A Probabilistic Flexible Abbreviation Expansion System for Users With Motor Disabilities

Tim Willis[1], Helen Pain[2], Shari Trewin[3]

[1], [2] ICCS, School of Informatics, University of Edinburgh, Edinburgh, Scotland EH8 9LE, UK
[3], Accessibility Research, IBM T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532, USA
twillis@gmail.com, helen@inf.ed.ac.uk, trewin@us.ibm.com

In this paper we describe the initial design, training and evaluation of a prototype system enabling the automatic and flexible expansion of an abbreviated, typed text input, into a reconstructed sentence. The system’s target user group is cognitively unimpaired users with motor disabilities, for whom typing can be slow and tiring. It is intended that, by reducing the number of keystrokes required to generate a sentence, without imposing a rigid correspondence between abbreviation and full word, their typed communication might be made more comfortable and expansive. The system employs several techniques and statistics, including vowel deletion, phonetic replacement, and word truncation, extracted from two studies of the methods used by people in abbreviation. Encouraging initial results and evaluation are discussed, along with planned future work.


1. INTRODUCTION

Many people with motor disabilities find that communication using a ‘standard’ QWERTY keyboard is tiring and slow, and systems that allow a reduction in the number of keystrokes required, via shortcuts or abbreviations, can help. Flexible Abbreviation Expansion is one approach to reducing keystrokes: users spontaneously generate abbreviations, and the system expands them into full text. We describe here the initial design, training and evaluation of a prototype algorithm that takes an abbreviated, typed text sentence and expands it into a ranked set of candidate full-form sentences. To inform the probabilistic component of this system, two studies were conducted to gather statistics on the techniques used by people in abbreviating text. The results gathered include the relative frequencies for omission of specific letters in specific surrounding letter contexts, and a set of ‘shorthands’ regularly used to truncate words, or sections of them.

The target user group is primarily users with adult-onset conditions and disabilities, including Motor Neurone Disease, spinal injury, and Muscular Dystrophy, who are literate, very familiar with typing and would prefer not to learn a symbol system, and who desire or need an alternative to speech recognition.

2. BACKGROUND

There are a number of Augmentative and Alternative Communication (AAC) systems which expand an input of a few letters or symbols into anything from basic greetings and expressions of need [2], through to fully-formed essays on any topic. One technique commonly used in such systems is Word Prediction, which uses input of the first letters of a word, phrase or sentence, to predict possible completions. Mobile phones generally come with a basic form, known as T9. Limitations of word prediction include a requirement for constant feedback, and commencing from a ‘standing stop’.

One way to expand an entire abbreviated sentence automatically it to use knowledge about how people shorten at the letter level, and about word collocation. For example, it may be assumed that function words are omitted, so these may be re-inserted automatically. An input of Jn wk scl yrdy might expand to John walked to school yesterday. A ‘flexible’ abbreviation expansion system does not require the user to memorise a fixed set of abbreviations, but instead allows them to invent new forms spontaneously. The system itself interprets them, in the context of a sentence, and may offer alternative interpretations where it cannot resolve the ambiguity. One such system is COMPANSION [3], which takes as input a few morphologically-uninflected words, and expands them to fully-formed, grammatically-correct sentences, using selectional restrictions, case frames, and preference semantics, based upon the assumption that the meanings of the words in a sentence are mutually constraining and predictive.
This paper describes how a number of linguistic techniques can be used as the basis of a flexible abbreviation expansion system. Knowledge of which letters are most likely to be omitted from a word, or replaced, is used to reverse-engineer the abbreviation of full-form words, and to rank potential candidates as having been the original form. This offers an alternative to entering entire words and morphemes, allowing potential for saving on keystrokes.

3. A STUDY OF ABBREVIATION TECHNIQUES

In our first study [1], 10 participants, all able-bodied and cognitively unimpaired, were given an imaginary scenario of sending a telegraph message down an expensive line. They were each given a text of 500 characters in length. They were asked to shorten their text into three progressively shorter versions (see 4.1 for details of this). They were given the example shown in Figure 1, where the number of letters is almost halved. Priming the participants in this way was intentionally done to mimic the examples that a flexible abbreviation expansion system would present to its users. They were asked in this study, and the next, not to paraphrase, nor to omit words entirely, because this would increase the already-large search space in the multi-word generation part (see Section 8) of our intended system.

