Autocompletion and autodeletion in binary typing systems

Andrew Fowler[†], **Brian Roark**[†], **Melanie Fried-Oken**[◦]
[†]Center for Spoken Language Understanding [◦]Child Development & Rehabilitation Center
Oregon Health & Science University
{fowlera,roarkb,friedm}@ohsu.edu

Abstract

In this extended abstract, we describe a method for language model-assisted text entry in an AAC (Alternative and Augmentative Communication) environment. We outline an algorithm for determining the optimal actions for a binary typing system in which the user is queried one character at a time. The goal of such a system is to maximize keystroke savings by automatically typing highly-probable characters and automatically deleting improbable characters. This algorithm is robust to uncertainty in user input, uses all past input to determine its next action, and allows the user to delete incorrect characters. We describe simulations that will be done, and discuss potential trade-offs between theoretically optimal performance and the overall user experience.

1 Introduction

Individuals with severe physical disabilities are often subject to extremely restricted forms of communication. Without the ability to speak, gesture effectively, or even type on a traditional keyboard, many such individuals are limited to binary forms of interaction, in which sequences of 'yes' and 'no' are the only way to voluntarily convey discrete information to those around them. These signals are typically triggered by a small physical motion such as a button click or an eye blink, but can also be extracted from EEG responses such as the P300 event-related potential (ERP).

Many systems for binary typing have been developed, including row/column scanning, Huffman scanning, and even historical methods such as Morse code (Beukelman and Mirenda, 1998; Roark et al., 2010). For the purposes of this abstract, we will be considering the RSVP Keyboard™ system (Rapid Serial Visual Presentation). This method presents a sequence of individual symbols to the user, and the user indicates with the binary switch when the desired symbol appears. When a symbol has been selected, it is typed on the screen.

A crucial consideration for the usability of any binary typing system is typing speed. While no implementation can be expected to compete with the speed of standard verbal communication, a well-designed typing system can mean the difference between a fruitful dialogue and a prohibitively tedious interaction. In fact, since many individuals with verbal impairments are often subject to other medical complications, one can easily imagine a scenario in which typing speed could become extremely vital. To this end, statistical language models are commonly used to improve typing speed in AAC environments, and binary typing is no exception. In the case of the RSVP Keyboard system, language models are used to determine the optimal ordering of the sequence of symbols to be presented, from most to least probable.

One of the most widespread applications of language models in AAC is in facilitating word and phrase completion. However, it is not immediately obvious how best to incorporate word completion into an RSVP Keyboard system. In other AAC systems, such as direct selection, probable word completions can be displayed in a separate area in an interface grid. In RSVP Keyboard binary typing, however, there are essentially only two options: Include word completions in the symbol sequence itself, or automatically complete probable words without explicit user input.

The latter of these choices, so-called *autocomplete*, is familiar to most people who have used SMS messaging on cellular phones. While many systems include simple heuristic rules for autocomplete (such as waiting until a certain fixed number of symbols have been typed in a word before autocomplete becomes available), the method presented in this abstract describes a more principled way of accumu-

lating posterior probabilistic observations about a string and using these to generate an optimal action.

2 Method

The crux of our method is to ask the following question: Given the characters typed so far in a string, a set of prior observed probability distributions (over characters) pertaining to different positions in that string, and a language model, what is the optimal action? There are four actions in RSVP Keyboard:

1. Display a character to the user (ask "is this your character?")
2. Display the delete symbol to the user (ask "was the previously-typed character incorrect?")
3. Type a character
4. Delete the rightmost character

In its simplest formulation, our method is as follows:

1. If any character is more than 50% probable, type it (Action 3)
2. If delete has more than 50% probability (i.e. the previous character drops below 50%), delete the rightmost character (Action 4)
3. Otherwise, query the user about the most probable symbol (Action 1 or 2)
4. Recalculate priors and repeat

The details of this method lie in how one calculates these probabilities. The basic idea is to get the prior probabilities for the given context from a language model of your choice, and then use Bayesian inference to update those priors based on observations. In our case, the posterior observations will be of the form "in context c, user was presented with symbol s and responded positively with certainty p". During the course of typing a string, a whole series of these observations will be collected, and each of them contributes to the recalculated prior probability at any point in the string. For instance, if the current string is `fo`, and the string `the` was rejected previously in the same session with probability 0.8, the probability of the next character being `r` will be slightly higher, since the rejection of the substring `the` makes all strings *not* `the` more likely.

One can make a handful of other simplifying observations:

1. The user saying *yes* with probability p is equivalent to the user saying *no* with probability $1 - p$.
2. The probability of delete at a given context is always $1 - p$, where p is the prior probability of the rightmost character.
3. If the user responds to the delete symbol with probability p, this is equivalent to the user responding to the rightmost character, in the previous context, with probability $1 - p$.

3 Discussion

Having run some preliminary tests, we have observed that this system exhibits a useful combination of autocorrect/autodelete and user querying. In uncertain contexts, it queries the user repeatedly, and in contexts with more peaked distributions it performs more automatic typing. The method is robust to uncertainty of observations, which makes it very suitable for noisy binary input modes such as the P300 ERP. There is also an important trade-off involved in autocomplete: It is perfectly reasonable to raise the decision criteria probability to some number higher than 50%. By doing so, one would sacrifice the theoretically optimal typing speed, but it may well be more satisfying to use the system from a user standpoint, because it will be less likely to type (and later have to delete) incorrect characters.

We will present the results of several simulations, examining the performance of this method under various observation uncertainties, language models, corpora, and autocomplete threshold values. We will also provide a detailed mathematical formulation of how the prior probabilities are calculated and updated.

Acknowledgments

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

D. Beukelman and P. Mirenda. 1998. *Augmentative and Alternative Communication: Management of Severe Communication Disorders in Children and Adults*. Paul H. Brookes, Baltimore, MD, second edition.

B. Roark, J. de Villiers, C. Gibbons, and M. Fried-Oken. 2010. 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)*, pages 28–36.