SLPAT 2016 Workshop on Speech and Language Processing for Assistive Technologies
13 September 2016, San Francisco, USA

Combining word prediction and r-ary Huffman coding for text entry

Seung Wook Kim[1], *Frank Rudzicz*[2,1]

[1]Department of Computer Science, University of Toronto;
[2]Toronto Rehabilitation Institute, University Health Network;
seungwook.kim@mail.utoronto.ca, frank@cs.toronto.edu

Abstract

Two approaches to reducing effort in switch-based text entry for augmentative and alternative communication devices are word prediction and efficient coding schemes, such as Huffman. However, character distributions that inform the latter have never accounted for the use of the former. In this paper, we provide the first combination of Huffman codes and word prediction, using both trigram and long short term memory (LSTM) language models. Results show a significant effect of the length of word prediction lists, and up to 41.46% switch-stroke savings using a trigram model.

1. Introduction

There are approximately 270,000 people in North America with spinal cord injuries, approximately 47% of whom develop quadriplegia (also called tetraplegia), which is a partial or total paralysis of the limbs and torso [1, 2]. In addition to these traumatic losses of motor function, millions more are affected by neuromotor disorders, collectively called *dysarthria*, that impair the production of speech secondary to various congenital or traumatic conditions, including cerebral palsy, stroke, Parkinson's disease, and multiple sclerosis.

Individuals with communication disorders often use augmentative and alternative communication (AAC) technologies to express themselves, specifically to synthesize speech from typed text or symbol sequences. These systems can employ a wide range of inputs, including hand gestures, typing, or eye and head movements [3] that are designed to minimize muscle movement, given global motor deficits. These modalities can interact with either screen-based or screen-free paradigms in which input is transduced to a cursor position [4]. Typically, symbols are selected when the user either dwells on them or performs a specific action, such as blinking or activating a switch, as shown in Figure 1.

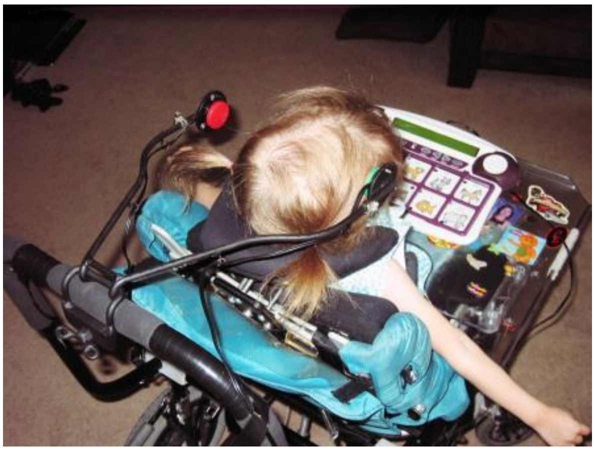

Figure 1: Example of a two-button head-switch mounted on a wheelchair. Image used by permission of the Tetra Society of North America.

Through a series of local interviews with AAC users, we have found that screen-based approaches can interfere with certain social aspects of conversation. In particular, users emphasized that screens often form a barrier to eye contact between conversants and that, given a shared screen, conversation partners will often "read-as-they-go," and interrupt the speaker, resulting in editorialization. For these reasons, we are optimizing a screen-free system using eye and head movements.

In this paper, we assign codes to alphanumeric English characters using r-ary Huffman coding, as is typical. However, since AAC devices are also likely to benefit from *word prediction*, the distribution of those characters in training data will not necessarily resemble actual use. For example, although the letter '*e*' is quite frequent, if it tends to occur to-

10.21437/SLPAT.2016-17

wards the ends of words, it is less likely to be typed if those words can be accurately predicted from context. We therefore provide the first work that adjusts Huffman codes given distributions subsequent to word prediction, using both trigram and long short term memory (LSTM) language models. The result is up to 41.46% switch-stroke savings using a trigram model.

1.1. Previous work

Previous AAC systems for gestural text entry have sought to minimize selection complexity by limiting the number of possible inputs. The H4, EdgeWrite, and 'Left, Up, Right, Down' Writer systems all relied on codes that used four discrete inputs [5, 6, 7], typically target regions placed at the edges or corners of a screen. The MDITIM (Minimal Device Independent Text Input Method) system uses a similar convention, with four inputs dedicated to the coding of characters and one input reserved as a modifier, for example, to achieve capitalization [8]. In order to further simplify the input process, the H4 and MDITIM systems, unlike EdgeWrite, have used prefix-free codes to avoid the need for a unique termination event, such as a finger-up or blink, to designate the end of each character [5, 8].

