August 14, 2018

User Interface for Input With Switches Using Machine Learned Huffman Codes

Etienne J. Membrives

User interface for input with switches using machine learned Huffman codes

ABSTRACT

Users with special circumstances, such as limited mobility or physical strength, are often unable to utilize the normal keyboard of a device. To overcome these difficulties, these users utilize alternative mechanisms for typed input, such as a mouse, trackpad, switches, buttons, etc. These mechanisms operate by mapping the full set of possible inputs onto a limited number of buttons, which makes their use cumbersome and slow. This disclosure utilizes Huffman coding to optimize the encoding of a large set of symbols into a set of codewords based on the probability of use of each symbol, calculated via a trained machine learning model. Given a reasonably accurate machine-learned prediction model, the techniques of this disclosure ensure that generating the desired typed input can be accomplished with minimal number of switch selections.

KEYWORDS

- Huffman codes
- accessibility
- Input device
- Input switches
- Input screen areas
- N-ary tree

BACKGROUND

Users with special circumstances, such as limited mobility or physical strength, are often unable to utilize the normal keyboard of a device. These limitations apply to a traditional hardware keyboard connected to a desktop or laptop computer as well as to on-screen keyboards

commonly utilized on mobile devices. The use of the normal device keyboard for typed input can be cumbersome even for regular users when they encounter atypical situations, such as typing while carrying other objects. To overcome these difficulties, these users utilize alternative mechanisms for typed input, such as a mouse, trackpad, switches, buttons, etc. These mechanisms typically operate by mapping the full set of possible inputs onto a limited number of buttons, thus making their use cumbersome and slow. Moreover, these mechanisms rely on the visual layout of the applications, which is not designed with such atypical users or circumstances in mind, thus often making it necessary to input a large number of switch selections to provide the desired input.

DESCRIPTION

This disclosure describes techniques for mapping a set of inputs to a limited set of switches to minimize the time and effort required to provide input via input switches. Specifically, the techniques utilize Huffman coding to optimize the encoding of a large set of symbols into a set of codewords based on the probability of use of each symbol. The probability is calculated via a trained machine learning model. If the user provides permission, the model takes into account the user's previous input and the user's context, such as location, date, time, currently used application, etc.

The large set of possible input symbols is first turned into a tree representation where each leaf node includes one of the symbols within the set along with the corresponding probability (provided by the trained machine learning model) of the symbol being the next input symbol desired by the user. This initial tree is processed iteratively and converted into an N-ary prefix tree where *N* is the number of input switches specified by the user, with a minimum of 2. At each iteration, a set of *N* nodes with the lowest probability of being the next input is merged

into a new node such that the original nodes are children of the new node. The resulting N-ary prefix tree forms a Huffman tree in which each non-leaf node has *N* children.

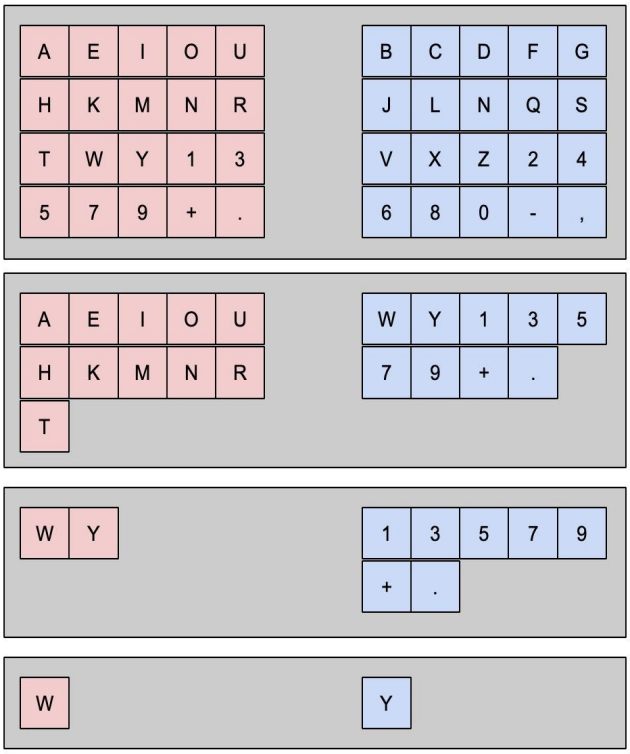

Fig. 1: Example Huffman tree for 40 symbols and 2 switches displayed in two on-screen areas to select the desired input "W"

