (RSVP) speller [5]. Like the P300 speller, this system elicits P300 signals by serially presenting visual stimuli in a graphical interface. In the RSVP speller, however, the user focuses on the center of the screen where characters appear in a random sequence. This system is generally slower than the P300 speller because showing characters one at a time requires more time to display all possibilities. It has the advantage of being gaze-independent, which improves signal fidelity, particularly in neurologically impaired patients with restricted eye movement.

Similar to visual systems, evoked responses (ERPs) can be generated by auditory stimuli. Systems such as the auditory multi-class spatial ERP (AMUSE) system present the user with a series of distinct auditory stimuli [6]. These stimuli vary based on pitch and/or location, with each combination assigned to a specific character. As in the visual P300 speller, the target stimulus elicits an ERP which is detected by the classifier in order to choose a character. Because the correspondence between characters and tones is not obvious, a graphical interface is often included. Given enough training, a user could learn the associations and the system could function without a visual interface. This would eliminate any dependence on eye gaze, but would have additional requirements for the user's environment such as a lack of distracting noise.

Several other systems have recently been developed which utilize other neural signal paradigms. Motor imagery has been used to navigate binary menus by mapping different imagined movements (e.g., right hand vs. left hand) to different options [7]. This paradigm has been adapted to create a spelling system by using these decisions to perform a binary search through the alphabet [8]. Similarly, motor imagery is used in the hex-o-spell system where the subject uses imagined movement to stop a cursor rotating between six sets of characters [9]. This system reduces the time to choose a character by making selections among six options rather than two, reducing the number of decisions required to select a character from 26 characters from five to two.

Steady state visually evoked potentials (SSVEP) have also been utilized in BCI communication systems. In SSVEP systems, multiple targets exist on screen which flicker a different frequencies. When the user focuses on one of the targets, a visually evoked signal in the user's EEG matches the frequency of the flashing. When this frequency is observed in

the signal, the system selected the corresponding item. Because of limits in the number of distinct frequencies that can be identified, systems generally either use a series of menus [10] or move a cursor [11] in order to select from the set of possible characters. Other systems have combined the P300 speller and SSVEP systems by flashing characters as in the P300 speller and simultaneously having the nonflashing characters flicker at different frequencies [12,13].

Because the signal-to-noise ratio (SNR) in EEG is relatively low, several trials are usually combined in order to correctly classify responses. The resulting typing speed is therefore slower than required for adoption [2], prompting significant research in the optimization of BCI communication systems, both with respect to speed and accuracy. Approaches that have been adopted to accomplish such optimization in the P300 speller include varying the grid size [14], modifying stimulus presentation paradigms [15,16], optimizing system parameters [17,18], changing the visual stimulus [19], and adopting different signal classification algorithms [20–23]. Studies have also used electrocorticography (ECoG) to increase the SNR for BCI communication [24,25].

1.2 Language Models

While the P300 speller is designed to provide a means for communication, BCI systems have not traditionally taken advantage of existing knowledge about the language domain. In 2000, Donchin et al. noted that “there are substantial sequential dependencies in English,” which could be utilized in classification [26]. However, traditional classification systems treat typing as a series of independent selections from a set of characters with no prior information. The domain of natural language has been well studied in other contexts and this knowledge can be used to aid in any communication system [27]. By exploiting known patterns and structures inherent in language, a bias can be added to a communication system which can improve typing speed and accuracy as well as adding other features such as word completion or automatic error correction. These methods are already widely employed in electronic communication systems such as word processing [28] and text messaging[29,30], but have only recently been considered for BCI communication.Language information can be included in a BCI system by storing a model of typical text that the system expects to generate as output. One example of such a model is a simple dictionary where the BCI system looks up the letters being typed to verify that they form valid words. This type of model can work as a filter that ensures that the system generates words that are valid in the language. If a string of characters does not start any word in the dictionary, the system can assume that there was a mistake and try to fix it by changing characters to match another word. Such a system can also look up words that start with the current output to generate “guesses” of the complete target word. These “guesses” can then be presented to the user so that the remainder of the word can be completed at once rather than by individual character selections, thereby increasing typing speed. One problem with these word based methods is that they are limited to the words that are seen during language model training and generally cannot handle out of vocabulary (OOV) words.

Other language models attempt to model character patterns based on corpora of existing text. These models can provide a probability distribution for target characters based on previous

selections, which can be used as a prior probability for future selections. The corpora for these models are generally publicly available sets of text from natural language research groups [31,32]. The simplest such model captures patterns by finding the relative frequency of n-grams, sequences of n consecutive characters. These models are created by parsing through a corpus of text and counting the number of occurrences of these sequences. The conditional probability of a character given the previous n−1 characters can then be computed:

$$p(x_t|x_{t-1},\ldots,x_{t-n+1})=\frac{c(x_t,\ldots,x_{t-n+1})}{\sum_{x_t}c(x_t,\ldots,x_{t-n+1})}$$

where $c(x_{t-2}, x_{t-1}, x_t)$ is the number of occurrences of the string '$x_{t-2}x_{t-1}x_t$' in the corpus. The value for n is determined based on the tradeoff between the amount of information contained in the model and the complexity of the model. The number of n-grams is exponential in n, so classification algorithms can run too slowly for online classification with a large value of n. Also, the system can be undertrained if the training corpus is not large enough to represent all possible n-grams. Of note, context-relevant corpora can theoretically be employed and provide improved performance over a general corpus.

1.3 Scope of the review

The use of language modeling in text entry systems for individuals with severe motor disabilities has been prevalent in the field of augmentative and assistive communication (AAC) for over 20 years, including applications such as word completion [33], grid scanning [34–37], and interface optimization [38]. Language modeling in BCI communication has followed similar paths, ranging from changes in the user interface to modifying the signal classification algorithms. However, BCI communication systems differ from other AAC systems because of the time consuming and challenging intent classification that must occur at each step in the process. This difference indicates that language modelling in BCI classifiers has great potential as users who have difficulty typing with a communication system stand to gain the most from language integration [33]. There are additional challenges as well because traditional language modelling methods need to be adapted to address the noise and dimensionality of neural data. The current review focuses on how language information can be used by a classifier to improve the speed and accuracy of decision making when classifying neural signals for BCI communication.

A review of language models in BCI communication systems has been previously conducted which was more broad in scope, focusing mainly on changes to the graphical interface rather than the classifier [39]. Many of the methods covered in the previous review are similar to those already used in other AAC systems such as arranging the interface optimally [40,41] or dynamically [8] based on letter frequency and choosing between groups rather than individual characters [42]. The review makes a cursory mention of using language information in classification, but does not do a thorough review of the area or include details of the methods. In addition, two years have passed since the publication of the previous review, so it does not include recent work. Because of the recent nature of this field, many groups are engaged in parallel tracks to try to utilize language information. Many of these

systems could potentially work in conjunction with one another or could benefit from building off of previous works. The goal of this review is to give an overview of the different projects that have been completed in this area and to serve as a primer for integrating these methods into a unified, next generation system.

Based on the literature cited, we identified and discuss 7 different domains in which language models have been used to enhance BCI communication: (i) word completion; (ii) signal classification; (iii) integration of process models; (iv) dynamic stopping; (v) unsupervised learning; (vi) error correction; and (vii) evaluation metrics. The review of current literature is followed by a discussion of how these methods interact and whether they can be combined to further improve BCI communication rates. Finally, future directions for language model integration are discussed.

