

Short Abstract

Computational language modeling and its contribution to speed of communication has not received adequate attention by the AAC research community. Language modeling is important when considering binary response typing interfaces, such as AAC spelling grids and alternative access methods. At OHSU, we are designing a communication system based on comprehensive language models and a brain-computer interface (BCI) for people who are functionally locked in. We have collected simulation data on two binary coding strategies: Optimal Huffman coding, which assigns binary codes with the minimum expected bits (keystrokes) per character; and linear binary coding, in which every symbol's code consists of zeros followed by a single one. Interest in the linear coding strategy is driven by our interest in a Rapid Serial Visual Presentation (RSVP) keyboard, where symbols are presented one at a time, thus precluding optimal Huffman coding. Simulations were run on the models described above without error, as well as with random errors. Data suggest that, as the probability of error increases, the keystrokes required with linear binary coding strategies approaches that of the optimal Huffman code. This information informs design of a binary choice AAC system we are developing for an RSVP keyboard interface.

Extended Abstract

Computational language models hold great potential for making AAC more efficient. A rich language model can theoretically minimize access time to the most likely targets during message construction as well as identify efficiencies sensitive to a particular access strategy or user context. Language modeling is particularly relevant for systems relying on a binary interface. For example, a person who physically activates a switch during row/column scanning, or whose ERP signals are interpreted by a brain-computer interface (BCI), is essentially providing a binary yes/no response to each potential selection.

Row/column scanning is not the only means by which a spelling grid can be used as a binary response typing interface. Rather than highlighting full rows or full columns, subsets of letters that do not necessarily fall in contiguous rows or columns could be highlighted. In this way the letter grid itself remains static while the most probable set of letters following each choice is highlighted. Which symbols are highlighted defines a binary code: 1 means highlighted; 0 means not-highlighted. Huffman coding [1] provides the minimal expected bits (or keystrokes) per character, and is hence optimal in this respect. This strategy does require, however, that users reorient their attention to the grid as the combination of highlighted symbols changes with each switch activation. Also, since more than one character can be highlighted at a time, this coding strategy requires multiple symbols to be displayed simultaneously.

Another method for producing a binary code, which we will call a linear code, provides more flexibility in the kind of interface that it allows. Given a list of symbols in rank probability order, a linear code assigns the symbol at rank k with the binary code of $k-1$ zeros followed by a 1. In other words, every position in the binary code is a decision between exactly one symbol and all of the others, with decisions on the most likely symbols occurring earlier. Using this coding strategy, a spelling grid could highlight exactly one letter at a time in rank order. Alternately, symbols could be presented individually in isolation without a grid. At OHSU, we are currently investigating the feasibility of a single symbol presentation in our Rapid Serial Visual Presentation (RSVP) keyboard as part of our brain-computer interface project.

The simplicity of an interface that presents a single letter at a time may reduce user fatigue, and even make typing feasible for users that cannot maintain focus on a spelling grid. Recent attempts to use a P300 Speller as a typing interface for individuals with ALS found that the many items in the grid caused problems for these patients, because of difficulty orienting attention to specific locations in the grid [2]. Additionally, single symbol auditory presentation would be possible, for visually impaired individuals, something that is not straightforwardly feasible with the sets of symbols that must be presented when using Huffman codes. Hence linear coding provides much more interface flexibility than Huffman or row/column coding. The key question is: how much worse than optimal Huffman coding is the linear coding in terms of bits (keystrokes/switch activations) per character? We conducted simulation trials to answer this question.

Empirical Trials

We prepared two corpora for evaluation (training and simulation): (1) newswire text from the New York Times (NYT) portion of the English Gigaword corpus (LDC2007T07); and (2) email text from the Enron email dataset (<http://www-2.cs.cmu.edu/~enron>). Both corpora were preprocessed for the current evaluation, to extract text that was actually typed. For the Enron Email Dataset, data from the "bodies" table of the SQL database made available by A. Fiore and J. Heer (<http://bailando.sims.berkeley.edu/enron/enron.sql.gz>) who performed an extensive amount of duplicate removal and name normalization. For both corpora, an iterative procedure was followed to reduce duplication in the corpus, whereby duplication was discovered and extracted. The final training sets were of size 42M (NYT) and 35M (Enron) characters; test sets were of size 237K (NYT) and 213K (Enron) characters.