Expert users, with about 2.5 hours of experience using the EyeS eye gesture communication system, had text communication rates of 6.8 words per minute (wpm), as compared with typical speech rates of 130-200 wpm, and typing rates of 30-40 wpm for unimpaired typists [9, 10, 11]. Similarly, users with 5.0 hours of practice using the MDITIM had an average text entry speed of less than 10 wpm [8]. One approach to improving communication rate is to reduce the number of inputs needed to enter each character. The H4 system uses Huffman codes to form a prefix-free code, and resulted in an average text entry rate of 20 wpm after 6.5 hours of experience [12]. Roark *et al.* [13] also uses Huffman coding to select the symbols to highlight during character scanning process, minimizing the expected bits per symbol.

Word prediction is another strategy for optimizing text entry. Trnka *et al.* [14] showed that word prediction, using a recency-of-use model, increased communication rates in an AAC-like onscreen keyboard system and that more advanced methods based on statistical language modelling proved more effective, increasing communication rates by 56.8%. The number of options presented is an important factor – longer lists increases the chances that the desired word will be found, but this also increases the visual or auditory scan time to evaluate the list. Mackenzie [15] suggested that a list size of $N = 5$ is optimal.

2. Data

We use three data sets:

Wall Street Journal (WSJ) Selected 2,499 stories from a three-year WSJ collection consisting of 1,098,785 word tokens (43,283 word forms). This dataset contains the most formal language of the three databases.

Essays A collection of essays, poems, and short stories from Grade 11 students in high-schools across Ontario recorded as part of their regular curricula. This consists of 5,831,405 word tokens (114,113 word forms) across 5,448 documents. The formality of the writing is appropriate for teenage writers.

NUS Short Message Service (SMS) Corpus [16] A collection of 55,835 SMS messages collected by the NLP group of the National University of Singapore. This dataset consists of 548,210 word tokens (33,694 word forms) and represents the least formal language of the three databases here.

For our purposes, alphabets are reduced to lowercase alphanumerics and 'space'. All capital letters are changed to lowercase equivalents and extraneous characters are deleted. Additional datasets were considered, including some artificial simulations of AAC text, but these were either too small for our purposes, or provided no additional benefit to the data sets described above.

3. Methods

We train language models to build our word prediction system that produces the list of N most probable next words given the history of characters typed. Each alphanumeric English characters and the indices of the prediction list is assigned a code using r-ary Huffman coding based on the information we get from the word prediction system, assuming that the user types with an AAC system that has r switches. Input savings by using the word prediction system is calculated for varying N and r values.

We describe the two language models used in word prediction in section 3.1, and our implementation of r-ary Huffman coding in section 3.2.

3.1. Language models

We train two types of language model for each data set. Each produces an N-best prediction list for each word w_i given the previous words $w_{i-n+1}, ..., w_{i-1}$. That is, we choose the top N probabilities from the list L such that

$$L = \{P(w^j|w_{i-n+1}...w_{i-1}) : 0 \leq j < |V|\} \qquad (1)$$

where $|V|$ is the size of the vocabulary V, and w^j is the j-th word in V.

3.1.1. Trigram model

We compute the probability of corpus C:

$$P(C) = \prod_{i=1}^{||C||} P(w_i|w_1...w_{i-1}) \qquad (2)$$

To address sparseness, we apply Witten-Bell smoothing [17] which linearly interpolates the trigram probability and lower-order smoothed probabilities recursively. In general, the nth-order Witten-Bell probability is:

$$\begin{aligned} &P_{wb}(w_i|w_{i-n+1}...w_{i-1}) = \\ &\lambda_{w^{i-1}_{i-n+1}} P(w_i|w_{i-n+1}...w_{i-1}) + \\ &(1 - \lambda_{w^{i-1}_{i-n+1}}) P_{wb}(w_i|w_{i-n+2}...w_{i-1}) \end{aligned} \qquad (3)$$

The parameters $\lambda_{w^{i-1}_{i-n+1}}$ are computed by

$$\lambda_{w^{i-1}_{i-n+1}} = 1 - \frac{N(w^{i-1}_{i-n+1})}{N(w^{i-1}_{i-n+1}) + SC(w^{i-1}_{i-n+1})} \qquad (4)$$

where

$$N(w^{i-1}_{i-n+1}) = |\{w_i : count(w_{i-n+1}...w_{i-1} > 0\}| \qquad (5)$$

$$SC(w^{i-1}_{i-n+1}) = \sum_{w_j} count(w_{i-n+1}...w_{i-1}w_j) \qquad (6)$$

The intuition is to give more weight to trigrams in the training set, and to back off to the lower-order probabilities for those that are not.

