Abbreviated Text Input

Stuart M. Shieber
Harvard University
33 Oxford Street
Cambridge, MA 02138 USA
shieber@deas.harvard.edu

Ellie Baker
Harvard University
33 Oxford Street
Cambridge, MA 02138 USA
ellie@eecs.harvard.edu

ABSTRACT

We address the problem of improving the efficiency of natural language text input under degraded conditions (for instance, on PDAs or cell phones or by disabled users) by taking advantage of the informational redundancy in natural language. Previous approaches to this problem have been based on the idea of *prediction* of the text, but these require the user to take overt action to verify or select the system's predictions. We propose taking advantage of the duality between prediction and *compression*. We allow the user to enter text in compressed form, in particular, using a simple stipulated abbreviation method that reduces characters by about 30% yet is simple enough that it can be learned easily and generated relatively fluently. Using statistical language processing techniques, we can decode the abbreviated text with a residual word error rate of about 3%, and we expect that simple adaptive methods can improve this to about 1.5%. Because the system's operation is completely independent from the user's, the overhead from cognitive task switching and attending to the system's actions online is eliminated, opening up the possibility that the compression-based method can achieve text input efficiency improvements where the prediction-based methods have not.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*interaction styles, natural language*; I.2.7 [**Artificial intelligence**]: Natural Language Processing—*language models*

General Terms

Human factors

Keywords

text input, abbreviation, compression, prediction, natural-language processing

IUI'03, January 12–15, 2003, Miami, Florida, USA.
ACM 1-58113-586-6/03/0001.

1. INTRODUCTION

The problem of text input with devices under degraded conditions is not new; disabled users, for instance, have had to interact with computers under sometimes severely degraded means, using mouth sticks, symbol-scanning systems, eye-gaze tracking, and so forth. The problem has renewed currency, however, because of the increased prevalence of small and embedded computing systems (PDAs, cell phones, digital video recorders, and the like) for which traditional text input and verification modalities (keyboard and monitor) are impractical.

Natural language text is highly redundant, inviting the possibility that the redundancies could be used to allow more efficient text entry. The traditional approach to take advantage of this redundancy relies on *prediction* of the user's text. For instance, at each keystroke, the system can predict the most likely future string the user may be typing and allow the user to merely verify the prediction rather than typing all the remaining characters. Alternatively, a set of predictions can be provided, allowing the user to select the correct prediction instead of typing its characters. A paradigm example is the *Reactive Keyboard* of Darragh and Witten [4], though the approach arose as early as the early 1970's. Though intuitive, the idea suffers in practice from severe problems: Because users must take overt action to verify or select, they must be constantly attending to the system's predictions. Typing moves from a fluent, unconscious task to one in which each keystroke requires a significant cognitive load. Previous research [6] has shown that the overheads involved swamp any advantages in speed gained unless the keystroke rate is extremely slow. For this reason, these predictive methods are only useful and have only found acceptance among severely disabled users.[1]

Our approach is based on the duality of prediction and compression [1]. A good statistical model of language, one that can generate good predictions, can inherently be used for compression as well. If we can have the user enter compressed text whose compression is based on a good predictive model, we can then use that model to decode the compressed text into the intended full text. The advantage of the compression approach over the previous prediction ap-

[1]Exceptions that prove the rule include such applications as URL completion in browsers or variable name completion in the emacs editor, which are useful because of the extremely low density and large keystroke count of the "vocabulary", and simple prediction methods used on handheld mobile devices, which amount to abbreviation methods after the user is trained. The method presented here is complementary to these.

proach is clear: The generation of the (compressed) text is not an interactive task that requires task switching, verification of system proposals, selection of options, and so forth. The cognitive load increase is limited to that induced by the ability to fluently generate compressed text.

Because a person must generate the compressed text fluently, we require a *human-centered compression method*. As a *reductio* imagine choosing a standard "computer-centered" method, say, some Lempel-Ziv variant, as used in the standard gzip compression facility. We might expect to obtain a two to one reduction in keystrokes or more, at the cost of requiring a user to compute the Lempel-Ziv compression of the original text mentally, an obvious absurdity. The question arises, then, as to how to devise a human-centered compression method to limit this cognitive load.

