Language Models in Keyboard Emulation

Brian Roark
Oregon Health & Science University
Portland, OR, USA
roarkb@ohsu.edu

Andrew Fowler
Oregon Health & Science University
Portland, OR, USA
fowlera@ohsu.edu

Melanie Fried-Oken
Oregon Health & Science University
Portland, OR, USA
friedm@ohsu.edu

Abstract

In this extended abstract, we first discuss some of our recent work on text entry methods, which focus on keyboard emulation interfaces within the area of Augmentative and Alternative Communication (AAC). We then will discuss some important considerations for the learning and use of language models for text entry, including questions about generative versus discriminative modeling and word prediction/completion.

Author Keywords

language modeling, AAC, keyboard emulation, text entry

ACM Classification Keywords

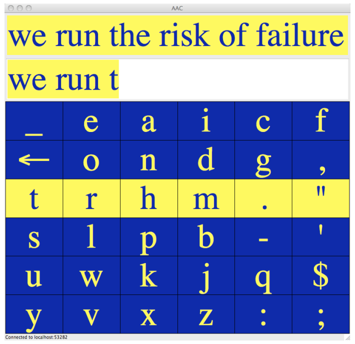
I.2.7 [Artificial Intelligence]: Natural Language Processing.

General Terms

Human Factors

Recent work in text entry

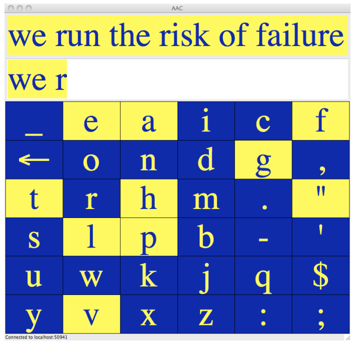
Our recent work in text entry is within the context of research on Augmentative and Alternative Communication (AAC). In particular, we have been working on an NIH-funded project "Translational refinement of adaptive communication system for locked-in patients," to develop a brain-computer interface (BCI) for text entry by individuals presenting with locked-in syndrome. For that project, we have developed a BCI text entry system that relies on language models to sequence stimuli (characters),


we run the risk of failure

we run t

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

Figure 1: Row/column scanning on a letter grid in frequency order


we run the risk of failure

we r

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

Figure 2: Huffman scanning on a letter grid in frequency order

which are then presented to the subjects in a rapid serial visual presentation (RSVP) paradigm [2, 7, 8]. Typing occurs when the system detects an event related potential (ERP) from the EEG signal – in particular the P300 – corresponding to the presentation of the target symbol (observed at roughly 300ms after onset of the target). This is a broadly interdisciplinary project, including experts in the following fields: speech and language pathology and AAC; clinical use of EEG; signal processing and pattern recognition from noisy signals; and language modeling. As evidenced by the title of the project, this is a user-centric project, with team members with locked-in syndrome providing feedback on system design.

The language modeling part of this project has been focused on questions of binary coding for keyboard emulation using a binary switch. Using the P300 ERP as a switch to drive text entry means that the information derived from the system user is a binary code, much like other binary switches widely used within AAC. This long-standing area of AAC research [5] has its correlates in general text entry research, such as the recent one-key challenge of MacKenzie [6]. Within AAC, scanning systems on grids of symbols are widespread, whereby rows and columns are highlighted and the user types by pressing a button when the row or column of their target symbol is highlighted. Figure 1 shows such a typing interface.

Row/column scanning of this sort assigns a binary code to each symbol in the grid. Scanning starts at the top of the grid, and each row that does not hold the target symbol is a zero bit; once the row is selected, column scanning commences from left-to-right. For instance, the binary code for the letter 'h' in the grid in Figure 1 is '001001' (**skip** row 1, **skip** row 2, **select** row 3, **skip** column 1, **skip** column 2, **select** column 3). Frequency ordering the grid, as has been done for the grid in Figure 1, places the most common symbols in the upper left-hand corner, where they receive the shortest codes and hence can be input more quickly. One might rely on the previously typed context to assign the codes, however with row/column scanning this would require dynamically re-organizing the grid to place the symbols in the upper left-hand corner, which imposes additional cognitive and visual scanning overhead – to the extent that conventional wisdom is that dynamic grid organization slows down scanning.