To provide input, the user is shown an on-screen grid split into N areas with each area displaying the *N* children of the currently selected node, ordered by probability. The probability of the next desired input symbol being included within a given node is computed as the sum of the individual probabilities of all children of that node. Initially, the tree is presented with the root node selected and the *N* children of the root node occupying the *N* on-screen areas. When the user selects the desired area via a switch, the node assigned to that area is selected. If the selected node has children, the *N* on-screen areas are updated to display the *N* corresponding children of the selected node. Alternatively, if the node is a leaf node representing one of the

input symbols, this symbol is provided as input to the system, and the process restarts to obtain the next input symbol. Fig. 1 shows an example of the screens presented to a user with a 40-symbol set and 2 on-screen areas when the desired user input is the letter "W."

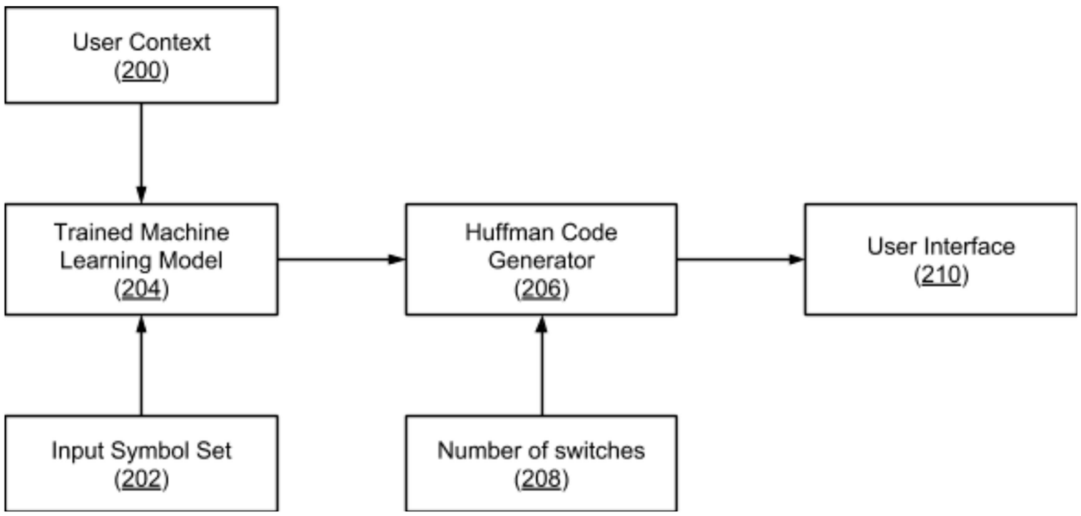

Fig. 2: Generating a Huffman tree for switch input using a trained machine learning model

Fig. 2 shows the components involved in the implementation of the techniques of this disclosure. With the user's permission, a trained machine learning model (204) utilizes the user's context (200) and other configuration information (202) in order to predict the probability corresponding to each of the symbols within the input set being the next input symbol desired by the user. The Huffman code generation module (206) takes these probabilities into account for generating a N-ary Huffman tree as described earlier, *N* is the number of switches (208) specified by the user. The user interface (210) displays the N-ary Huffman tree within *N* on-screen areas and the users to select among areas with switches until the desired input symbol is reached.

In the description above, the switches used for symbol selection may be any suitable hardware or software selection mechanism, such as physical buttons, surfaces, etc. The symbols

in the input set can describe a single character, as on keyboards. However, a symbol may be any input, such as groups of characters, entire words, or control commands (e.g., function keys, arrow keys, etc.), to which the machine learning model can ascribe a probability.