Trigram t	Count	$logP(t)$	$logP_{wb}(t)$
come up with	20	-0.1140	-0.1760
come up to	0	$-\infty$	-1.8059
come up sing	0	$-\infty$	-6.1763

Table 1: Example trigram probabilites and smoothed probabilites from the WSJ dataset.

3.1.2. Long Short-Term Memory model

Long short-term memory (LSTM) [18] units are a special type of unit in recurrent neural networks (RNNs) designed to solve the vanishing-exploding gradient problem.

Let $s_1, ..., s_N$ be sentences in corpus C, which has N sentences. Suppose $w^i_1, ..., w^i_J$ are words in sentence s_i with J words. We define x^i_k to be the vector word *embedding* of w^i_k. We also define $h^l_t \in \Re^n$ to be the hidden state of layer l at timestep t. Then, for a sequence $w^i_1, ..., w^i_j$, we have $x^i_k = h^1_k$ for $1 \leq k \leq j$. We apply dropout regularization only to non-recurrent connections:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \rho \end{pmatrix} T_{2n,4n} \begin{pmatrix} D(h^{l-1}_t) \\ h^l_{t-1} \end{pmatrix} \qquad (7)$$

$$c^l_t = f \cdot c^l_{t-1} + i \cdot g \qquad (8)$$

$$h^l_t = o \cdot \rho(c^l_t) \qquad (9)$$

The vectors $i, f, o, c \in \Re^n$ represent input, forget, output, and cell vectors respectively. $T_{2n,4n}$ represents a linear transformation from $\Re^{2n}$ to $\Re^{4n}$, D represents the dropout operator which sets a percentage of its parameter to zero, σ represents the element-wise sigmoid activation function and ρ represents the element-wise tanh activation.

The hidden states h^L_t of the top layer L are used to infer y^i_t given a sequence $x^i_1, ..., x^i_t$. The model is trained to maxmize the probability $\prod_{i=1}^{N} \prod_{t=1}^{J} P(w^i_t|x^i_1...x^i_{t-1})$.

We train a 2-layer LSTM language model with 1,500 hidden units in each layer. Our vocabulary V contains the 50,000 most frequent words in the given corpus, and replaces all other words with $< unk >$. We use a dropout rate of 65% to the non-recurrent connections as described above.

3.2. r-ary Huffman coding

The r-ary Huffman coding method constructs trees in which each leaf node is a unique character from the alphabet. This results in a prefix-free coding in which the coded string for each character cannot be a prefix of the coded string of some other character. This is in contrast to Morse code in which, e.g., the letter '*e*' is encoded as '·', which is the prefix of 17 other alphanumeric characters, including '*s*' ('· · ·'), which can lead to ambiguities.

Huffman coding depends on the prior probability of each character, c_i, in the alphabet, which is simply the frequency of that character in the training corpus (i.e, $P_H(c_i) = Count(c_i)/\sum_j Count(c_j)$ [19].

The key to the present work is that the corpora upon which these frequencies are based are first processed by the word prediction software. A corpus to be studied is divided into training, development, and test sets. The development and test sets are processed using language models trained on the training set, so that as soon as a word appears in the prediction list, all remaining characters are replaced with a single special character corresponding to their index in the prediction list as exemplified in in Table 2. This processed *development* set is used to compute r-ary Huffman codes, and the input savings are calculated between the original and processed *test* sets as follows:

$$IS(org, proc) = \frac{len(org) - len(proc)}{len(org)} \cdot 100 \quad (10)$$

where org and $proc$ represent the original and processed $test$ sets respectively, and len calculates the number of switch-strokes needed to type characters in the sets which would be equal to the number of characters if Huffman coding is not used. For example, if '*p*' has code length 3, '*o*' has code length 2, and '#' has code length 1, then len('*pop*') is 8 where len('*p#*') is 4.

4. Experiments

Original sentence
the results met estimates of analysts who had already slashed their projections after the company said in late august that its 1989 earnings could
Processed sentence
t* re* met es@ **$** an**%** w@ **#** a* sla@ th**$** proj* af@ @ **$** **%** i**%** l a@ t* i@ 1**%** **$** c**#**

Table 2: Example of pre-processing, with $N = 5$ from WSJ dataset. Each special character (*,@, $, %, #) in **bold** represents different indices in the prediction list.