As an alternative to row/column scanning, we have been pursing an approach which we call Huffman scanning [9, 1]. In this work, we use n-gram language models, which condition the probability of each symbol given the previously typed symbols. Using these multinomial models, we can then build a Huffman code, which produces codes with the minimum expected bits [3]. We then highlight the letters in the grid that correspond to bit '1' at the current position. Figure 2 shows this sort of scanning approach on the same grid as is used for row/column scanning. The user focuses on the target symbol as symbol groups of varying size are highlighted in an optimal sequence. When the target symbol is among those highlighted, the user selects that group, quickly winnowing the set down to the target symbol. Since the highlighted groups can be any subset of the grid, and can vary according to the context, we can produce much shorter codes than with row/column scanning without having to reconfigure symbol positions in the grid. Importantly, the approach takes into account the probability of error at each bit of the code. We have demonstrated typing speedups over row/column scanning with this approach.

Issues in language modeling for scanning

Within the context of the BCI project mentioned in the earlier section, as well as for general keyboard emulation work being done by the language modeling team of that project, there are several important issues that we have been or are interested in investigating. At the heart of each of these issues is the need to speed up text entry,

Generative vs discriminative modeling
An important distinction in natural language processing is generative models versus discriminative sequence models. For example, for word tagging tasks such as part-of-speech tagging, generative (joint) models like hidden Markov models – which assign a joint probability of hidden state sequences and observations – are contrasted with discriminative (conditional) models like conditional random fields [4], which assign the conditional probability of the hidden state sequence given the observed sequence. In language modeling, discriminative approaches perform parameter estimation with a globally conditional likelihood objective [10], leading to models that are optimized to disambiguate between application-specific competitors. For example, in speech recognition, parameter estimation in discriminative language modeling occurs by looking at the output of the baseline speech recognizer (e.g., a conditionally normalized word lattice) and moving parameters so as to favor low error rate hypotheses in the output while penalizing higher error rate hypotheses. Similar approaches can be used for text entry systems.

which is unacceptably slow for many of the individuals we are trying to support in the AAC community: whereas spoken language reaches more than a hundred words per minute and an average-speed typist using standard touch typing will achieve approximately 35 words per minute, a user of an AAC device will typically input text in the 3-10 words per minute range. In addition, we would like to approach these topics as optimization problems, with well-motivated information theoretic solutions.

The row/column and Huffman scanning approaches discussed above assign a unique binary code to each symbol in the grid. The use of language models to assign binary codes contrasts with more typical use of language models to disambiguate between symbols that have been assigned to the same key, as is typically used for predictive text entry in mobile computing (e.g., T9). In the latter approach, ambiguity persists and is resolved downstream; in scanning methods, bits are provided until the symbol is uniquely identified (e.g., at the intersection of the selected row and column). This difference has important ramifications for how to approach language model parameter estimation (see the generative versus discriminative language modeling sidebar). The key distinction is disambiguation of competing *sequences* of symbols or competing *single* symbols.

Language models are widely used in a number of applications, such as automatic speech recognition, machine translation and optical character recognition. In these applications, the task is to transcribe or translate entire sequences of symbols or words during decoding. At each position in the sequence, what has come before that word in the sequence is not known for certain. Rather, there is typically a probability distribution over possible previous histories, and decoding is a *global* inference procedure, such as widely used Viterbi decoding.

In text entry, the system user is typically reviewing what has been typed during typing and correcting that text when errors arise. Hence, much more than in these other applications, the prefix sequence is known. For the scanning approaches, under the assumption that the typed sequence is the intended sequence, the previously typed sequence is fully given at each letter. For typical predictive text entry approaches in mobile computing, the current word is not fully known – rather, it is ambiguous between several possibilities – but the words farther back in the history are known. Once the current word is completed (a space is typed), that word becomes part of the known letter history. In either case, having relatively high confidence in previously typed letters is a notable difference from other language modeling applications.

Note that for scanning methods, which use language models to assign codes to letters in a sequence, if there is no ambiguity in what the previously typed sequence was, then a maximum likelihood optimized generative language model (e.g., standard relative frequency estimated n-gram models) also optimize the conditional likelihood. In other words, for this special case, the distinction between generative and discriminative estimation goes away. However, when the language model is also performing some kind of disambiguation from among competing sequences, then discriminative language models should yield improvements in practice. Yet most text entry systems continue to rely upon generative language models.