![FIGURE 1: Abbreviation Example Given To Participants](image)

Examination of the results of the study highlighted a number of interesting regularities. Common techniques included taking account of the sounds of words, regardless of whether or not the participant had had any formal phonetic training; a disposition to omission of vowels over consonants, and to omission of letters more often from the end of a word than from the beginning. First letters of words were almost always retained. The exception was initial letters that were vowels, and, more precisely, if the second letter was one of the set ‘l’, ‘m’, ‘n’ and ‘x’. One generalisation about this particular set of sounds is that the first three can all be vocoids – that is, consonants that can stand alone as syllables, in a way otherwise exclusive to vowels. For example, in the second syllables in ‘bottle’, ‘bottom’, and ‘button’. Also, individually they are pronounced beginning with an ‘e’ sound. In these cases the initial vowel was often omitted, e.g. ‘allowing’ as ‘lowng’.

From this study, a ‘general’ set of ‘Replacement Rules’ were generated, showing how participants had used one or more characters to replace longer sequences, for example in ‘fr8’ for ‘freight’, or to make a clear distinction between two possible, confusable words. This was supplemented by filling in gaps suggested by the data, and from the experimenter’s intuition of conventional ‘shorthands’. For example, the abbreviation ‘scr’ could be, amongst other things, ‘saucer’ or ‘soccer’. Replacing the letter ‘c’ with either a soft ‘s’ or hard ‘k’ would clarify which is intended (though would potentially opening up a new set of possibilities). Some further examples are given in Table 1. Building an awareness of these techniques into a system should help in ordering potential expansions, so that the most probable candidates can be given prominence.

<table>
<thead>
<tr>
<th>Edit operation</th>
<th>Situation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>double letter → single</td>
<td>anywhere</td>
<td>tall → tal</td>
</tr>
<tr>
<td>g → O, also ing → n</td>
<td>after ‘in’, i.e. a gerund ‘ing’ becoming ‘in’.</td>
<td>skiing → skiin</td>
</tr>
<tr>
<td>final x of y letters → O</td>
<td>word final</td>
<td>distance → dist</td>
</tr>
<tr>
<td>any letter → O</td>
<td>where it is phonetically silent</td>
<td>cupboard → cuboard, cuburd</td>
</tr>
<tr>
<td>vowel → O</td>
<td>no other rule already applying</td>
<td>tip → tp</td>
</tr>
<tr>
<td>[schwah] → O (vowel deletion)</td>
<td>unstressed vowel (Phonetic)</td>
<td>simile → simle</td>
</tr>
<tr>
<td>w → with (word-specific rule)</td>
<td>word-initially, could be part of longer word. Especially likely in high-frequency words.</td>
<td>with → w (could be part of ‘within’, ‘without’)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Replacement</th>
<th>Situation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch / qu(e) → k</td>
<td>anywhere it is phonetically a [k]</td>
<td>technique → teknike</td>
</tr>
</tbody>
</table>

TABLE 1: Example Deletion and Replacement Rules from First Study

Accessible Design in the Digital World Conference 2005
These rules were hypothesised to represent a set of frequently-used shorthands in general use for abbreviation, where a sequence of one or more characters is used to represent a longer one. Many of these are in standard use in mobile phone text messaging, which is becoming ubiquitous in many countries due to its low cost relative to calls, the fact that its content can be kept more private (and less intrusive) in a public space, and that the receiver can attend to the message at their convenience. Sometimes the inserted characters are different to those in the sequence replaced; at other times the right-hand-side is simply an expansion of the left, encapsulating a conventional shorthand. A set of arbitrary costs were assigned to the rules, intended to reflect their relative likelihood of use, based upon the relative frequency with which they were observed in the data. Due to limitations of time and quantity of data, these are inevitably somewhat subjective, and the set limited.

4. ABBREVIATION TECHNIQUES IN MORE DETAIL

A second study was run to gather a larger and more detailed body of data than the first study, from which to extract statistics on the relative frequencies with which letters, and sequences of letters, were omitted or retained. It was our hypothesis that this information could be used to produce a ranked set of candidate words, and candidate full-form sentences, when presented with an abbreviated sentence.