2. Methods

The PubMed and IEEE databases were searched using the query:

> (“language model” OR “natural language processing” OR “NLP” OR “predictive spelling”) AND (“brain computer interface” OR “BCI” OR “P300” OR “SSVEP”)

The initial search produced 29 articles. These articles were then filtered by reading their titles and abstracts to ensure that they fit within the scope of the review. Articles were further excluded if they were theses or conference proceedings that duplicated journal articles by the same authors, resulting in 13 articles. The query was then expanded by including any references in these articles which fell within the scope of this review, resulting in a final set of 28 articles.

3. Results

Among the 28 selected articles, two systems used a motor imagery based system, one used an SSVEP system, 20 used the P300 speller, and six used the RSVP speller. Six articles proposed systems that included word completion. Twenty of the articles proposed methods for including language models in the classifier, including threshold methods, naïve Bayes, reliability-based automatic repeat request (RB-ARQ), partially observable Markov decision processes (POMDPs), and hidden Markov models (HMMs). Six articles utilize language models to make further improvements to the system including dynamic stopping, unsupervised training, and integration into invasive BCI systems. Eight articles addressed error correction, both manual and automatic. Two articles introduced new evaluation metrics taking language models into consideration.

3.1 Word completion

The earliest application of language information in BCI was integrating word and phrase completion based on previous selections [43], similar to methods employed in text messaging [30] and AAC devices [33]. These systems contain language models that consist of dictionaries, usually with weightings based on the frequency of usage of the word. After each character selection, the system performs a lookup of the partially completed word and returns potential completions. These completions are then presented as selection options.

The user then has the option of continuing to spell by selecting individual letters, or selecting one of the completions. This method has the potential to increase typing speed by allowing the user to type multiple characters at once with a single selection. Errors can potentially be harder to correct in this system because an incorrect completion could result in multiple incorrect characters that would then all need to be deleted. This problem can be mitigated by adding additional commands such as “undo,” which can remove all of the characters at once.

Ryan et al. [43] conducted the first study using such a system. Their implementation ran a P300 speller concurrently with the WordQ2 word completion software (version 2.5, Quillsoft, Ltd., Toronto, ON). WordQ2 is assistive software designed to help people with difficultly writing by suggesting word completions, helping the user with spelling. Middleware was developed that routed the P300 output as input to WordQ2, which used dictionary lookups to find potential word completions. The top completion suggestions were then sent from WordQ2 to be presented to the user. The ten number spaces in the P300 grid were remapped to WordQ2 commands such as selection of a completed word or undoing a previous command. Their study showed that accuracy using this system decreased due to the added complexity of the task, but typing speed increased because of the ability to select multiple characters at once using word completion.

Lee et al. [44] presented a similar dictionary lookup scheme in a menu-based motor imagery system. In their system, the user selects a series of numbers, each mapped to a group of characters based on the T9 predictive text system used in some older mobile phones. For instance, if a user wanted to spell the word “good,” the string “3552” would be selected because “3” maps to “ghi,” “5” maps to “mno,” and “2” maps to “def.” These numbers could also map to the word “home,” so the user needs a way to choose between the possible target words. The system handles this by providing a column of words that match the typed numbers, which the user selects from once the word is completed. Unlike the Ryan et al. system, this method does not allow the user to type multiple characters at one time, and actually increases the number of necessary selections by one. Instead, this system improves typing speed by reducing the number of choices, making decisions easier for the classifier and reducing the time required to cycle through possible selections.

Kaufmann et al. [47] integrated a dictionary lookup scheme for common German words into the P300 system. Their system created a list of words that appeared in German internet pages and sorted them by the number of times the occurred. After each selection, their algorithm scanned the list for words starting with the current partially typed word. The top six matches were returned and were presented in the first column of the P300 speller matrix. To account for erroneous completions, they also provided a “delete word” option in the grid which would delete back to the last previously typed space.

3.2 Classification

While automatic completion can increase typing speed by reducing the number of selections required, embedding a language model into the classifier has the potential to increase the accuracy and reduce the amount of time required for individual selections. In general, these systems work by representing common character patterns in a language model, usually

generated from a corpus of text. These patterns are used to create a probability distribution over the next character to be selected given the text already typed, $p(x_t|x_{t-1}, ..., x_0)$. This probability distribution is then used to bias the system in favor of those characters that are more likely to be typed given knowledge of letter patterns in language. For instance, if the user is typing in English and the previous character is "Q," the system can be fairly certain the next character is "U" before even considering the corresponding EEG signal. This method effectively reduces the number of characters the system considers, making decisions easier and requiring less data before making an accurate selection.

Speier et al. [45] presented the first such system, which incorporated trigram probabilities through a naïve Bayes classifier. The system converts the output from a traditional classifier, linear discriminant analysis (LDA) [68], into a conditional distribution by assuming that the resulting scores would follow a normal distribution for each class

$$p\left(y_t^i|x_t\right) \propto \begin{cases} \frac{1}{\sqrt{2\pi\sigma_a^2}} e^{-\frac{1}{2\sigma_a^2}(y_t^i-\mu_a)^2} & if\ x_t \in A_t^i \\ \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{1}{2\sigma_n^2}(y_t^i-\mu_n)^2} & if\ x_t \notin A_t^i \end{cases}$$

where μ_a, μ_n, σ_a^2, and σ_n^2 are the means and variances of the scores for attended and nonattended flashes, respectively, and A_t^i is the set of characters flashed in group i for letter t. The prior probability was determined using a trigram character model

$$p(x_t|x_{t-1}, x_{t-2}) = \frac{c(x_t, x_{t-1}, x_{t-2})}{\sum_{x_t} c(x_t, x_{t-1}, x_{t-2})}$$

The posterior probability distribution over the possible target characters was found by multiplying the probability of the observed signals by the prior probability based on trigram counts.

$$p(x_t|\boldsymbol{y}_t, x_{t-1}, \ldots, x_0) \propto p(x_t|x_{t-1}, \ldots, x_0) p(\boldsymbol{y}_t|x_t, \ldots, x_0)$$

After normalization, this distribution yields the probabilities that each of the characters were the attended target. Once the system is ready to make a selection the character with the highest probability is selected. The probability distribution over the characters also create opportunities for other improvements such as dynamic stopping (see section 3.4) and error correction (see section 3.6).

Other groups have extended the trigram model to create models with other types of n-grams. Samizo et al. [54] tested the relative performance of their subjects using unigrams, bigrams and trigrams. They found that trigrams provided the fastest typing speed as they were able to capture a larger variety of language patterns. Orhan et al. [46] further extended the model to use 6-grams, which could capture larger character patterns, including many complete words.

These studies show that increasing the length of the patterns captured allows the model to better represent language, resulting in improved performance. However, as the model complexity increases, the amount of data required to train it increases exponentially, known as the curse of dimensionality. Thus, larger values of n will reduce the precision of the prior probabilities using an n-gram model as the training corpus will not be large enough to observe all possible character combinations. For instance, any combination that does not occur in the corpus given a zero probability, regardless of whether the string is possible in the language. This is a particular problem when typing uncommon strings that might not have occurred in the training corpus, such as OOV words. When designing a system, then, one must trade off the size of the model with the precision of its estimates, based on the size of the training corpus.