Of course, the assumption that the typed history is "correct" is not generally true, as typed sequences often contain unintended errors, misspellings, or deliberate errors. Models for text entry systems should be designed to accommodate them. One approach to this problem is to treat it as a decoding task, retaining ambiguity in the typed history and giving some probability mass to previous histories containing common typographical errors and abbreviations. In such an approach one must decide how far back into the typed history ambiguities persist.

Practical considerations
An important question is whether or not one's language model approach to text entry actually demonstrates an improved user experience, both in terms of typing speedups and general usability. A language model that is a poor match to the user or the context might result in situations where effective text entry is impeded rather than assisted. Research has shown that, for users of AAC, the quality of word completion and prediction, for example, influences user attitude towards making use of such utilities [11]. This would suggest that fewer high precision predictions is a much more effective strategy than high recall strategies such as always predicting the top k completions or predictions. In addition, effective estimation of non-printable character probability, e.g., probability of word or character delete as an option, is critical to enabling effective repair. As noted in our description of word prediction, cognitive overhead must also be carefully considered. A system that displays word predictions, corrects spelling, alters previously typed characters, and changes dynamically may well be too much for a reasonable user to manage. This is a particularly important perspective when building systems for users with severe impairments.

Beyond text entry disambiguation, language models are also used for word completion and prediction, in which the language model displays the most probable words consistent with the currently typed prefix. Because word completions must be located and read by the user every time they are provided, showing them every time likely introduces unneeded cognitive load. In ambiguous contexts, such as the beginning of a word or sentence, it might not be useful to show any word completions. Similarly, showing too many word completions – in any context – can be cognitively and graphically difficult. The pragmatic solution to this problem is to fix the maximum number of displayed word completions, and to refrain from showing any completions until the user is some fixed distance into the current word (two or three letters). A more sophisticated solution is to decide dynamically, based on the language model word probabilities, whether word completions should be shown, and how many to show. With a high quality probabilistic model, it should be possible to base interface decisions – such as when to suggest word completions – on information-theoretic calculations of expected utility, rather than heuristic rules of thumb.

Acknowledgements
This research was supported in part by NIH Grant #1R01DC009834-01. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NIH.

References
[1] Beckley, R., and Roark, B. Asynchronous fixed-grid scanning with dynamic codes. In *Proceedings of the 2nd Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)* (2011), 43–51.
[2] Hild II, K., Orhan, U., Erdogmus, D., Roark, B., Oken, B., Purwar, S., Nezamfar, H., and Fried-Oken, M. An ERP-based brain-computer interface for text entry using rapid serial visual presentation and language modeling. In *Proceedings of the ACL 2011 Demo Session* (2011).
[3] Huffman, D. A method for the construction of minimum redundancy codes. In *Proceedings of the IRE*, vol. 40(9) (1952), 1098–1101.
[4] Lafferty, J., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning* (2001), 282–289.
[5] Lesher, G., Moulton, B., and Higginbotham, D. Techniques for augmenting scanning communication. *Augmentative and Alternative Communication 14* (1998), 81–101.
[6] MacKenzie, I. The one-key challenge: Searching for a fast one-key text entry method. In *Proceedings of the ACM Conference on Computers and Accessibility (ASSETS)* (2009), 91–98.
[7] Orhan, U., Erdogmus, D., Roark, B., Purwar, S., Hild II, K., Oken, B., Nezamfar, H., and Fried-Oken, M. Fusion with language models improves spelling accuracy for ERP-based brain computer interface spellers. In *Proceedings of the 33nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (2011).
[8] Orhan, U., Hild II, K., Erdogmus, D., Roark, B., Oken, B., and Fried-Oken, M. RSVP keyboard: an EEG based typing interface. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (2012).
[9] Roark, B., de Villiers, J., Gibbons, C., and Fried-Oken, M. Scanning methods and language modeling for binary switch typing. In *Proceedings of the NAACL-HLT Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)* (2010), 28–36.
[10] Roark, B., Saraclar, M., and Collins, M. Discriminative n-gram language modeling. *Computer Speech and Language 21*, 2 (2007), 373–392.
[11] Trnka, K., Yarrington, D., McCaw, J., McCoy, K., and Pennington, C. The effects of word prediction on communication rate for AAC. In *Proceedings of HLT-NAACL; Companion Volume, Short Papers* (2007), 173–176.