For a character-based n -gram model of order k , the previous k typed symbols are used to assign probabilities to all possible following symbols. We used a version of Witten-Bell smoothing [3] with an optimized hyperparameter K , following [4], where readers are referred for further details. Symbols are ranked in descending probability order, and the rank and probabilities are used to build the binary codes. Our symbol set was of size 100, including the 96 ASCII characters above 31 (including the DEL symbol) plus tab, newline, end-of-file, and an "out-of-vocabulary" symbol. The language models were smoothed so that all symbols received a non-zero probability, even if unobserved in the training set.

Random error production was simulated. At each possible keystroke, the current bit of the target character is either 0 or 1. For some parameter p , we choose the correct bit with probability $1-p$ (using a random number generator); and we choose the incorrect bit with probability p . If a non-target (incorrect) symbol is typed, the DEL (delete) symbol must then be chosen to correct the error, after which the typing interface returns to the previous position. The probability of the DEL symbol is exactly the probability of error p .

The first set of results reports bits per character typed for the NYT and Enron test sets under various conditions, with no simulated errors. Our baseline row/column scanning system achieved 4.7 and 4.8 bits per character for the NYT and Enron test sets respectively.

Corpus	Code	Markov order of n-gram model							
		2	3	4	6	8	10	15	20
NYT	Huffman	7.9	5.0	3.8	2.7	2.3	2.2	2.1	2.1
	Linear	22.2	12.2	7.8	4.1	3.3	3.1	2.9	2.9
Enron	Huffman	7.0	5.0	3.7	2.6	2.3	2.2	2.1	2.1
	Linear	20.3	13.7	8.1	4.1	3.4	3.2	3.1	3.0

Table 1: Bits (keystrokes) per character typed with no simulated errors with various language models

As the Markov order of the n-gram model grows, the quality of the language model increases, resulting in fewer bits (keystrokes) per character. There is just under a 1 bit per character difference for optimal Huffman versus linear coding when the language models are high quality.

Next we look at introducing random error of a particular probability into the simulation, computing bits-per-character for a 15-gram model. Baseline row/column scanning degrades to over 6 bits (keystrokes) per character even with the relatively low error rates of 2%, en route to tens of keystrokes per character at higher error rates.

Corpus	Code	Percentage probability of keystroke error							
		0	2	5	10	15	20	25	30
NYT	Huffman	2.1	2.3	2.5	2.9	3.4	3.9	4.6	5.3
	Linear	2.9	3.2	3.4	3.8	4.2	4.6	5.1	5.6
Enron	Huffman	2.1	2.4	2.6	3.0	3.5	4.0	4.7	5.5
	Linear	3.1	3.5	3.8	4.2	4.6	5.0	5.5	5.9

Table 2: Bits (keystrokes) per character for 15-gram language model with simulated errors

Interestingly, the absolute keystroke differences between optimal Huffman coding and linear coding narrow for both corpora as the error rate increases, to under half a bit/keystroke. This is important because, for our prospective work with a brain-computer interface that relies on single trial ERP detection, there will be some level of classification error. For example, the results in [5] achieved ERP classification accuracies in the 73-97% range for non-ALS subjects, and in the 61-80% range for ALS

subjects. Together with random user errors, an overall error rate of 30% is not unexpected.

Preliminary trials at OHSU indicate that using a Huffman code on a spelling grid yields relatively high error rates versus a linear code with an RSVP interface, but further investigation is required. If the differences prove large enough, however, the RSVP interface may be as efficient as or more efficient than Huffman coding with a grid.

In summary, we have presented language modeling and coding methods for binary response typing interfaces, along with empirical results that illustrate the potential viability of linear scan RSVP user interfaces as an alternative to more traditional AAC spelling grids.

References:

- [1] D.A. Huffman. 1952. A method for the construction of minimum redundancy codes. In *Proceedings of the IRE*, volume 40(9), pages 1098-1101.
- [2] E.W. Sellers, G. Schalk, and E. Donchin. 2003. The p300 as a typing tool: tests of brain-computer interface with an ALS patient. *Psychophysiology*, 40:77.
- [3] B. Carpenter. 2005. Scaling high-order character language models to gigabytes. In *Proceedings of the ACL Workshop on Software*, pages 86-99.
- [4] I.H. Witten and T.C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085-1094.
- [5] E.W. Sellers and E. Donchin. 2006. A p300-based brain-computer interface: initial tests by ALS patients. *Clinical Neurophysiology*, 117:538-548.