One way to mitigate the concerns of undertraining the system is to incorporate smoothing into the model. Smoothing works by moving probability mass in areas of a model that did not have sufficient observations in the training corpus. These methods will essentially default to a smaller, easier to train model in cases where the larger model was not sufficiently trained. Orhan et al. [46] used Witten-Bell smoothing [69] in the 6-gram model mentioned above. This method uses multiple models and creates the probabilities using a weighted average of the probabilities. In this example using a 6-gram and a 5-gram model, the probability becomes

$$p(x_t|x_{t-5:t-1})=\lambda\hat{p}(x_t|x_{t-5:t-1})+(1-\lambda)\hat{p}(x_t|x_{t-4:t-1})$$

where $\hat{p}(x_t|x_{t-5:t-1})$ and $\hat{p}(x_t|x_{t-4:t-1})$ are the 6-gram and the 5-gram probabilities, respectively. The value of λ is determined so that it is close to 1 when the 6-grams are common in the corpus, and close to 0 when the 6-grams are less common and the model is not trained as well.

$$\lambda=\frac{c(x_{t-5:t-1})}{c(x_{t-5:t-1})+T(x_{t-5:t-1})}$$

where $T(x_{t-5:t-1})$ is the number of distinct characters that follow $x_{t-5:t-1}$ in the corpus and $c(x_{t-5:t-1})$ is the total number of occurrence of $x_{t-5:t-1}$ in the corpus. The probability distribution will be close to $\hat{p}(x_t|x_{t-5:t-1})$ in cases where sufficient training data exists and will otherwise approach $\hat{p}(x_t|x_{t-4:t-1})$. Witten-Bell can be reapplied between the 5-gram probability and the 4-gram probability in cases where the 5-gram is uncommon, and can cascade as necessary until a fully trained model is reached.

Kindermans et al. [55] used an alternative smoothing method, Kneser-Ney smoothing [70]. Like Witten-Bell, this method moves probability from complicated to simple models in cases with insufficient training data. In this model, a fixed value, δ, is subtracted from the counts using the more complicated models and the remaining probability is assigned using the simpler model.

$$p(x_t|x_{t-5:t-1}) = \frac{\max(c(x_{t-5:t}) - \delta, 0)}{c(x_{t-5:t-1})} + \frac{\delta}{c(x_{t-5:t-1})} N_{1+}(x_{t-5:t-1}, \cdot) p(x_t|x_{t-4:t-1})$$

where $N_{1+}(x_{t-5:t-1},\cdot)$ is the number of distinct 6-grams that start with $x_{t-5:t-1}$ in the corpus. Like Witten-Bell, this approach can be applied on progressively simpler models until a fully trained model is reached.

3.3 Process models

A more sophisticated approach to language integration treats spelling as a process model. In these methods, typing is treated as a state model representing the user's target characters. While the subject is focusing on a character, the system is in the corresponding state. When the character is selected, the system transitions to the state represented by the next character with a transition probability determined from character patterns in natural language. In general, this model is unobserved, so estimations of the current state must be made based on the observed EEG signal and the transition probabilities determined by the language model.

Park and Kim [48] created the first such model using a partially observable Markov decision process (POMDP). In a POMDP, typing is modeled as a series of discrete time points. At each time point, the system is in one state in the model, it receives an input, and it performs an action. In this implementation, the states of the system are characters being typed, the input is an EEG response to a single stimulus, and an action is either continuing to look at the current character, or to make a decision about the current character and transition to another character. Probabilities of being in a state were computed based on bigram probabilities. The probability of being in state $\boldsymbol{x}_t$ was the total probability of being in any state $\boldsymbol{x}'_{t-1}$ and transitioning into that state

$$p(\boldsymbol{x}_t|\boldsymbol{y}_t, \ldots, \boldsymbol{y}_0) \propto p(\boldsymbol{y}_t|x_t) \sum_{\boldsymbol{x}_{t-1}} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{t-1}, \ldots, \boldsymbol{y}_0)$$

The decision to transfer to a new character in this method is dependent on value and policy functions that are optimized based on the state probabilities. This optimization is generally intractable, and requires estimation based on approximation algorithms. In their study, the point-based value iteration (PBVI) algorithm [71,72] was employed to determine the value and policy functions for the system.

Orhan et al. [49] and Ulas and Cetin [52] created similar systems that models BCI spelling as a hidden Markov model (HMM) in an offline setting with trigram transition probabilities. This methods was later extended by Speier et al. [60] into an online system that incorporated dynamic stopping and automatic error correction (see sections 3.4 and 3.6). Similar to the POMDP method, this system models typing with a state model where a single state represents attempting to spell a target character with transitions based on the co-occurrence of characters in a language model. The model itself is not observable, so the system must estimate the location in the model based on indirect observations. Unlike the POMDP

method, a time point in this model represents the total amount of time that the user focuses on a single character and the associated observations are all of the stimulus responses for that character. The system computes the probability distributions in a similar manner, modified to account for the trigram model

$$p(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}|\boldsymbol{y}_t, \ldots, \boldsymbol{y}_0) \propto p(\boldsymbol{y}_t|x_t) \sum_{\boldsymbol{x}_{t-2}} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}) p(\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}|\boldsymbol{y}_{t-1}, \ldots, \boldsymbol{y}_0)$$

The optimal state sequence can then be found using the Viterbi algorithm

$$V_t(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}) = \max_{\boldsymbol{x}_{t-2}} p(\boldsymbol{y}_t|x_t) p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}) V_{t-1}(\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2})$$

POMDPs and HMMs compute transitions by finding a sum or maximum over the possible state space. While this is possible in simple n-gram models, it quickly becomes intractable as language models increase in complexity. Sampling methods are necessary for estimating the probability distribution over such models so that high probability sequences can still be tracked without losing the ability to run analysis in real time. Speier et al. [66] applied sequential importance resampling, a standard particle filtering (PF) method to handle more complicated language models. In this system, a probabilistic automaton was used to represent word frequency in English text. Because the model contains over 200,000 states, maximizing over the entire state space in not possible in a real time system. PF methods estimate the distribution over the state space by projecting possible realizations of the system (called particles) through the model over time. Particles are resampled periodically based on the observed signal, so the existing particle distribution closely reflects the posterior probability of a given character. This method was tested online against simpler language models and showed significant improvements in both typing speed and accuracy.

The main concern with using this method is the number of particles to use. Using more particles increases the processing necessary for estimating the distributions. However, a low number of particles could result in undersampling the distributions and missing important possible sequences. Sensitivity analysis in the Speier et al. study showed that performance levelled off in offline analysis when using more than 10,000 particles and it was sufficient for good results in their online study.

3.4 Dynamic stopping

Traditional BCI communication experiments are designed with a static, predetermined number of stimuli presented for each selection, often including 10–15 trials. The system displays these stimuli and then runs the classifier to determine the most probable character and a selection is made. In many cases, enough information could be obtained before all of the stimuli are presented, so a character could be selected earlier, reducing the amount of time required significantly since trial repetition is one of the most time consuming components of the P300 speller system. This time savings can be achieved by running the classification algorithm after each stimulus and testing the results against a target threshold

value. This process is analogous to methods used in other AAC systems where dwell times can be reduced for probable characters [36]. Termed dynamic stopping or dynamic classification, methods for adapting the number of stimuli presented have been presented before [22], but they have started with a uniform probability distribution and therefore took longer to reach the required threshold.