4.1 Procedure

The second study was run online, on 21 participants, again all able-bodied and cognitively unimpaired. Each was asked to abbreviate a different text of roughly 200 words, taken at random from the written sections of the British National Corpus. The texts were selected to be as general and non-technical as possible, in order to reflect the target area of language for the planned expansion system. All participants abbreviated different texts, with the exception of two who inadvertently used the wrong materials, resulting in two texts being repeated once each.

Each text was abbreviated in three steps, each step progressively shortening it. A slight difference from the first study was that although the same three tasks were undertaken they were in a different order:

1. Concentrate on clarity, using letter-deletion and replacement only where confident it would not compromise comprehensibility and unambiguity for a reader (= long form).
2. Concentrate on maximum brevity, taking the risk of a reader potentially being uncertain, in order to concentrate on losing as many letters as possible (= short form).
3. Find a point the participant judged to be mid-way between the two previous extremes (= medium form).

Instructions included that no word should be removed entirely, though the character(s) remaining need not have been present in the original word, nor should people paraphrase. The same example sentence was given, as in the first study (see above). They were instructed to ignore punctuation, as the expansion system initially does not deal with it, for simplicity. If a task took longer than fifteen minutes, they were instructed to leave it and move onto the next, so that sometimes the number of words differs between tasks.

A set of web pages was built for each participant; the first an introduction and instructions page, then one page for each abbreviation task, and finally ‘thanks’ and links to explanations of the research. A page also asked participants for an indication of how frequently they used SMS (Short Message Service) text messaging on a mobile phone. At the bottom of each task, a text box was provided for them to indicate which techniques they had used, and for any other comments.

Data from two participants was set aside to be used as final test data for the completed system, and was not examined further. A total of 9,798 tokens were generated in total, in the other 19 datasets: the results reported here relate to those 19 datasets.

4.2 Processing of Data

4.2.1 Data Format and Markup

The data from each participant was formatted into a series of rows, each containing an original word followed by the 'long', 'medium' and 'short' abbreviated forms used by the participant. Mark-up was manually added to note the fairly frequent occasions where the participant had either omitted or inserted words (a placemarker token was inserted for blank slots), scrambled their order, or (against instruction) paraphrased. Sentence and clause boundaries, and punctuation, were also marked-up for use by the programs generating statistics from the data.

4.2.2 Sorting Lines Into ‘Ambiguous’/‘Unambiguous’ Mapping Between Abbreviation and Original Word
Next, a program was run to separate the data for each subject into two files, dependent on whether or not all 3 abbreviated forms could be unambiguously ‘matched’ or ‘mapped’ against the original word. Data would be put into the ‘not matched’ file for one of two reasons:

1. One or more of the abbreviated forms contained one or more characters which were absent from the original ‘full’ form they represented. For example:-

   says  sez  sez  sez

2. One or more of the abbreviated forms could not be matched unambiguously to the full form; that is, matching from the left across to the right gave a different set of ‘matching points’ compared to the set from right to left. For example, in the 4th token below, the ‘n’ could match the third, or the fifth letter of the original, and so which should match was ambiguous:

   dining  din  din  dng

Forms also ‘failed’ to match if there was a mismatch of upper and lower case. The intention was to check every mismatch in case it had been intentional; in practice very few had been, and most were due to participants casually swapping case. Quite a number of forms (detailed in Section 6) failed to match due to typos by the participant, either accidental insertion of letters which could not reasonably be interpreted as deliberate, or transposition of adjacent letters.

4.2.3 Generation of Subject-Specific “Replacement Rules” for Ambiguous Mappings
The data in the ‘not matched file’ was examined, and for each subject a set of Replacement Rules which they appeared to have used was constructed. Data from short, medium and long forms were combined.

4.2.4 Generation of Subject-Specific ‘Retention and Deletion Trigrams’ for Unambiguous Mappings
A set of data was generated for each subject and task, from the data that could be unambiguously mapped. Single caret characters were inserted to represent omitted letters. For example:-

   paintings  p^^ntings  p^^nt^ngs  p^^nt^^gs

For every letter, counts were gathered of the surrounding letter contexts in which it had been deleted or retained. To clarify: trigrams were counted, for each letter in these contexts:

- the two letters preceding it
- the two following
- one letter either side

At word boundaries, hashes were inserted, following a linguistic convention from morphology/phonology. Data from neighbouring words was not included. For example, the deletion and retention trigrams for the first letters of the abbreviated form: ##p^^nt^ngs## for ‘paintings’ are given in Figure 2.

