Automatic Abbreviation Generation

Gregg M. Stum, Patrick W. Demasco, and Kathleen F. McCoy

Applied Science and Engineering Laboratories
University of Delaware/ A.I. duPont Institute
Wilmington, Delaware USA

Abstract

This work is part of an augmentative communication project being conducted at the Applied Science and Engineering Laboratories of the University of Delaware and the A.I. duPont Institute. The goal is to reduce the demands of traditional abbreviation expansion systems, on both the clinician and the client, by providing a clinical aid for constructing a fixed set of abbreviations for a vocabulary, and a user tool for recognizing a variable set of abbreviations for a vocabulary item. To this end an abbreviation generator produces all the possible abbreviations associated with a vocabulary under a well defined model of abbreviation.

Background

An abbreviation expansion system is usually a computer-based typing tool that replaces a sequence of characters, representing an abbreviation for a word, with that word. The set of abbreviations associated with a vocabulary is an abbreviation scheme. An abbreviation strategy is the rule or set of rules used in forming the abbreviation.

Vanderheiden and Kelso [5] identify abbreviation expansion as an important technique in an acceleration vocabulary of an individual using a communication aid. They show that a relatively large amount of communication is composed of a relatively small number of words, and that there is a high degree of consistency in this set of the most frequent words among different users. After analyzing several abbreviation schemes for this vocabulary, they find their idiosyncratic-logical scheme the most effective.

As illustrated by this study, choosing a good abbreviation scheme for a specific vocabulary requires considerable research time and effort. For a set of the most frequently used vocabulary items this investment is worthwhile since the set is essentially fixed. However, this set only constitutes the majority of text produced, not its totality. The remaining portion is drawn from a variety of context-specific vocabularies, that may be different among individuals.

Thus a specific vocabulary is not uniformly appropriate to all users. This means that the clinician is responsible for constructing abbreviation schemes for a number of clients. This task is even more ominous considering that the abbreviations must be tailored to the client and that different clients have different preferences about how something should be abbreviated. So a specific scheme for a vocabulary is not uniformly appropriate, and the task of generating abbreviations that are easy for each client to recall is enormous.

Statement of the Problem

For each user of an abbreviation expansion system a vocabulary of words to be abbreviated must be specified, and the individual abbreviations for the words determined. This requires knowledge in constructing abbreviation schemes, consideration of the user's cognitive skills, physical abilities, and abbreviation preferences, and the time and effort of the clinician. These requirements are not generally insignificant.

Once a scheme is established for a vocabulary, the user must either memorize it, or have it available to reference. For any scheme of appreciable size, the cost of cognitive load or lookup time is significant, and may in fact exceed the savings in keystroke reduction.

These considerations identify two basic related problems: the construction of an effective abbreviation scheme for a given vocabulary; and the overhead imposed on the user of such an abbreviation scheme.

Approach

The *abbreviation scheme constructor* provides to the clinician several sets of fixed abbreviation assignments for a vocabulary, with each of these schemes representing the application of a specific abbreviation strategy. From these, the clinician selects the most appropriate scheme for a specific client. The *adaptive flexible abbreviation expander* obviates the need for any fixed scheme by recognizing any well-defined abbreviation for a vocabulary item with roughly the same speed as a fixed abbreviation expander. Naturally, expansions for such abbreviations are generally not unique. This expander presents a list of the most preferred expansions based on human preferences and the user's history. The *automatic abbreviation generator* is a preprocessor for both the constructor and the expander that provides their required information and reduces their run-time computational requirements to make implementations practical. The generator does not use strategies in producing the abbreviations, but rather it rates an abbreviation according to the strategy it best represents for the word from which it was obtained.

The Basic Abbreviation Context

In order to identify the fundamental design and implementation issues in creating the generator, a basic model of the abbreviation context is defined. This model is intended to be only sophisticated enough to exhibit the computational requirements without being complicated by features that might be desired in an implementation, but do not contribute to its computational complexity. These features are then supported as extensions to the basic model.

With these objectives, the model is this. The vocabulary being abbreviated consists of single word items. An abbreviation is taken to consist of only letters from the word appearing in the same relative order as in the word. Thus, the number of possible abbreviations for a given word of length n is 2^n-1. The important characteristic to note is that this number grows exponentially on the length of the word.