The naïve Bayes method presented in Speier et al. [45] was the first system to incorporate dynamic stopping along with a language model. Several subsequent methods have since incorporated similar methods and it is quickly becoming the standard in P300 classification [48,54,55,60,66]. Dynamic classification was implemented by setting a threshold probability, p_{Thresh}, to determine when a decision should be made. The program flashed characters until either one of the characters' probabilities exceeded the threshold

$$\max_{x_t} p(x_t|\boldsymbol{y}_t, \ldots, \boldsymbol{y}_0) \geq p_{Thresh}$$

or the number of flashes reached the maximum. The classifier then selected the character that satisfied

$$\operatorname{argmax}_{x_t} p(x_t|\boldsymbol{y}_t, \ldots, \boldsymbol{y}_0)$$

When the subject attempts to spell a word that commonly occurs in the language model, the prior probability is heavily weighted towards the target characters. In this case, selections can be made after very few stimuli. Depending on how common the word is and the value of the target threshold, this model can even select characters without presenting any stimuli, effectively employing an autocomplete method. When the user is attempting to spell a word that is less common, the prior probability tends to bias the system away from the correct letters. Usually, the probability of the correct letters is still greater than a uniform distribution over the characters, so the system does not perform more slowly unless another character exceeds the threshold before enough EEG information can be collected to make the correct decision.

When choosing the threshold value to use, system designers need to consider the speed accuracy tradeoff. A lower threshold is easier to reach, meaning a system will make selections faster and typing speed will increase. However, a lower threshold also increases the risk that a non-target character will exceed the threshold spuriously, resulting in an error. In offline studies, the threshold value is typically optimized by finding results for various values and choosing the best for each subject. For online studies, it is generally impractical to optimize the threshold value and an empirical value around 0.95 is used.

3.5 Unsupervised learning

Because neurological signals vary between people and over time, each BCI session is usually preceded by a training session to calibrate the system. Because "locked-in" subjects are prone to fatigue, minimizing this training session could maximize the amount of time

available for using the system for actual communication. Prior work has attempted to create a general classifier that would remove the necessity for this training session [73]. In general, these generic classifiers perform significantly worse because of the variation in the neural signals between people. A better approach is to use a subject's own signals to adapt a classifier while the subject is using the system. This approach is difficult because the target character is unknown prior to classification. A language model can make such an approach possible by adding a bias to the system, allowing it to determine likely target characters and train the classifier.

Kindermans et al. [74] proposed a method using expectation maximization (EM) to train the system during an unsupervised session of free spelling. During use, the subject selects characters for a target word or phrase as in the traditional system. After each selection, the classifier attempts to retrain itself using an iterative process. First, EEG signals are classified based on a random initial system configuration. Then, treating these classifications as true labels, system parameters are optimized as in a training session. Using these parameters, EEG signals are again classified and the process alternates until convergence to a single configuration. This method is dependent on the initial configuration and can result in local optima that do not accurately classify signals. In this study, the problem was addressed by creating multiple initial configurations and running EM separately for each. The result with the highest log likelihood would then be chosen as the true classifier.

While the initial system only used constraints of the system, Kindermans et al. [50] and Speier et al. [58] created similar systems that extended the HMM method from section 3.4 to train a BCI system using unlabeled data. As in the initial Kindermans study, this system uses EM (in this case the Baum-Welch algorithm) to find the optimal system configuration given the observed EEG data and the underlying language model. The expectation step models typing as an HMM and uses the forward-backward algorithm to find the optimal state labels by breaking the computation into two steps: the total probability of all sequences into state and the total probability out of the state given the observed EEG data. These probabilities are computed by assuming that current estimates of model parameters are correct values. The total probability of the state can then be determined by multiplying these two probabilities together for each time point.

Given the observed data and the labels from the expectation step, the maximization step finds the values of model parameters, including the classifier weights which maximize the log likelihood of the data. The optimal values of the weights can be found through a traditional classifier by treating the estimated states as supervised labels. The maximal values other parameters can then be determined by maximizing the log likelihood function. The expectation and maximization steps are then alternated until the labels converge to the optimal configuration. The Speier et al. study showed the capability of the classifier to learn accurate labels given different starting conditions. If the initial conditions are completely unbiased, containing no prior information about neural signals, the classifier is able to learn accurate system parameters for most subjects. For two of the 15 subjects, however, the classifier converged to a local optimum and did not classify any characters correctly. Alternatively, a generic set of parameters learned from a separate population of subjects can be used as a starting point for the algorithm. In this case, the classifier quickly learned

parameters for the system that matched those learned through traditional supervised training for all subjects.

3.6 Error Correction

When assigning prior probabilities to characters, the additional problem arises about what probability to give non-character selections, particularly backspace. When a language model is not used, the backspace option is generally given the same probability as the other elements in the grid [15,75], but it is less obvious how to weight this element in systems that distribute prior probability unevenly based on a language model. The simplest solution is to give these selections a static probability. Orhan et al. [46] proposed an option of giving the backspace selection a static probability of 0.1. They also proposed an alternate possibility of giving the backspace selection a probability based on the previous selection. In this case, a backspace would have the probability

$$p(x_t=\text{'backspace'}|x_{t-1},\ldots,x_0)=1-p(x_{t-1}|x_{t-2},\ldots,x_0)$$

The remaining characters would then be scaled down by a factor of $p(x_{t-1}|x_{t-2},\ldots, x_0)$ to make the total probability equal one. Kindermans et al. [55] proposed a similar method that works in two steps. Initially, the backspace character is given a weight $\hat{p}(x_t = \text{'backspace'}|x_{t-1},\ldots, x_0)$ defined s above and all characters are given probabilities $\hat{p}(x_t|\, x_{t-1},\ldots, x_0)$ based on the language model. All of the probabilities are then reweighted based on the new total probability including the backspace selection:

$$p(x_t|x_{t-1},\ldots,x_0)=\frac{\hat{p}(x_t|x_{t-1},\ldots,x_0)}{1+\hat{p}(x_t=\text{'backspace'}|x_{t-1},\ldots,x_0)}$$

In both of these methods, the values of $p(x_{t-i}|\, x_{t-i-1},\ldots, x_0)$ must be stored for all i so that the system can accurately assign probabilities to the backspace selection due to the possibility that the user will want to perform multiple consecutive backspace operations.

Several of the presented studies have included the possibility of automatically correcting errors through the algorithm. These methods are widely used in text messaging [29] and have been applied previously to other AAC systems [33]. They generally work by having the user continue to use the system after an error without attempting to correct it. After subsequent characters are selected, the system looks back at previous selections and determines whether they agree with the language model. If errors are detected, the system attempts to replace them with more probable selections with little or no additional input from the user.

The system presented by Ryan et al. [43] allows for some error correction when the user selects word completions. When the WordQ software produces proposals for word completion, it does not require an exact match with the incomplete word already typed. Thus, even if a user mistypes a character, it is possible that WordQ would match the string to the correct word and the system would present it as one of the options for completion. In this

case, the correct word would overwrite the mistyped string, thereby correcting the error. In cases where the error prevented the program from matching to the correct completion, the user would be required to use a backspace key to make a manual correction.

Ahi et al. [41] presented a system that would attempt to match an entire input string to a word in the dictionary. While the system was presented as only performing the matching after the entire word was typed, it can be intuitively extended to a system where the matching is performed to substrings of the words after each selection. Because the optimization is performed over the entire substring after each selection, the output characters can possibly change as additional characters are typed. For example, consider a situation where the optimal string after two characters could be found to be “ta,” but after a third character was typed the optimal string was “the.” In this case, the second character would be overwritten from the initial decision, ‘a,’ to a new decision, ‘h,’ that agreed more with the subsequent selection. This method assumes that boundaries between words are correct as each optimization starts at the beginning of the word. Characters in a word are also in flux until the word is completed, meaning that the user does not know if the system will be able to correctly classify characters until the word is completed.