<table>
<thead>
<tr>
<th>Letter</th>
<th>Context</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>##p</td>
<td>Add 1 to the counts for letter ‘p’ word-initially, being retained.</td>
</tr>
<tr>
<td></td>
<td>#pa</td>
<td>Add 1 for word-initial ‘p’ followed by ‘a’ being retained.</td>
</tr>
<tr>
<td>pai</td>
<td>Add 1 for ‘p’ followed by ‘ai’ being retained.</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>#pa</td>
<td>Add 1 to the count for letter ‘a’ following word-initial p, being deleted</td>
</tr>
<tr>
<td>pai</td>
<td>Add 1 for ‘a’ between ‘p’ and ‘i’, being deleted</td>
<td></td>
</tr>
<tr>
<td>ain</td>
<td>Add 1 for ‘a’ before ‘in’, being deleted …</td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 2:** “Deletion/Retention Trigrams”

4.2.5 Lower case conversion of all data
For simplicity, the program converted all input, from the tasks and from the dictionary, to lower case. This led to a loss of some information, where candidates had in some cases introduced capitals which had not been in the original text, following one text messaging convention of capitalised letters being read phonetically. An example might be “mEt” for “meet”, to differentiate it from “met”. As capitalising involves at least one extra keystroke, and generally only saves a single letter, it is not thought that the target user group are likely to use this particular mechanism.

5. THE EXPANSION SYSTEM

5.1 Overview

1 Though often intuition would often indicate one over the other. Or it could be hypothesised that the participant was aware of the ambiguity and considered the letter to represent both originals at once.
The expansion algorithm utilises statistics on the likelihood of omission of a letter in a particular context of surrounding letters (letter-level trigrams) along with an enhanced set of ‘replacement rules’, both of which were extracted from the empirical studies. These statistics are used to find and rank candidate expansions for a given abbreviation, based upon the manner in which words would most likely have been abbreviated. We hypothesised that, when presented with two candidate words of similar corpus frequency, which would require a similar number of letters to be inserted into the abbreviation, the system could be trained to give prominence to a word for which the abbreviation is a more ‘natural’ short form.

The algorithm takes an abbreviated sentence and processes each word independently. For each abbreviation, a candidate set of all words in the dictionary that could be expansions is produced. The members of this set are then ranked using three mechanisms:

- insertion costs,
- replacement costs and
- frequency of the candidate word.

It is planned that the expansion system will eventually be able to take an input of an abbreviated sentence and offer the user a set of candidate words for each abbreviated word, although the interface is yet to be designed. We present the results of successively testing the system on the data of each subject, having trained on the other data.

The expansion system currently works through the data, reading a sentence at a time, along with its three corresponding ‘long’, ‘medium’ and ‘short’ abbreviated forms. It then takes each abbreviated word form and generates a candidate set, containing all words in its current dictionary, which could be expansions (i.e. potentially the original word). Our empirical studies showed that participants would almost always preserve the first letter of the original word in their abbreviation, and the expansion program uses this knowledge to constrain the candidate set for an abbreviation (see section 6 for further discussion of exceptions to this rule). For each candidate, a cost is given in terms of the replacement and insertion processes needed to change the abbreviation into it. This method has been successful in spelling checkers [5]. For example, re-insertion of an often-deleted letter, such as a ‘u’ after a ‘q’, would lead to a low cost, whereas re-insertion of a letter that people normally avoid omitting would be a high one. Similarly, a cost is calculated for any replacements used. A maximum of 2 possible replacements per word was imposed. This considerably speeded up the calculations without appearing to cause any of the original words to be missed. The ‘replacement costs’ were subjectively assigned, based upon a sense of how useful they would be. However, letter-insertion costs were based upon the statistics obtained from our studies, in as precise a way as the available data allowed. In cases where precise numbers were unavailable, due to a sequence being absent from the study data, other methods are used to calculate insertion costs (see below).

5.2 “Falling Back” Method for Unseen Data in “Insertion Costs”

The set of letter-level trigrams used by the system is sparse. When calculating insertion costs for trigrams not represented in the data, the system falls back on a set of progressively more general statistics. The system searches for insertion costs using the following algorithm:-

- If defined, use the cost for the specific letter, in the specific context of the two other letters of the sought trigram.
- Else use cost for the specific letter, in the context of vowels/consonants (as appropriate to the context letters)
- Else use cost for a vowel or consonant – whichever applies to the specific letter, in the context of vowels/consonants
- Else use the frequency with which all vowels/consonants - whichever applies to the specific letter – are retained/deleted.