Extensions

Examples of extensions to the basic abbreviation model include: using numerals, special characters, and letters not appearing in the word for phonetic or mnemonic value; and allowing phrases, utterances, and templates for them as vocabulary items. None of these extensions change the exponential nature of the number of possible abbreviations, and are accommodated by the generator.

The Automatic Abbreviation Generator

The generator has a rather straightforward design. It takes each vocabulary item, produces its set of possible abbreviations, and stores this set in an expansion table. Each abbreviation in the table indicates both the word from which it was obtained, and the abbreviation strategy it most closely represents for that word. It is often the case that an abbreviation represents the application of several strategies. When the same abbreviation is produced from more than one word, its entry in the table indicates a list containing all the words from which it was obtained. This list is maintained in sorted order according to preference ratings assigned to the various abbreviation strategies. Extensions to the basic model are incorporated into the generator by the use of special auxiliary functions. Each special function encodes a specific extension and is invoked by the generator as required.

Implications

The Abbreviation Scheme Constructor

An abbreviation strategy is taken to be a rule used consistently across the entire vocabulary. Only one rule is used and this rule is applied to each word. Ehrenreich [2], Hodge and Pennington [3], and Streeter, Ackroff and Taylor [4], have all demonstrated regularities in human generation of abbreviations for a given vocabulary item, and preferences for particular abbreviation strategies.

The three most basic strategies identified in all these studies are: truncation, vowel-deletion, and combination. Truncation is taking a prefix of the word. Vowel-deletion is eliminating all occurrences of vowels in the word. Combination is first an application of vowel-deletion, then truncation of the result to a maximum length if necessary. These strategies are taken as the basic model for the constructor, with more complex strategies included as extensions.

The constructor is a clinical tool used by a clinician in preparing a communication aid that includes a fixed abbreviation expander. The clinician identifies the user's vocabulary and selects the strategies to be represented. The constructor then gives an abbreviation scheme for each strategy specified. After reviewing these various alternatives, the clinician selects the most appropriate scheme for the user. If possible, this scheme is sent automatically to the fixed abbreviation expander; otherwise it is entered manually. Figure 1 illustrates this process

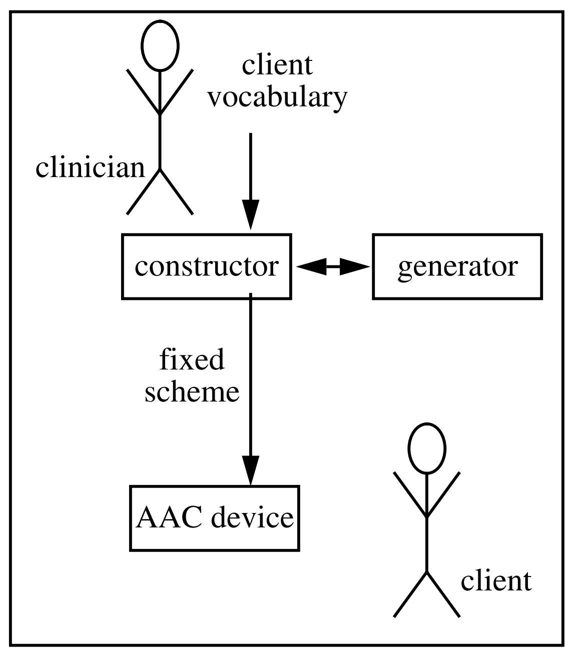

Figure 1: The Abbreviation Scheme Constructor produces a fixed abbreviation scheme for the expander of an AAC device.

Automatic Abbreviation Generation

The Adaptive Flexible Abbreviation Expander

A major drawback of fixed abbreviation schemes is the cost of cognitive load or lookup time imposed on the user. An expander that does not require a predefined abbreviation scheme, can deal efficiently with a set of well-defined abbreviations for a word, and can adapt to both the user's preferences and vocabulary, has the promise of giving the user an effective means for realizing meaningful keystroke savings.

The basic model of the expansion is that as the user types each letter of the abbreviation, the expander presents, separate from the text, a menu of some number of possible expansions for the abbreviation so far. The user then presses a key associated with the desired expansion. If the desired expansion is not presented, the user either types a special key for more expansions, or types another letter of the abbreviation. The expander then records this selection and replaces the abbreviation with the expansion. This model is intended only to demonstrate the expander's adaptiveness and flexibility. Its user interface is an entirely seperate issue and does not impact on its design.