Process model approaches such as those presented by Speier et al. [60,66] optimize over the entire output string and are able to update mistyped characters as more selections are made. Characters are changed if subsequent characters create strings that do not agree with the language model. Because the optimization is over the entire string, word boundaries are not required to be correct. For example, the output string “themc” can be corrected to “the c,” splitting the word to create an optimal output string. In the HMM model, these corrections are based on local character patterns and can still type strings that do not match words in the corpus. The PF method makes corrections using a word-based model, so it will correct text to match words that appeared in the training corpus. In both cases, the system will be unable to make corrections of errors that do not conflict with the language model. The string “then,” for instance, could not be corrected to “than” because both are valid words and use common character patterns. Manual corrections would need to be possible to make such corrections.

While optimization over the entire string allows for more errors to be corrected, it can lead to problems in the case where manual corrections are also allowed. If a user believes the system will correct an error and continues to type rather than correcting it, many additional selections may be needed to delete subsequent characters in order to correct the error. Also, there is an additional cognitive overhead if the user is expected to look at the entire typed string to verify if any changes have been made. In practice, most corrections will involve only the most recent word, so this overhead may not be significantly greater than in the standard system. In addition, word suggestions could help with making changes to characters that the system fails to correct without requiring multiple deletions (see discussion). Usability studies need to be performed in order to determine the optimal strategy for correction in this type of system, identifying situations where the system is able to make the corrections automatically and when users should fix an error manually.

3.7 Evaluation metrics

As new systems are proposed, evaluation metrics play an important role in the direction of research as they are used to evaluate the improvement and relative value of systems' results. Several metrics have been developed for BCI communication output, but most were created with traditional classifiers in mind. They therefore treat all characters as equally probable and do not take interactions between subsequent selections into account. For example, information transfer rate (ITR) essentially defines the amount of information contained in a selection as the difference between selection accuracy and the accuracy from choosing randomly. However, if the prior character is known to be 'Q,' a reader would likely assume that the next character would be 'U' based on their knowledge of the English language. Typing 'U' in this case does not provide much additional information. As a result, metrics generally overestimate the amount of information that is conveyed in a system's output.

Ryan et al. [43] demonstrated an additional problem with ITR in systems with word completion that arises because ITR considers selections rather than output characters. In their study, subjects were able to complete the target sentence in less time using their proposed system, but the control system had a higher average ITR. This was because selections in their system could contain multiple characters when words were completed, but they were still considered a single selection by ITR. This disconnect is similar to concerns in other AAC methods where economy of selections do not always translate into faster typing speeds due to cognitive overhead, which is generally addressed by comparing total times to produce a target output string [35]. Ryan et al. proposed a similar metric, output characters per minute (OCM), which directly addresses typing speed by dividing the total number of output characters by the time required to type them. This metric does not address errors in output as it assumes subjects manually correct all errors. The metric also assumes all target sentences are equally difficult to type, which is not true in general as those that are probable in a language are generally easier in systems that use language models because they have a higher prior probability or are likely to occur as suggestions for autocomplete.

Speier et al. [56] proposed a metric that incorporates some of this information to more accurately assess the true amount of information that is conveyed in a BCI output string. It achieves this by measuring the mutual information between the target string and the actual output string with using a language model to represent the interactions between subsequent selections. Given a perfect language model, this method would give an accurate estimate of the information contained in an output string as it essentially subtracts out all of the language information captured by the model. In practice, however, this method overestimates the amount of information contained in the string because no language model exists that perfectly accounts for all structure in language. The estimates from this method still improve over those that ignore language information, and provide a framework for incorporating more advanced models to further refine the estimates.

4. Discussion

Each of these avenues of integrating language information into BCI communication systems has provided significant improvements over traditional implementations. Several of these methods are currently employed simultaneously in systems: dynamic stopping is frequently

used in systems that use language models in the classifier; unsupervised learning has been used to train a classifier in systems that model language as a process model; and a word completion implementation has allowed for correcting of some errors. However, many of these methods have been developed along parallel tracks and their interactions have not been explored. In many cases, these methods could work in concert to further improve communication speed and accuracy, resulting in a system closer to fulfilling the needs of the target population.

Word completion could be integrated with a system with a language model in its classification. In systems with an n-gram model, the two systems could run independently: the n-gram model would be used for producing prior probabilities for individual characters and a separate dictionary-based model could provide suggested completions. The main task would be determining the prior probability for selecting one of the completions. This probability could either be static or determined based on the relative frequencies of the suggested words.

For systems with a word-based model, word completion and classifier integration can be more closely combined. Word-based models can project the current state through the model to determine the most likely completions of the current text. In systems with a dictionary-based model, this method consists of simply looking up words that start with the typed text. In the particle filtering algorithm, some particles could be projected through the model until they reach a terminal state. A histogram of the paths these particles took would then represent relative probabilities of the possible words that the user could intend to spell.

Word completion could provide an additional benefit when combined with an automatic error correction method. BCI systems that use automatic error correction do so by optimizing over the possible target strings. When computing the most probable string, these methods also find alternative, less probable strings. These strings are currently discarded, but could be presented to the user as options as in the completion method. By doing this, the system is more likely to present the correct string, reducing the number of times that a user will be required to manually correct the error. This method would also reduce the number of times that characters are incorrectly changed, which is possible in cases where a user is trying to spell a word that is similar in spelling, but less common than another word. For example, a user trying to spell “theme” might have the word autocorrected to “there” because of the relative frequency of the words, but both could be presented as options to allow the user to override this mistaken correction.

Word and character based models could be combined in a classifier using a smoothing method. As currently posed, word-based models such as the probabilistic automata and word dictionaries do not allow users to spell OOV words. Because these models are based on general corpora of language, they will often miss uncommon words, particularly proper nouns. The smoothing methods currently used in n-gram models could be implemented here to combine word and character level models. Using these methods, some probability would be given to strings that follow character patterns that are consistent with language, even if they do not make up words that occur in the model. This method would allow systems to

accept OOV words and they could subsequently be added to the word level model allowing it to adapt to the user (see future directions).

The error correction methods used here have the potential to improve performance of unsupervised learning methods. Kindermans et al. [76] recently implemented their unsupervised method in an online system using the AMUSE auditory ERP system. In their study, the classifier was only able to achieve an accuracy comparable to random chance as the system started without any prior information, but as subjects continued to use the system, performance increased significantly. The system then reanalyzed the previous selections as the classifier was retrained using more data, allowing some earlier errors to be corrected. These corrections did not take language information into account, however, as corrections were made based only on a better-trained classifier. Combining this method with a language model would improve these corrections further and could accelerate the unsupervised learning by providing better labels for the EM algorithm.

4.1 Future directions

Current language models used in BCI systems are based on patterns in general corpora of language, and are therefore not necessarily optimal for the specific context of BCI communication. The frequency of words in published text differs from the word frequency in everyday speech, and likely differs even further from locked-in patients who may use the system frequently to communicate things related to their disease (e.g., asking for help adjusting equipment). While general language models can help the user create valid words, it could likely perform better if it can predict the topics of conversation, thereby tailoring the model to the more likely words to be used. Topics of interest will vary between subjects so a language model that reflects one person’s speech patterns will not likely be optimal for any other subject. Ideally, a unique language model would be built for each individual user based on the text that they generate using the system. Such a system could start with a more general model, and gradually tailor it to the user over time as has previously been done in other AAC systems [33]. The likely words and topics will also vary between time and context, so an optimal language model should adjust to outside information such as the time of day, the state of other devices in the room, or the identities of the people with whom the subject is speaking. This adaptation to outside context has been shown in other BCI systems [75], but has not yet been incorporated into language model-based communication.