5.3 Calculated vs. Assigned Costs

To speed up the program, candidate words whose length was above a certain multiple of the abbreviation’s length were simply assigned a cost of ‘high’, rather than discarded. They were retained so that we could calculate which other words were being missed entirely. The decision whether to assign a high cost took account of how many ‘segments’ had been replaced: in the following case a non-high cost would still be calculated, because after replacement the candidate does not need many letters inserted.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>full form 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>42n81</td>
<td>for-tu-n-ate-l-y</td>
</tr>
</tbody>
</table>

5.4 Weighting factor for position-within-word of letter insertions

Finally, a small balancing factor is added – insertions that take place to the right of the middle of a word are ‘boosted’ to be cheaper, the further towards the end, the more so; earlier insertions are weighted to be more expensive.

---

2 As our system currently stands, this particular form would actually be missed, since it contains more than 2 replacements. An interim solution is to hard-code certain forms into a single rule for the whole word.

Accessible Design in the Digital World Conference 2005
5.5 Cross-training and testing
The system was cross-trained on the data from 19 participants. Each time, it loaded a set of all ‘Replacement Rules’ and ‘Deletion/Retention-Trigrams’ made up of all the statistics from the other 18. The intention was to simulate the eventual function of the program on new text.

5.6 Output of ranked candidate lists
The output consists of three lists for each input abbreviation/full-form-original pair: the same set of candidate words, ordered firstly by the lowest replacement cost, secondly by the lowest insertion cost, and thirdly, simply by dictionary frequency. The first list was sub-sorted by insertion cost, as many words shared the same replacement cost. Subsorting substantially improved results for the first list, so that in a couple of Top 10 lists (i.e. where the correct word was sought in the highest-ranked Top 10 candidates), it outperformed sorting only by insertion cost. Finally, boosting the most-frequent 60 words to the top of the first list, wherever in it they occur, outperformed all other forms of ranking, beating using frequency alone.

5.7 Generation of the Dictionary
The system uses a dictionary of approximately 10,000 words, which includes detailed information including frequency in the British National Corpus, phonetic and stress transcription, and part(s)-of-speech [4]. This is a reduced version of an original 72,000 word dictionary [12]. The actual number of words in the dictionary is somewhat greater than 10,000, as it includes all words of the same frequency as the ten-thousandth.

6. RESULTS

<table>
<thead>
<tr>
<th>Task</th>
<th>Number of original words abbreviated</th>
<th>No. in dictionary potentially to be found</th>
<th>% of full number found</th>
<th>% of those in dictionary found</th>
<th>% of ‘failed’ forms, fitting pattern (a) or (b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘long’</td>
<td>3263</td>
<td>2915</td>
<td>83%</td>
<td>93%</td>
<td>67%</td>
</tr>
<tr>
<td>‘medium’</td>
<td>3283</td>
<td>2935</td>
<td>82%</td>
<td>91%</td>
<td>70%</td>
</tr>
<tr>
<td>‘short’</td>
<td>3252</td>
<td>2904</td>
<td>79%</td>
<td>88%</td>
<td>53%</td>
</tr>
</tbody>
</table>

Table 2 shows, for each task, the total number of abbreviations produced by 19 participants in the second study, the number of corresponding original words that were in the dictionary, and the number for which the correct expansion was found. For the longer abbreviated forms, the system found 93% of those for which the intended word was in the dictionary. For short forms, 88% were found.

It was observed that first letters of words were sometimes dropped if they were vowels. The algorithm did not account for this phenomenon. Those words that were in the dictionary but not found were analysed, and the final column of Table 2 shows the proportion of these that fitted one or both of the following patterns:

(a) 1st letter of candidate word = vowel and 1st letter of abbreviation = 2nd letter of candidate word
(b) (1st letter of candidate word = vowel) and (1st letter of abbreviation = L,M,N or X, in either upper or lower case). This is predominantly a subset of (a), but with some exceptions, for example ‘md’ for ‘aimed’ (not from data).