As a proof of concept, we devised a human-centered compression method based on a simple stipulated word abbreviation method. A simple stipulated model of abbreviation, that seems relatively well matched to the natural method, is simply to drop all vowels.[2] (We consider "y" a consonant always.) Noting that letters early in the word are most predictive of the remainder, we can retain the first letter even when it is a vowel. (This solves the problem of what to do with words consisting of only a single vowel as well.) Finally, we might allow dropping of consecutive duplicate consonants. Thus, the word "association" would be abbreviated "asctn" under this method, and the sentence "We have conducted some preliminary experiments on the problems of disabbreviation that show the potential for this method." would be abbreviated as "W hv cndctd sm prlmnry exprmnts on th prblms of dsbrvtn tht shw th ptntl fr th mthd." with 38 fewer characters, 31% of the 123 in the original.

2. IMPLEMENTATION

In order to decode text that has been abbreviated in this way, we constructed a statistical model of the abbreviation process as a weighted finite-state transducer [9].[3] The model transduces word sequences, weighted according to a language model, to the corresponding abbreviated character sequence. Viterbi decoding, a standard algorithm for efficiently computing the best path through an automaton, can then be used to reconstruct the maximum likelihood word sequence that would generate a given abbreviated form.

The weighted finite-state transducer technology is well suited to this task in that the model can be composed as a cascade of simpler transducers in an elegant fashion. These include:

1. An n-gram language model (LM). This model was trained on some 1.8 million words of text from the Wall Street Journal using the CMU-Cambridge Statistical Language Modeling Toolkit. [2] Special tokens are inserted for unknown words and numbers. The model was represented as a weighted finite-state automaton.

2. A spelling model (SP). This transducer converts the vocabulary of the language model (words) into the input language of the following transducer (characters). The special tokens are preserved in the transduction.

3. A compression model (CMP). This transducer implements the stipulated abbreviation model, removing the vowels and doubled consonants.

4. An unknowns model (UNK). This transducer replaces the special tokens for unknowns and numbers with sequences of characters or digits, respectively, according to a simple generative model.

The composition of these four transducers forms the entire abbreviation model.

For instance, the string of words "⟨*an*⟩ ⟨*example*⟩ ⟨*of*⟩ ⟨NUM⟩ ⟨*words*⟩" would be successively assigned a probability according to the language model (LM); converted to the sequence of characters "`an␣example␣of␣`⟨NUM⟩`␣words`" (SP); abbreviated to the sequence "`an␣exmpl␣of␣`⟨NUM⟩`␣wrds`" (CMP); and completed by instantiation of the special token ⟨NUM⟩ to, e.g., "`an exmpl of 5 wrds`" (UNK). Through this transduction, then, the model associates the word sequence "⟨*an*⟩ ⟨*example*⟩ ⟨*of*⟩ ⟨NUM⟩ ⟨*words*⟩" as the underlying source for the abbreviation "`an exmpl of 5 wrds`". Of course, other word sequences may be transduced to the same character sequence, for instance, "⟨*an*⟩ ⟨*example*⟩ ⟨*off*⟩ ⟨NUM⟩ ⟨*wards*⟩". The transducer, through the probabilities manifest in the submodels, assigns different probabilities to the various sources of the abbreviated string. Viterbi decoding efficiently selects the maximum likelihood source.

Once the presumed source for the string is computed by this method, the decoded string can be generated by a simple post-process. The special tokens ⟨NUM⟩ and ⟨UNK⟩ are replaced by the corresponding tokens from the abbreviated form, and the capitalization and punctuation found in the abbreviated form are reapplied to the spellings of the source tokens. Thus, the string "`An Exmpl of 5 WRDS.`" decodes as "`An Example of 5 WORDS.`" These extra stages of post-processing could likely be eliminated by extensions to the channel model; the current approach was taken as a simple expedient.

3. EVALUATION

As a first study of the potential effectiveness of this input method, we ran a test to determine the compression ratio (character reduction) of the stipulated abbreviation method, along with the error rate of decoding.