The machine learning model may be based on any suitable machine learning algorithm, such as decision trees, neural networks, support vector machines, etc. Given a reasonably accurate machine-learned prediction model, the techniques of this disclosure can ensure that minimal number of switch selections are required for generating the desired typed input. In a variation of the above described operation, the system can be simplified by pre-computing generic probabilities and the Huffman code associated with each symbol and utilizing this static code instead of computing the probabilities for all symbols each time a new symbol is typed. While such a simplification can reduce the amount of run-time computation, it may result in suboptimal performance, with more switch selections needed by the user to input the desired symbols.

The ease of selecting the displayed symbols can be improved by using the Hu-Tucker algorithm to create an equivalent Huffman tree where the symbols are sorted alphabetically, thus ensuring that the symbols are sorted when presented on screen. While such an approach makes it easier and faster for the user to find the symbol on the screen, it may reduce the optimality of the generated Huffman tree. The symbols can also be made easier to scan by using colors to code symbols by type, such as uppercase, lowercase, numbers, special characters, control commands, etc.

For users who are only able to use a single switch, the techniques of this disclosure may be provided by combining them with a regular switch access system and sweeping the on-screen areas. In such cases, areas later in the sweeping order need more time to be selected. This is

taken into account when building the Huffman code by assigning a cost to each area that increases with its position, thus reflecting the increased time needed to select that area, and using an algorithm such as that described in M.J. Golin and G. Rote in [1].

The techniques of this disclosure may be used as an additional mode or a replacement for on-screen keyboards on mobile devices as well as traditional computers. The techniques may also be provided as extensions or add-on for specific applications, such as web browsers.

Further to the descriptions above, a user may be provided with controls allowing the user to make an election as to both if and when systems, programs or features described herein may enable collection of user information (e.g., information about a user's social network, social actions or activities, profession, a user's preferences, or a user's current location), and if the user is sent content or communications from a server. In addition, certain data may be treated in one or more ways before it is stored or used, so that personally identifiable information is removed. For example, a user's identity may be treated so that no personally identifiable information can be determined for the user, or a user's geographic location may be generalized where location information is obtained (such as to a city, ZIP code, or state level), so that a particular location of a user cannot be determined. Thus, the user may have control over what information is collected about the user, how that information is used, and what information is provided to the user.

CONCLUSION

Mechanisms that allow typed input via a limited number of switches operate by mapping the full set of possible inputs onto a limited number of buttons, thus making their use cumbersome and slow. This disclosure utilizes Huffman coding to optimize the encoding of a large set of symbols into a set of codewords based on the probability of use of each symbol, calculated via a trained machine learning model. Given a reasonably accurate machine-learned

prediction model, the techniques of this disclosure can ensure that minimal number of switch selections will be required for generating the desired typed input. An input symbol may be any input, such as groups of characters, entire words, or control commands (e.g., function keys, arrow keys, etc.), to which the machine learning model ascribes a probability. The techniques of this disclosure may be used as an additional mode or as a replacement for on-screen keyboards on mobile devices as well as traditional computers.

REFERENCES

1. Golin, Mordecai J., and Günter Rote. "A dynamic programming algorithm for constructing optimal prefix-free codes with unequal letter costs." IEEE Transactions on Information Theory 44, no. 5 (1998): 1770-1781.
2. Hu, Te C., and Alan C. Tucker. "Optimal computer search trees and variable-length alphabetical codes." SIAM Journal on Applied Mathematics 21, no. 4 (1971): 514-532.
3. Roark, Brian, Melanie Fried-Oken, and Chris Gibbons. "Huffman and linear scanning methods with statistical language models." Augmentative and Alternative Communication 31, no. 1 (2015): 37-50.
4. Kim, Seung Wook, and Frank Rudzicz. "Combining word prediction and r-ary Huffman coding for text entry." SLPAT 2016 Workshop on Speech and Language Processing for Assistive Technologies, 13 September 2016, San Francisco, USA
5. Huffman, D. (1952). "A Method for the Construction of Minimum-Redundancy Codes". Proceedings of the IRE. 40 (9): 1098–1101.