Between ½ and 2/3 of abbreviations which failed to match fitted these patterns; 15% of ‘failed’ words could not reasonably be expected to be found by the system, due to either paraphrasing or typos by the subject; 10% were missed due to punctuation and symbols (such as ‘+s’ for ‘adds’ and ‘w’ for ‘with’, which the system does not yet cover); and only 25% (less than 2% of the words present in the dictionary but missed) were down to ‘genuine’ new, unforeseen abbreviation techniques. These included ‘v’ for ‘or’ (using Logical Notation), and ‘hu’ for ‘who’. Many of these common forms such as ‘sez’ for ‘says’ could easily be built into a lookup system.

The numbers of original words, and hence words to be found, differed between tasks as participants stopped at different times in some tasks, and also paraphrased or omitted words in some tasks but not others.

The results in Table 2 were generated using a 10,000 word dictionary. Experiments with 2,000, 20,000 and 30,000 word dictionaries were also performed. As might be expected, with larger ones the percentage of words found rose, but the location of the correct word in the list of candidates went down substantially, to the extent that with 20,000 words, the process of selection of the correct words would probably negate any keystroke saving.

Finding an expansion means that the correct word was included in the set of candidate words, generated by the algorithm, as possible expansions of the abbreviation. This set typically included a dozen candidate words, but could contain as many as several hundred. Table 3 shows the average (mean) ranking of the correct word within the candidate set, for those cases where the correct word was found. The lower this number, the better the

---

3 An input of ‘s’, the most productive abbreviation, matched over 1/5 of the dictionary, most matches being given a ‘high’ cost.
performance. Three ranking mechanisms are compared: firstly, ordering by the sum of any replacement costs incurred in transforming the abbreviation into the full form candidate (lowest first, sub-sorted by insertion, and boosting high-frequency words, see 5.6); secondly, ordering by the sum of any insertion costs (lowest first), and finally, a simple ordering by frequency (highest first). Table 4 shows the extent to which words were abbreviated in each task. As can be seen, participants were able to lose between 25% and as many as 40% of the keystrokes needed for their texts.

<table>
<thead>
<tr>
<th>Table 3: Average Ranking Of Found Words</th>
<th>Table 4: Percentages Of Length Abbreviated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>by replacement</td>
</tr>
<tr>
<td>'long'</td>
<td>4.6</td>
</tr>
<tr>
<td>'medium'</td>
<td>6.5</td>
</tr>
<tr>
<td>'short'</td>
<td>9.5</td>
</tr>
</tbody>
</table>

Tables 5 & 6 show the percentage of time the right word, if found, was in the Top 5 or 10 lowest-cost, the more frequently the better. Sorting by frequency gives the correct word in the Top 5 over 9 out of 10 times for the least-shortened text ('long'), and still manages an impressive 85% for even the most aggressively-abbreviated ('short'). The Top 10 frequency lists contain the right word between 90% & 94% of the time.

<table>
<thead>
<tr>
<th>Table 5: Percentage Of Words Found In Top 5</th>
<th>Table 6: Percentage Of Words Found In Top 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>by replacement</td>
</tr>
<tr>
<td>'long'</td>
<td>96%</td>
</tr>
<tr>
<td>'medium'</td>
<td>94%</td>
</tr>
<tr>
<td>'short'</td>
<td>90%</td>
</tr>
</tbody>
</table>

7. DISCUSSION

When the intended word was present in the dictionary, the abbreviation expansion algorithm was able to identify that word as a candidate expansion for an abbreviation 88-93% of the time, depending on the aggressiveness of the abbreviation. A further improvement of 2-6.5% is potentially possible, if words beginning with vowels are treated according to our empirical observations. Interestingly, this improvement would appear to eliminate the performance difference across the three abbreviation task conditions. Dealing with punctuation and symbols would also help improve this.

Although candidate sets for a given abbreviation were sometimes large, combining the best parts of the different ranking algorithms was successful in bringing the correct word into the top 5 or 10 choices 90-97.5% of the time. Replacement-only-based costs, and some combinations of insertion and replacement costs, simply by addition, were less effective (though combining the two by subsorting gave slightly better results than insertion-only, on the Top 10 results). This may be because the insertion cost data was based on empirical data rather than ad hoc assignment of cost weightings. Insertions may also be a more important source of abbreviation than replacements. For the more aggressive abbreviations, performance was poorer. The 'medium' condition, in which participants balanced brevity with comprehensibility, showed similar (2-3% poorer) performance to the 'long' abbreviations while maintaining a 6% reduction in abbreviation length. This suggests that this may be an appropriate abbreviation level to encourage users to aim for.