Existing BCI systems limit their usage of language models to single words. Significant information available from the relationships between words such as sentence structure and parts of speech is ignored. Common phrases also make up a large part of everyday communication, which could significantly accelerate typing speed if included in a language model. While a complete model of language would be impractical for this application, simplified models can be used to include some of the information, which could yield significant improvements in system performance. The n-gram models currently used for characters in BCI can be extended to words, which would provide a starting point for including this information.

Prior information from the language model can also be used to change the stimuli presented to the user. The number or frequency of character stimuli can be augmented based on the

probability of the character possibly reducing the number of necessary stimuli to make a selection and thereby increasing system speed. The RSVP speller may be particularly suitable for this approach as low probability characters can be removed from the set of presented characters [46,67]. The P300 speller could be similarly use this approach by applying pseudo-random flashing paradigms such as those presented by Townsend et al. [15] and Jin et al. [16] to a reduced set of potential characters determined by the language model.

While the target population for this system is “locked-in” patients, most of the systems presented here tested exclusively on healthy volunteers with only three including ALS patients [59,61,63]. The translation of results from healthy subjects to the target population is a general concern for the field as ALS patients can have additional difficulties such as the loss of eye gaze control. The systems here that involve changing graphical interfaces such as the word completion methods could therefore be more difficult for these subjects to control. Systems that include automatic error correction could be problematic as well, as patients would have the additional task of deciding which errors the system would be able to fix on its own and which require manual correction. Some of these issues might possibly self-correct over time as the patient becomes more accustomed to suing the system. Testing in the target population will be required in order to determine the true improvement seen by these patients when using a system with language models. Additionally, longitudinal studies can demonstrate the long-term effectiveness of the systems as the patient familiarizes with the system. Longitudinal studies could also allow for tailoring language models to the individual subject to further improve performance.

5. Conclusion

While the exploration of integration of language information into BCI communication only recently, significant advances have already been demonstrated. These improvements come from various aspects of the system, ranging from changes in the user interface to modifying the signal classification algorithms. Because of the recent nature of this concept, many groups are engaged in parallel tracks to try to utilize this information. Many of these systems could potentially work in conjunction with one another or could benefit from building off of previous works.

Acknowledgments

This work was supported by the National Institute of Biomedical Imaging and Bioengineering Award Number K23EB014326 (NP) and the UCLA Scholars in Translational Medicine Program (NP).

References

1. Ball L, Nordness A, Fager S, Kersch K, Pattee G, Beukelman D. Eye-gaze access of AAC technology for persons with amyotrophic lateral sclerosis. J Med Speech Lang Pathol. 2010; 18:11–23.
2. Huggins JE, Wren PA, Gruis KL. What would brain-computer interface users want? Opinions and priorities of potential users with amyotrophic lateral sclerosis. Amyotroph Lateral Scler. 2011; 12:318–24. [PubMed: 21534845]
3. Wolpaw JR, Birbaumer N, McFarland DJ, Pfurtscheller G, Vaughan TM. Brain-computer interfaces for communication and control. Clin Neurophysiol. 2002; 133:767–91.

4. Farwell LA, Donchin E. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. Electroencephalogr Clin Neurophysiol. 1988; 70:510–23. [PubMed: 2461285]
5. Acqualagna L, Blankertz B. Gaze-independent BCI-spelling using rapid serial visual presentation (RSVP). Clin Neurophysiol. 2013; 124:901–8. [PubMed: 23466266]
6. Schreuder M, Rost T, Tangermann M. Listen, you are writing! Speeding up online spelling with a dynamic auditory BCI. Front Neurosci. 2011; 5:112. [PubMed: 22016719]
7. Blankertz B, Curio G, Muller K-R. Classifying single trial EEG: Towards brain computer interfacing. Adv Neural Inf Process Syst. 2002; 1:157–64.
8. D'albis T, Blatt R, Tedesco R, Sbattella L, Matteucci M. A predictive speller controlled by a brain-computer interface based on motor imagery. ACM Trans Comput Interact. 2012; 19:20.
9. Blankertz B, Dornhege G, Krauledat M, Schröder M, Williamson J, Murray-Smith R, Müller K-R. The Berlin Brain-Computer Interface presents the novel mental typewriter Hex-o-Spell. Proc 3rd Int Brain-Computer Interface Work Train Course. 2006:108–9.
10. Cecotti H. A self-paced and calibration-less SSVEP-based brain–computer interface speller. Neural Syst Rehabil Eng IEEE Trans. 2010; 18:127–33.
11. Volosyak, I., Moor, A., Gräser, A. Advances in Computational Intelligence. Springer; 2011. A dictionary-driven SSVEP speller with a modified graphical user interface; p. 353-61.
12. Xu M, Qi H, Wan B, Yin T, Liu Z, Ming D. A hybrid BCI speller paradigm combining P300 potential and the SSVEP blocking feature. J Neural Eng. 2013; 10:26001.
13. Yin E, Zhou Z, Jiang J, Chen F, Liu Y, Hu D. A novel hybrid BCI speller based on the incorporation of SSVEP into the P300 paradigm. J Neural Eng. 2013; 10:26012.
14. Sellers EWE, Krusienski DJDJ, McFarland DJ, Vaughan TM, Wolpaw JR. A P300 event-related potential brain-computer interface (BCI): The effects of matrix size and inter stimulus interval on performance. Biol Psychol. 2006; 73:242–52. [PubMed: 16860920]
15. Townsend G, LaPallo BK, Boulay CB, Krusienski DJ, Frye GE, Hauser CK, Schwartz NE, Vaughan TM, Wolpaw JR, Sellers EW. A novel P300-based brain-computer interface stimulus presentation paradigm: Moving beyond rows and columns. Clin Neurophysiol. 2010; 121:1109–20. [PubMed: 20347387]
16. Jin J, Horki P, Brunner C, Wang X, Neuper C, Pfurtscheller G. A new P300 stimulus presentation pattern for EEG-based spelling systems. Biomed Tech. 2010; 55:203–10.
17. McFarland DJ, Sarnacki WA, Townsend G, Vaughan T, Wolpaw JR. The P300-based brain-computer interface (BCI): Effects of stimulus rate. Clin Neurophysiol. 2011; 122:731–7. [PubMed: 21067970]
18. Lu J, Speier W, Hu X, Pouratian N. The effects of stimulus timing features on P300 speller performance. Clin Neurophysiol. 2012; 124:306–14. [PubMed: 22939456]
19. Kaufmann T, Schulz SM, Grünzinger C, Kübler A. Flashing characters with famous faces improves ERP-based brain–computer interface performance. J Neural Eng. 2011; 8:56016.
20. Kaper M, Meinicke P, Grossekathoefer U, Lingner T, Ritter H. BCI Competition 2003 - Data Set IIb: Support Vector Machines for the P300 Speller Paradigm. IEEE Trans Biomed Eng. 2004; 50:1073–6.
21. Xu N, Gao X, Hong B, Miao X, Gao S, Yang F. BCI Competition 2003 - Data Set IIb: Enhancing P300 Wave Detection Using ICA-Based Subspace Projections for BCI Applications. IEEE Trans Biomed Eng. 2004; 51:1067–72. [PubMed: 15188880]
22. Serby H, Yom-Tov E, Inbar GF. An improved P300-based brain-computer interface. IEEE Trans Neural Syst Rehabil Eng. 2005; 13:89–98. [PubMed: 15813410]
23. Krusienski DJ, Sellers EW, Cabestaing F, Bayoudh S, McFarland DJ, Vaughan TM, Wolpaw JR. A comparison of classification techniques for the P300 Speller. J Neural Eng. 2006; 3:299–305. [PubMed: 17124334]
24. Brunner P, Ritaccio AL, Emrich JF, Bischof H, Schalk G. Rapid Communication with a "P300" Matrix Speller Using Electrocorticographic Signals (ECoG). Front Neurosci. 2011; 5:5. [PubMed: 21369351]
25. Krusienski DJ, Shih JJ. Control of a visual keyboard using an electrocorticographic brain–computer interface. Neurorehabil Neural Repair. 2011; 25:323–31. [PubMed: 20921326]