On a small held-out test corpus of some 28,045 characters (5099 words) taken from the Wall Street Journal, this resulted in a compression ratio of 1.36 to 1, or roughly 26.5% reduction in the number of characters. (As a reference upper bound, Lempel-Ziv 77 compression on this corpus provides a 60.4% reduction. Traditional predictive methods, such as ANTIC, ANTICIPATOR, PAL, and, PREDICT, have reported maximal keystroke savings of 20 to 50%. See the discussion by Soede and Foulds [11] and references cited therein.) The error rate was only 3.0%, that is only 155 of the 5,099 words were decoded incorrectly.[4] In addition, a major source of incorrect decoding is repeated out-of-vocabulary items. In

[2]Something like this has been proposed by Tanaka-Ishii [13] for Japanese.

[3]Weighted finite-state transducers constitute a simple general technology for modeling probabilistic string-to-string transformations, which generalize hidden Markov models and other such techniques. Their nice closure properties, especially closure under composition, make them ideal for the present application.

[4]The simpler method of merely dropping all vowels provides a slightly greater compression ratio, 1.41 to 1, or 28.9% character reduction, but the error rate of 4.2% is 40% larger.

Model	*Words incorrectly decoded* (out of 5099)	*Error rate* percent
uniform	2586	50.7
unigram	310	6.1
bigram	177	3.5
trigram	155	3.0

Figure 1: Performance of the disabbreviation method using a variety of language models.

a scenario in which a user is correcting errors on a sentence-by-sentence basis, an adaptive language model should be able to reconstruct later occurrences of words that were initially unknown, providing a further reduction in error rate. We expect that a 1.5% error rate should be achievable in this way.

Processing resources seem potentially practical as well; our unoptimized implementation requires sub-second per word processing times on stock hardware. We expect that more sophisticated implementation techniques should be able to reduce this by an order of magnitude or more.

The benefits of language modeling can be clearly seen by comparing performance against cascades using simpler language models. Figure 1 provides performance of the system under increasingly complex language models, from uniform to unigram, bigram, and trigram. Of particular importance is the improvement of the bi- and trigram models over the unigram model, demonstrating that this approach is likely to have application to any abbreviation method that ignores context, as prior methods do.

The use of cascaded finite-state transducers to build the model allows for a modularity that makes changes to the model, both small adjustments and wholesale modifications, straightforward. For example, simply by replacing the CMP submodel by a model of keypad hashing (which replaces letters with their standard digit equivalent on a phone keypad, that is, the letters 'a', 'b', and 'c' with the digit '2', 'e', 'f', and 'g' with '3', etc.), we generate a keypad dehasher that obtains a 5% error rate on the same test corpus. By inserting the keypad hashing model after CMP, instead of replacing CMP with it, we obtain a system allowing *keypad* input of *abbreviated* text; this obtains an error rate of some 12%.

It should be emphasized that these experiments are quite preliminary. We have made no efforts to address important frailties in the initial implementation, such as limitations in vocabulary, lack of adaptivity, and so forth, which we expect could greatly lower error rate. Alternative stipulated compression models, for instance, ones incorporating a wider range of abbreviation techniques (such as those adduced by Stum and Demasco [12]) would be interesting to pursue.

4. REVIEW OF RELATED RESEARCH

As noted above, text input methods based on predicting what the user is typing have been widely investigated; see the work by Darragh and Witten [4] and references cited therein. Such systems can be found in a variety of tools for the disabled, and some commercial software. Methods based on static lookup in a fixed dictionary of codes for words or phrases include Vanderheiden's Speedkey [14], along with a wide range of commercial keyboard macro tools that require user customization. All rely on the user's memorization of the codes, which must be extensive to provide much compression advantage. Systematic stipulated compression models can be found hidden in stenographic methods such as Speedwriting, though there is no provision for automated decompression.

Some human factors research on the design of command abbreviations for small vocabularies has been performed. John et al. [7], for instance, show that vowel-dropping leads to more easily recalled abbreviations but slower throughput than abbreviations based on escaped special characters. Extrapolation of such results to abbreviation of arbitrary text is problematic, but the results are not inconsistent with the possibility of throughput benefits under reasonable conditions.

Study of the structure of natural abbreviation behavior has been limited: Rowe and Laitinen [10] describe a system for semiautomatic disabbreviation of variable names (such as "tempvar" for "temporary variable") in computer programs, based on their analysis of attested rules for constructing such abbreviations. Stum and Demasco [12] investigate a variety of rules that people seem to use in generating abbreviations, but do not place the rules in a system that allows the kind of automated disabbreviation we are able to perform.