We should point out that the savings illustrated do not include the spaces and new lines needed to signal word and sentence boundaries, nor punctuation, though of course these would still need to be present in a system without abbreviation. Our data also suggests that there is a danger that the keystroke savings made (approx 1.5 characters per word) may be eliminated by the actions required to choose from a set of candidate words.

Using these results, a rough estimate can be made of the performance of an abbreviation expansion system employing this algorithm, using an insertion-cost-, or frequency-based ranking scheme (or combination), and offering the user a choice of five possible expansions. The correct expansion would be offered if the word was in the dictionary, the candidate set included the word (80% chance in this data), and the ranking algorithm placed the correct word in the top 5 (96% chance in this data) giving an overall approximate probability of 77% (80% * 96%). Potential improvements are identified below, and a larger dictionary could improve this figure significantly. Considering the minimal nature of the data on which this system is based, these results are encouraging.

8. FURTHER WORK

The results presented here, and experience with the current prototype, have suggested a number of potential improvements to the core expansion system. One of these is the introduction of rules allowing candidate words
beginning with vowels to be assessed using rules (a) and (b) described in the results. This would increase the likelihood of the correct word being found. Different ways of combining the three ranking schemes or calculating costs may also yield improvements, for example, making use of the phonetic, and stressed-syllable information available in the dictionary. Our data would seem to indicate that unstressed vowels are much more likely to be omitted. This could be used to decide between two candidate words where one requires insertion of a stressed vowel and the other an unstressed one. The trigram-retention/deletion statistics may carry some, but definitely not all, of this information.

The process of calculating and assigning 'costs' to each abbreviation-to-full-form transformation is highly computer-intensive. The current implementation requires on average about a minute, on a current mid-range PC, to calculate sets of ranked candidate transformations for a 5-word sentence, using a 10,000 word dictionary. A more efficient implementation could be based on a dictionary of abbreviations, each associated with a set of pre-calculated candidate words and their costs. Each entry in the dictionary could have the form:

<abbreviation> -> [Option1] <word1> <cost1> <frequency1> [Option2] <word2> <cost2> <frequency2>

For example:


Such a system could also adapt to the individual user, promoting recently-used forms and possibly related vocabulary. Training on a user's previously produced text may also enhance performance. The system would probably work best when trained on a corpus of abbreviated and full-form texts for an individual user, along with personalised “Replacement Rules”.

This paper has not discussed the form a user interface to a flexible abbreviation expansion system might take. Our experiments to date involved participants assumed to be similar in cognitive abilities to our intended users, i.e. literate in English and computer usage. Development and evaluation of an appropriate user interface would require direct involvement of the target user group itself. One possible user interface would allow users to type a complete abbreviated sentence, and offer possible expansions for each word. When the user chose an expansion, the choices for the remaining words could be refined, based upon collocational information about likely word sequences. We are currently experimenting to see whether this approach would improve prediction performance.

Promotion of recently-used words, and their semantic cognates, or words commonly found in a small window of surrounding words, may also improve performance.

Ultimately, an expansion system could be built into specific applications such as a web blogging or email form. In this context, a restricted vocabulary might be appropriate, which could improve accuracy or allow more aggressive abbreviations to be used. The system could also be provided as a stand-alone text generation application feeding into other programs on a desktop PC, PDA, mobile phone or other personal electronic device.

9. CONCLUSIONS

We have presented an algorithm for proposing candidate expansions for abbreviated words spontaneously generated by users. The current system is able to put the correct word in the top 5 candidates approximately 77% of the time, based on general texts and a 10,000 word dictionary. Several potential ways to improve this performance have been identified. Ultimately, an algorithm like this could form the basis of a system capable of supporting users who find typing difficult, allowing them to type using spontaneously generated abbreviations instead of full words.

REFERENCES


---

4 This isn’t the whole story: it would depend on how many stressed syllables each word had, word length etc.

Accessible Design in the Digital World Conference 2005


ACKNOWLEDGEMENTS.
The authors are indebted to Jennifer Pedler of Birkbeck College, University of London, for providing the dictionary used in this system, and also to Stephen Clark of Oxford University Computing Laboratory, for generous and substantial contributions. The first author is funded by The Engineering and Physical Sciences Research Council (EPSRC), grant number 99303114.