The expander is a gateway to some other software application, such as a communication aid. As with the constructor, the user's vocabulary must be identified, and in this case given to the generator. The generator then passes the resulting sets of abbreviations to the expander. Figure 2 illustrates this process.

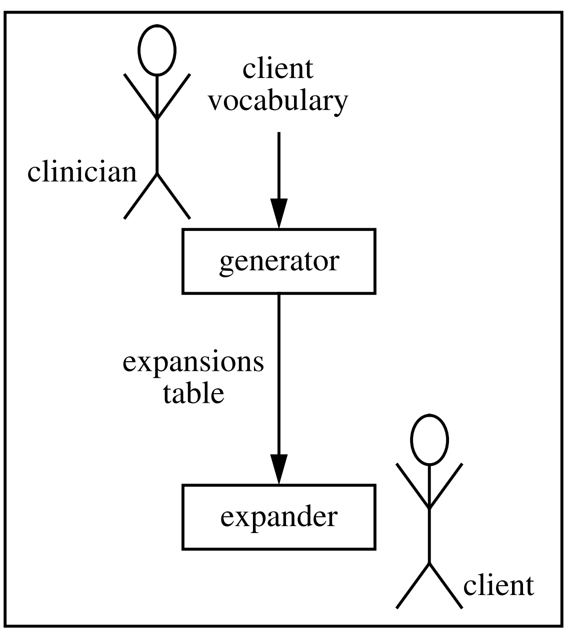

Figure 2: The Adaptive Flexible Abbreviation Expander serves as the expander of an AAC device

Discussion

This work is an extension of word compansion as described in Demasco, Lillard, and McCoy [1]. In word compansion, all the work in interpreting an abbreviation occurs at run-time. That is, after the compansion expander is given an abbreviation, it both manipulates this abbreviation and filters subsets of possible expansions from the dictionary. This interpretation is rather computationally expensive, especially compared to the simple table lookup of fixed abbreviation expansion. The objective of the adaptive flexible expander is to eliminate this disparity by moving the computationally expensive work to the generator, leaving only a table lookup and some optional bookkeeping for the expander.

The constructor represents applying knowledge about human abbreviation behavior to the results of the generator. For users not having sufficient cognitive ability either to appreciate the expander's flexibility, or to adjust to its adaptiveness, a fixed abbreviation expander is more appropriate. In this case, the constructor is a useful clinical aid for tailoring the fixed abbreviation scheme to the user.

Implementation considerations have a direct bearing on the actual limits of the generator. By design it has a high order of computational complexity. That complexity comes from both the problems it is addressing, and the fact that it is taking the computational burden away from the constructor and the expander. Given that the generator itself is not required by the run-time system, it is reasonable to assume that it runs on a hardware platform containing a large memory, a fast processor, and a large, fast secondary storage like a disk. The actual values of these parameters affect the limits of the generator's performance. A prototype of the generator is currently being used to examine basic implementation properties such as reasonable limits on the size of the vocabulary, length of vocabulary items, and number and choice of abbreviation strategies supported.

Acknowledgments

This work is supported by Grant Number H133E80015 from the National Institute on Disability and Rehabilitation Research. Additional support has been provided by the Nemours Foundation.

References

1. Demasco, P., Lillard, M., and McCoy, K. (1989). Word compansion: allowing dynamic word abbreviations. Proceedings of the RESNA Twelfth Annual Conference, 282 - 283.

2. Ehrenreich, S. L. (1982). Computer abbreviations: evidence and synthesis. Human Factors, vol. 27, no. 2, 143 - 155.

3. Hodge, M., and Pennington, F. M. (1973). Some studies of word abbreviation behavior. Journal of Experimental Psychology. vol. 98, no. 2, 350 - 61.

4. Streeter, L. A., Acroff, J. M., and Taylor, G. A. (1983). On abbreviating command names. The Bell System Technical Journal, 62, 1807 - 1826.

5. Vanderheiden, G. C., and Kelso, D. P. (1987). Comparative Analysis of Fixed-Vocabulary Acceleration Techniques. AAC Augmentative and Alternative Communication, vol. 3, no. 4, 196 - 206.

Contact

Gregg M. Stum
Applied Sciences and Engineering Laboratories
A. I. duPont Institute
PO Box 269
Wilmington, DE 19899
Email: stum@asel.udel.edu