Abbreviation methods at the sentence level include the "compansion" method of Demasco, McCoy, and colleagues [5, 8] and the template approach of Copestake [3]. These techniques, though bearing their own limitations, are fully complementary to the character-based disabbreviation techniques proposed here, and the user interface techniques for error correction developed for our application may be applicable there as well.

5. CONCLUSION

Our approach to reducing the effort for natural-language text input by using abbreviation as a human-centered compression method, rather than prediction, provides a simple method to attain both reasonable keystroke (or equivalent) reduction and reduced task-switching cognitive load. Whether the method provides significant increased throughput (in contrast to most prediction-based methods) awaits user studies that we hope to begin shortly.

This work can be extended in various ways. First, the naturalness of abbreviation might be improved by allowing the user to enter any sort of abbreviation and using a language model trained on a corpus of such naturally abbreviated text for decoding. Second, more sophisticated stipulated abbreviation methods can be tested, which might provide better compression ratios at the cost of learnability and fluency of generation.

6. ACKNOWLEDGEMENTS

The authors are indebted to Winston Cheng, Bryan Choi, and Reggie Harris for their help in implementation of the software described herein. This work was supported in part by grant IRI-9712068 from the National Science Foundation.

7. REFERENCES

[1] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression*. Prentice Hall, Englewood Cliffs, NJ, 1990.

[2] P. Clarkson and R. Rosenfeld. Statistical language modeling using the CMU-Cambridge toolkit. In *Proc. Eurospeech '97*, pages 2707–2710, Rhodes, Greece, 1997.

[3] A. Copestake. Augmented and alternative NLP techniques for augmentative and alternative communication. In *Proceedings of the ACL Workshop on Natural Language Processing for Communication Aids*, pages 37–42, Madrid, 1997. ACL.

[4] J. J. Darragh and I. H. Witten. *The Reactive Keyboard*. Cambridge Series on Human-Computer Interaction. Cambridge University Press, Cambridge, England, 1992.

[5] P. W. Demasco and K. F. McCoy. Generation text from compressed input: An intelligent interface for people with severe motor impairments. *Communications of the ACM*, 35(5):68–78, May 1992.

[6] C. Goodenough-Trepagnier, M. J. Rosen, and B. Galdieri. Word menu reduces communication rate. In *Proceedings of the Ninth Annual Conference on Rehabilitation Technology*, pages 354–356, Minneapolis, MN, June 23-26 1986. RESNA.

[7] B. E. John, P. S. Rosenbloom, and A. Newell. A theory of stimulus-response compatibility applied to human-computer interaction. In *Proceedings of the CHI '85 Conference on Human Factors in Computing Systems*, pages 213–219. ACM Press, 1985.

[8] K. McCoy, P. Demasco, M. Jones, C. Pennington, P. Vanderheyden, and W. Zickus. A communication tool for people with disabilities: Lexical semantics for filling in the pieces. In *Proceedings of ASSETS '94*, Marina del Ray, CA, 1994.

[9] F. C. N. Pereira and M. Riley. Speech recognition by composition of weighted finite automata. In E. Roche and Y. Schabes, editors, *Finite-State Devices for Natural Language Processing*. MIT Press, Cambridge, MA, 1997.

[10] N. C. Rowe and K. Laitinen. Semiautomatic disabbreviation of technical text. *Information Processing and Management*, 31(6):851–857, 1995.

[11] M. Soede and R. A. Foulds. Dilemma of prediction in communication aids and mental load. In *Proceedings of the Ninth Annual Conference on Rehabilitation Technology*, pages 357–359, Minneapolis, MN, June 23–26 1986. RESNA.

[12] G. M. Stum and P. Demasco. Flexible abbreviation expansion. In J. Presperin, editor, *Proceedings of the RESNA International '92 Conference*, pages 371–373, Washington, D.C., 1992. RESNA.

[13] K. Tanaka-Ishii, Y. Inutsuka, and M. Takeichi. Japanese input system with digits: Can Japanese be input only with consonants? In *Proceedings of the Human Language Technology Conference*, San Diego, CA, March 2001.

[14] G. C. Vanderheiden and D. P. Kelso. Comparative analysis of fixed-vocabulary communication acceleration techniques. *Augmentative and Alternative Communication*, 3(4):196–206, 1987.