WSJ	$r = 3$	$r = 4$	$r = 5$	$r = 6$	$r = \infty$
$N = 3$	37.85	37.82	37.29	37.72	35.52
$N = 4$	39.37	39.25	39.18	38.80	37.69
$N = 5$	40.60	39.93	40.32	39.83	39.22
$N = 6$	41.46	40.39	41.16	40.62	40.51
Essay	$r = 3$	$r = 4$	$r = 5$	$r = 6$	$r = \infty$
$N = 3$	30.64	30.42	30.03	30.07	29.06
$N = 4$	32.22	31.70	32.07	31.29	31.35
$N = 5$	33.52	32.30	33.34	32.44	33.02
$N = 6$	34.45	32.77	34.17	33.39	34.39
SMS	$r = 3$	$r = 4$	$r = 5$	$r = 6$	$r = \infty$
$N = 3$	20.70	20.02	20.42	19.45	18.99
$N = 4$	22.10	20.54	21.73	20.81	20.81
$N = 5$	22.70	20.61	22.40	21.50	22.19
$N = 6$	22.96	20.72	22.76	21.90	23.27

Table 3: Input savings (in %) on each dataset for different N and r values using the trigram model.

We partition each dataset into five chunks; each is iteratively used for development and testing, and the others are used for training. As described in the previous section, all reported input savings count the proportion of *code symbols* saved – not characters; *this is an important distinction* – counting the latter, as is typical in AAC research, would not take our application of the Huffman code into account.

We run 5-fold cross validation for each number of coding symbols $r = \{3, 4, 5, 6, \infty\}$, where $r = \infty$ is the baseline character code length of 1, which mimics the case where Huffman coding is *not* used, and the length of the prediction list $N = \{3, 4, 5, 6\}$.

Table 3 shows input savings for each dataset using the trigram model for word prediction. As N increases, we get more input savings because the probability of the target word being in the prediction list goes up. However, a two-way F-test on N and r (Table 4) shows that the value of r does not affect the savings and that N and r do not interact.

Table 5 shows results obtained from the LSTM model for word prediction. The trigram model per-

	$SumSq$	$MeanSq$	F	$Pr(> F)$
N	0.0274	0.027399	5.554	0.0193
r	0.0012	0.001227	0.249	0.6184
$N : r$	0.0000	0.00010	0.002	0.9645

Table 4: Two-way F-test on N and r.

WSJ	$r = 3$	$r = 4$	$r = 5$	$r = 6$	$r = \infty$
$N = 3$	25.57	25.24	25.81	25.78	25.47
$N = 4$	27.11	26.96	27.64	27.81	27.70
$N = 5$	28.39	28.15	28.71	29.20	29.38
$N = 6$	29.35	29.22	29.66	30.06	30.70
Essay	$r = 3$	$r = 4$	$r = 5$	$r = 6$	$r = \infty$
$N = 3$	18.56	18.28	18.81	17.82	19.15
$N = 4$	20.02	19.74	20.43	19.78	21.23
$N = 5$	21.21	20.76	21.52	21.10	22.87
$N = 6$	22.08	21.87	22.17	22.08	24.16
SMS	$r = 3$	$r = 4$	$r = 5$	$r = 6$	$r = \infty$
$N = 3$	14.43	13.51	14.53	14.01	14.26
$N = 4$	15.61	14.20	15.66	15.24	16.06
$N = 5$	16.25	14.94	16.16	15.96	17.42
$N = 6$	16.83	15.55	16.52	16.36	18.52

Table 5: Input savings (in %) on each dataset for different N and r values using the LSTM model.

forms much better than the LSTM model; validating this finding on other sets of data should be the subject of future work.

The most input savings are obtained from the WSJ, and the least from the SMS dataset. This may be due to a more consistent grammatical structure in the former. The results clearly show that the input savings vary a lot depending on the corpus used. Trnka *et al.* [14], who showed that word prediction increased communication rates, also reported higher input savings when experimented on the Switchboard corpus, which has a different topic domain and vocabulary size. Analyzing the relation between the characteristics of a corpus (e.g., vocabulary size, level of formality, and grammatical structure) and the input saving rate is ongoing.