26. Donchin E, Spencer KM, Wijesinghe R. The mental prosthesis: assessing the speed of a P300-based brain-computer interface. Rehabil Eng IEEE Trans. 2000; 8:174–9.
27. Jelinek, F. Statistical methods for speech recognition. MIT press; 1997.
28. Hart-Davis, G. Office 2010 Made Simple. Springer; 2011. Entering and Editing Text in Your Documents; p. 165-97.
29. Fowler, A., Partridge, K., Chelba, C., Bi, X., Ouyang, T., Zhai, S. Effects of Language Modeling and its Personalization on Touchscreen Typing Performance. Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems; ACM; 2015. p. 649-58.
30. Dunlop MD, Crossan A. Predictive text entry methods for mobile phones. Pers Technol. 2000; 4:134–43.
31. Bird, S. Proceedings of the COLING/ACL on Interactive presentation sessions. Association for Computational Linguistics; 2006. NLTK: the natural language toolkit; p. 69-72.
32. Francis, WN., Kucera, H. Brown Corpus Manual. Providence, RI: Dept of Linguistics, Brown University; 1979.
33. Darragh JJ, Witten IH, James ML. The reactive keyboard: A predictive typing aid. Computer (Long Beach Calif). 1990; 23:41–9.
34. Foulds R, Soede M, van Balkom H. Statistical disambiguation of multi-character keys applied to reduce motor requirements for augmentative and alternative communication. Augment Altern Commun. 1987; 3:192–5.
35. Lesher G, Moulton B, Higginbotham DJ. Techniques for augmenting scanning communication. Augment Altern Commun. 1998; 14:81–101.
36. Nantais T, Shein F, Treviranus J. A predictive selection technique for single-digit typing with a visual keyboard. Rehabil Eng IEEE Trans. 1994; 2:130–6.
37. Roark B, Beckley R, Gibbons C, Fried-Oken M. Huffman scanning: using language models within fixed-grid keyboard emulation. Comput Speech Lang. 2013; 27:1212–34.
38. Lesher GW, Moulton BJ, Higginbotham DJ. Optimal character arrangements for ambiguous keyboards. Rehabil Eng IEEE Trans. 1998; 6:415–23.
39. Mora-Cortes A, Manyakov NV, Chumerin N, Van Hulle MM. Language Model Applications to Spelling with Brain-Computer Interfaces. Sensors. 2014; 14:5967–93. [PubMed: 24675760]
40. Volosyak, I., Cecotti, H., Valbuena, D., Gräser, A. Evaluation of the Bremen SSVEP based BCI in real world conditions. Rehabilitation Robotics, 2009. ICORR 2009. IEEE International Conference on; IEEE; 2009. p. 322-31.
41. Ahi ST, Kambara H, Koike Y. A dictionary-driven P300 speller with a modified interface. IEEE Trans Neural Syst Rehabil Eng. 2011; 19:6–14. [PubMed: 20457551]
42. Höhne J, Schreuder M, Blankertz B, Tangermann M. A novel 9-class auditory ERP paradigm driving a predictive text entry system. Front Neurosci. 2011; 5:99. [PubMed: 21909321]
43. Ryan DB, Frye GE, Townsend G, Berry DR, Mesa-GS, Gates NA, Sellers EW. Predictive spelling with a P300-based brain–computer interface: increasing the rate of communication. Intl J Human–Computer Interact. 2010; 27:69–84.
44. Lee, S., Lim, H-S. Future Information Technology. Springer; 2011. Predicting text entry for brain-computer interface; p. 309-12.
45. Speier W, Arnold C, Lu J, Taira RK, Pouratian N. Natural language processing with dynamic classification improves P300 speller accuracy and bit rate. J Neural Eng. 2011; 9:016004. [PubMed: 22156110]
46. Orhan, U., Hild, KE., Erdogmus, D., Roark, B., Oken, B., Fried-Oken, M. RSVP keyboard: An EEG based typing interface. Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on; IEEE; 2012. p. 645-8.
47. Kaufmann T, Völker S, Gunesch L, Kübler A. Spelling is just a click away–a user-centered brain–computer interface including auto-calibration and predictive text entry. Front Neurosci. 2012; 6:72. [PubMed: 22833713]
48. Park J, Kim K-E. A POMDP approach to optimizing P300 speller BCI paradigm. Neural Syst Rehabil Eng IEEE Trans. 2012; 20:584–94.