5. Conclusion and Future Work

In this paper, we examined input savings by combining word prediction models and r-ary Huffman coding on datasets with different levels of formality. Future work should evaluate performance 'online' with human participants, which may affect the optimal value of N, given a possible interaction effect with scanning time. Moreover, even though the value of r is not significant in terms of input savings, people might have different levels of difficulty in memorizing different code lengths[1]. Piloting the combination of word prediction and Huffman coding with real users is the next step, but the theoretical basis established in this paper is a requisite *first* step, since recruiting and training a sufficient number of participants will depend on constraining N and r, in order to obtain the appropriate statistical power. Alternatives to N-gram and LSTM models, initialized with pre-trained word embedding vectors, should also be applied to increasingly large datasets.

6. References

[1] R. Walls, J. J. Ratey, and R. I. Simon, *Rosen's Emergency Medicine: Expert Consult (Premium ed.)*, premium ed. St. Louis Missouri: Mosby, 2009.

[2] R. A. Spears and A. Holtz, *Spinal Cord Injury*. Oxford UK: Oxford University Press, 2010.

[3] S. L. Glennen and D. C. DeCoste, *The Handbook of Augmentative and Alternative Communication*. San Diego, CA: Singular, 1996.

[4] E. Dymond and R. Potter, "Controlling assistive technology with head movements a review," *Clinical Rehabilitation*, vol. 10, no. 2, pp. 93–103, 1996.

[5] S. J. Castellucci and I. S. Mackenzie, "Gestural text entry using Huffman codes," in *Proceedings of the International Conference on Multimedia and Human-Computer Interaction*, vol. 119, 2013, pp. 1–8.

[6] J. O. Wobbrock, B. A. Myers, and J. A. Kembel, "EdgeWrite: A Stylus-Based Text Entry Method Designed for High Accuracy and Stability of Motion," in *Proceedings of the 16th Annual ACM Conference on User Interface Software and Technology (UIST 03)*, 2003, pp. 61–70.

[1] We note that the fact that characters in the Morse code can be encoded in up to five symbols.

[7] T. Felzer and R. Nordmann, "Alternative text entry using different input methods," in *Proceedings of the 8th ACM Conference on Computers and Accessibility (ASSETS 06)*, 2006.

[8] P. Isokoski and R. Raisamo, "Device independent text input: A rationale and an example," in *Proceedings of the ACM Working Conference on Advanced Visual Interfaces (AVI 00)*, 2000, pp. 76–83.

[9] M. Porta and M. Turina, "Eye-S: a Full-Screen Input Modality for Pure Eye-Based Communication," in *Proceedings of the ACM 2008 Symposium on Eye Tracking Research and Applications (ETRA 08)*, 2008, pp. 27–34.

[10] B. Arons, "Techniques, perception, and applications of time-compressed speech," in *Proceedings of the 1992 Conference of the American Voice I/O Society*, 1992, pp. 169–177.

[11] A. Newell, S. Langer, and M. Hickey, "The role of natural language processing in alternative and augmentative communication," *Natural Language Engineering*, vol. 4, no. 1, pp. 1–16, 1998.

[12] I. S. Mackenzie, R. W. Soukoreff, and J. Helga, "1 Thumb, 4 Buttons, 20 Words Per Minute: Design and Evaluation of H4-Writer," in *Proceedings of the 24th Annual ACM Conference on User Interface Software and Technology (UIST 11)*, 2011, pp. 471–480.

[13] B. Roark, R. Beckley, C. Gibbons, and M. Fried-Oken, "Huffman scanning: using language models within fixed-grid keyboard emulation," *Comput Speech Lang*, vol. 27, no. 6, Sep 2013.

[14] K. Trnka, J. McCaw, D. Yarrington, K. F. McCoy, and C. Pennington, "Word Prediction and Communication Rate in AAC," in *Proceedings of the Fourth IASTED Conference on Telehealth and Assistive Technologies (Telehealth/AT 2008)*, 2008.

[15] I. S. Mackenzie, "KSPC as a characteristic of text entry techniques," in *Proceedings of the 4th ACM International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI 02)*, 2002, pp. 195–210.

[16] T. Chen and M.-Y. Kan, "Creating a live, public short message service corpus: the nus sms corpus," *Language Resources and Evaluation*, vol. 47, no. 2, pp. 299–335, 2012.

[17] I. H. Witten and T. C. Bell, "The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression," *IEEE Transactions on Information Theory, July*, vol. 37, no. 4, pp. 1085–1094, 1991.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] J. van Leeuwen, "On the construction of Huffman trees," in *Proceedings of ICALP*, 1976, pp. 382–410.