49. Orhan, U., Erdogmus, D., Roark, B., Oken, B., Purwar, S., Hild, KE., Fowler, A., Fried-Oken, M. Improved accuracy using recursive bayesian estimation based language model fusion in ERP-based BCI typing systems. Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE; IEEE; 2012. p. 2497-500.
50. Kindermans P-J, Verschore H, Verstraeten D, Schrauwen B. A P300 BCI for the masses: Prior information enables instant unsupervised spelling. Advances in Neural Information Processing Systems. 2012:710–8.
51. Ulas, C., Cetin, M. The first Brain-Computer Interface utilizing a Turkish language model. Signal Processing and Communications Applications Conference (SIU), 2013 21st; IEEE; 2013. p. 1-4.
52. Ulas, C., Çetin, M. Incorporation of a language model into a brain computer interface based speller through HMMs. Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on; IEEE; 2013. p. 1138-42.
53. Speier W, Fried I, Pouratian N. Improved P300 speller performance using electrocorticography, spectral features, and natural language processing. Clin Neurophysiol. 2013; 124:1321–8. [PubMed: 23465430]
54. Samizo, E., Yoshikawa, T., Furuhashi, T. Foundations of Augmented Cognition. Springer; 2013. A Study on Application of RB-ARQ Considering Probability of Occurrence and Transition Probability for P300 Speller; p. 727-33.
55. Kindermans P-J, Verschore H, Schrauwen B. A Unified Probabilistic Approach to Improve Spelling in an Event-Related Potential-Based Brain–Computer Interface. Biomed Eng IEEE Trans. 2013; 60:2696–705.
56. Speier W, Arnold C, Pouratian N. Evaluating True BCI Communication Rate through Mutual Information and Language Models. PLoS One. 2013; 8:e78432. [PubMed: 24167623]
57. Casagrande, A., Jarmolowska, J., Turconi, M., Fabris, F., Battaglini, PP. Brain and Health Informatics. Springer; 2013. PolyMorph: A P300 Polymorphic Speller; p. 297-306.
58. Speier, W., Knall, J., Pouratian, N. Unsupervised training of brain-computer interface systems using expectation maximization. Neural Engineering (NER), 2013 6th International IEEE/EMBS Conference on; IEEE; 2013. p. 707-10.
59. Orhan U, Erdogmus D, Roark B, Oken B, Fried-Oken M. Offline analysis of context contribution to ERP-based typing BCI performance. J Neural Eng. 2013; 10:66003.
60. Speier W, Arnold C, Lu J, Deshpande A, Pouratian N. Integrating language information with a hidden markov model to improve communication rate in the P300 speller. IEEE Trans Neural Syst Rehabil Eng. 2014; 22:678–84. [PubMed: 24760927]
61. Oken BS, Orhan U, Roark B, Erdogmus D, Fowler A, Mooney A, Peters B, Miller M, Fried-Oken MB. Brain–Computer Interface With Language Model–Electroencephalography Fusion for Locked-In Syndrome. Neurorehabil Neural Repair. 2014; 28:387–94. [PubMed: 24370570]
62. Mainsah BO, Colwell KA, Collins LM, Throckmorton CS. Utilizing a language model to improve online dynamic data collection in P300 spellers. Neural Syst Rehabil Eng IEEE Trans. 2014; 22:837–46.
63. Mainsah BO, Collins LM, Colwell KA, Sellers EW, Ryan DB, Caves K, Throckmorton CS. Increasing BCI communication rates with dynamic stopping towards more practical use: an ALS study. J Neural Eng. 2015; 12:16013.
64. Saa JFD, de Pesters A, McFarland D, Çetin M. Word-level language modeling for P300 spellers based on discriminative graphical models. J Neural Eng. 2015; 12:26007.
65. Moghadamfalahi M, Orhan U, Akcakaya M, Nezamfar H, Fried-Oken M, Erdogmus D. Language-Model Assisted Brain Computer Interface for Typing: A Comparison of Matrix and Rapid Serial Visual Presentation. Neural Syst Rehabil Eng IEEE Trans. 2015; 23:910–20.
66. Speier W, Arnold CW, Deshpande A, Knall J, Pouratian N. Incorporating advanced language models into the P300 speller using particle filtering. J Neural Eng. 2015; 12:046018. [PubMed: 26061188]
67. Moghadamfalahi, M., Gonzalez-Navarro, P., Akcakaya, M., Orhan, U., Erdogmus, D. Foundations of Augmented Cognition. Springer; 2015. The Effect of Limiting Trial Count in Context Aware BCIs: A Case Study with Language Model Assisted Spelling; p. 281-92.
68. Draper, NR., Smith, H. Applied regression analysis. John Wiley & Sons; 2014.

69. Witten IH, Bell T. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. Inf Theory, IEEE Trans. 1991; 37:1085–94.

70. Kneser, R., Ney, H. Improved backing-off for m-gram language modeling. Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on; IEEE; 1995. p. 181-4.

71. Kim K-E. Exploiting Symmetries in POMDPs for Point-Based Algorithms. AAAI. 2008:1043–8.

72. Doshi, F., Roy, N. The permutable POMDP: fast solutions to POMDPs for preference elicitation. Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems; International Foundation for Autonomous Agents and Multiagent Systems; 2008. p. 493-500.

73. Kaper, M., Ritter, H. Generalizing to new subjects in brain-computer interfacing. Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE; IEEE; 2004. p. 4363-6.

74. Kindermans P-J, Verstraeten D, Schrauwen B. A bayesian model for exploiting application constraints to enable unsupervised training of a P300-based BCI. PLoS One. 2012; 7:e33758. [PubMed: 22496763]

75. Faller, J., Torrellas, S., Miralles, F., Holzner, C., Kapeller, C., Guger, C., Bund, J., Muller-Putz, GR., Scherer, R. Prototype of an auto-calibrating, context-aware, hybrid brain-computer interface. Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE; IEEE; 2012. p. 1827-30.

76. Kindermans P-J, Schreuder M, Schrauwen B, Müller K-R, Tangermann M. True Zero-Training Brain-Computer Interfacing–An Online Study. PLoS One. 2014; 9:e102504. [PubMed: 25068464]

Table 1

List of BCI spelling systems included in this review. The results varied between studies based on the method they were compared to and the metric used for analysis. Missing values indicate either that the proposed method was not compared to a method without a language model or that numerical data was not provided. The metrics used were: output characters per minute (OCM), information transfer rate (ITR), theoretical bit rate (TBR), accuracy (ACC), gUtility, and symbols per minute (SPM).

Paper	System	Language Model	Algorithm	Subjects	% Improvement (metric)
Ryan [43]	P300 Speller	Dictionary	Word completion	24 online	40.4 (OCM)
Ahi [41]	P300 Speller	Dictionary	Rank sum	14 offline	241.7 (TBR)
Volosyak [11]	SSVEP	Dictionary	Word completion	7 online	9.1 (ITR)
Lee [44]	Motor Imagery	Dictionary	Word completion	NA	NA
Speier [45]	P300 Speller	N-gram	Naïve Bayes	6 offline	50.2 (ITR)
Orhan [46]	RSVP	N-gram	Naïve Bayes	3 online	NA
Kaufmann [47]	P300 Speller	Dictionary	Word completion	19 online	71.7 (TBR)
Park [48]	P300 Speller	N-gram	Partially observable Markov decision process	10 online	61.8 (ITR)
Orhan [49]	RSVP	N-gram	Hidden Markov Model	2 offline	6 (ACC)
D'albis [8]	Motor Imagery	N-gram	Word completion	3 online	NA
Kindermans [50]	P300 Speller	N-gram	Expectation maximization	22 offline	19.2 (ACC)
Ulas [51]	P300 Speller	N-gram	Hidden Markov model	6 offline	55.3 (ITR)
Ulas [52]	P300 Speller	N-gram	Hidden Markov model	6 offline	55.3 (ITR)
Speier [53]	P300 Speller	N-gram	Naïve Bayes	2 offline	29.1 (ITR)
Samizo [54]	P300 Speller	N-gram	Reliability-based automatic repeat request	3 online	42.3 (gUtility)
Kindermans [55]	P300 Speller	N-gram	Naïve Bayes	22 offline	61.4 (SPM)
Speier [56]	P300 Speller	N-gram	Mutual Information	NA	NA
Casagrande [57]	P300 Speller	Dictionary	Word completion	10 online	NA
Speier [58]	P300 Speller	N-gram	Expectation maximization	15 offline	NA
Orhan [59]	RSVP	N-gram	Naïve Bayes	3 online	NA
Speier[60]	P300 Speller	N-gram	Hidden Markov model	5 online	89.3 (ITR)
Oken [61]	RSVP	N-gram	Naïve Bayes	18 online	NA
Mainsah [62]	P300 Speller	N-gram	Naïve Bayes	17 online	12.2 (ITR)
Mainsah [63]	P300 Speller	N-gram	Naïve Bayes	10 online	254.6 (ITR)

Paper	System	Language Model	Algorithm	Subjects	% Improvement (metric)
Saa[64]	P300 Speller	Dictionary	Word Level Model	7 offline	10.3 (ACC)
Moghadamfalahi [65]	P300 Speller, RSVP	N-gram	Naïve Bayes	12 online	NA
Speier [66]	P300 Speller	Probabilistic automata	Particle Filtering	15 online	21.6 (ITR)
Moghadamfalahi [67]	RSVP	N-gram	Naïve Bayes	11 offline	NA