Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 67 (2015) 105 – 112

Procedia Computer Science

6th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Infoexclusion (DSAI 2015)

Reflective text entry: a simple low effort predictive input method based on flexible abbreviations

Frode Eika Sandnes[a,b]*

[a]*Faculty of Teknology Art and Design, Oslo and Akershus University College of Applied Sciences, Oslo, Norway*
[b]*Faculty of Technology, Westerdals School of Art, Communication and Technology, Oslo, Norway*

Abstract

Users with reduced physical functioning such as ALS patients need more time and effort to operate computers. Most of the previous assistive technologies use prefix based predictive text input algorithms. Prefix based predictive text entry is suitable for languages such as English where the average word length is approximately 5 characters. Other languages such as Norwegian and German have longer mean word lengths as words are combined into longer compound words and prefix approaches are thus less effective. This paper proposes a new abbreviation expansion algorithm. Users mentally determine an abbreviation of the word, typically comprising significant consonants and the system proposes words that contain the matched characters. The approach is non disruptive in that it does not require the user to learn a new system or abbreviation mnemonics, and it can be used with any text input device. The system is dynamic and adapts to the users style of abbreviated input. The algorithm is easier to implement than previous approaches and no a priori system training is required. Our experimental evaluations demonstrate that the algorithm achieves real time performance with modest computer hardware.

Peer-review under responsibility of organizing committee of the 6th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2015)

Keywords: text input; low physical effort; ALS; dyslexia; abbreviation expansion; longest common subsequence

* Corresponding author. Tel.: +47 67 23 50 03.
E-mail address: Frode-Eika.Sandnes@hioa.no

Peer-review under responsibility of organizing committee of the 6th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2015)
doi:10.1016/j.procs.2015.09.254

1. Introduction

The majority of text entry research focuses on text entry speed [1, 2, 3, 4, 5]. For some users the physical effort associated with text entry is a more pressing issue than actual text entry speed. For example, the muscles of ALS (Amyotrophic lateral sclerosis) patients gradually break down and these users eventually often have to rely on gaze-based text input systems which are slow and require high concentration compared to physical QWERTY-keyboards [6, 7]. Users with dyslexia are also known to struggle with text input [8]. To overcome these problems researchers proposed predictive text-entry where the users starts to input the first characters, or prefix of a word, and the system proposes a set of matching words which the user can select from [9, 10, 11, 12, 13, 14]. The user therefore does not have to input the remaining characters. Such systems have recently found their way into mainstream products aimed at the general population. Examples include the Google suggest search engine [15] and mobile phones with reduced keyboards [16].

Predictive text entry was probably first developed for the English language and later adopted into numerous other languages. However, the languages of the world are quite different in nature. For instance basic word units in languages such as Norwegian, Dutch and German are commonly combined to form longer compound words, whereas in English compound words are rare. A problem with long compound words is that one may have to input a long prefix character sequence before the set of unique alternatives becomes sufficiently small. Moreover, prefix based methods require the user to spell the words correctly.

Several researchers have investigated text input based on abbreviation expansions to overcome the problems of prefix-based predictive text input [17, 18, 19]. In this paper this approach is coined reflective text input. Instead of entering a prefix the user inputs a subset of the characters making up the word, and the system determines which words contain these characters in the inputted order from which the user can select. If the subset sequence is unique the intended word can be determined automatically. The user is free to decide which characters to input, but a pedagogical strategy is to input consonants typical of the word. For instance, “pedagogical strategy” in the previous sentence could be input as “pdgcl strtg” reducing the number of entered characters from 20 to 11. The rationale behind the strategy is that some users with motor challenges may have full mental capabilities and may prefer to perform more cognitive intensive tasks requiring less physical effort. The previous work on abbreviation expansion has been either based on linguistic models using phonetic abbreviation rules [17], or large corpus of text which are used to train a Hidden Markov Model [18]. The phonetic approach is language specific and not readily transferable to other languages. Moreover, the phonetically based abbreviation-expansion rules may be incomplete or not match the abbreviation strategies employed by users. The corpus-based methods are based on phrases and will not work as easily on single words or uncommon work combinations. Moreover, corpus-based methods require a priori training of the Hidden Markov Model. The novel approach presented herein improves upon previous approaches. It is language neutral using a word list for the target language. The proposed method does not rely on phrases and is easy to implement, requiring no complicated a priori training.

The remainder of this paper is organized as follows. Section 2 presents the proposed method, Section 3 gives a case study, Section 4 analyses the method and Section 5 concludes the papers.

2. The input method

The algorithm works by catching characters from the keyboard, being it a physical keyboard, software keyboard or some other input system. A sequence of letters is processed as a word once the system encounters a delimiter such as space, enter, or a digit while a list of proposed suggestions is displayed. If the character sequence acquired exists in the dictionary as is, it is interpreted as a fully entered word. If not the sequence is further processed.

The prediction engine is activated once three or more characters are acquired from the user. The entered character sequences are matched against a dictionary with all grammatical forms of the words in the language using the longest common subsequence (LCS) algorithm [20]. The set of words in the dictionary that contain the query subsequence are selected. The resulting list is presented to the user as suggested word alternatives. If the user does not make an explicit choice, but simply inputs a delimiter such as space, the shortest word in the list of suggested alternatives are chosen as it is considered to have the closest match.

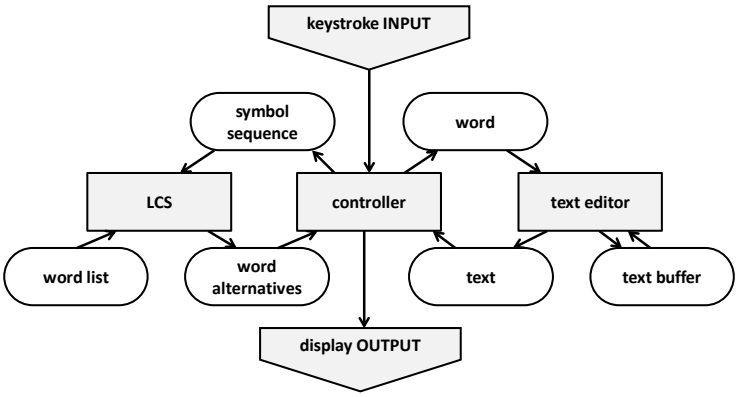

Fig. 1. Text entry system architecture.

Only nine word alternatives will be displayed as these are easily selectable using single numeric keystrokes. If there are more alternatives the word alternatives are filtered according to two criteria. 1) The shortest words and 2) the words with matching prefix. Hence, the algorithm also implicitly provides prefix-based predictive text input.

To limit the amount of computation the dictionary is organized according to word length. The LCS algorithm is thus only applied to words of similar length to the query string and longer. LCS is first applied to strings of equal length and then words of increasing word length until nine candidate alternative suggestions are identified. This allows unnecessary searching of words of shorter length or longer words, which will not be used in subsequent steps. The architecture of the text entry system is illustrated in Fig. 1.

3. LCS-based abbreviation expansion example

The following example illustrates the workings of the reflective text input algorithm. Imagine that the user wants to input the English word “excellent” and selects the dominant consonant characters as an abbreviation “xlnt”. The system will simply echo the first two characters as they are input as “x” and “xl” since two characters are likely to produce too many matches. Once the third character “n” is input the prediction engine is invoked. The string “xln” are then applied to all the words of the dictionary using the LCS algorithm yielding the following 54 words – all with 3 character matches:

extrapolations, explosion, dixieland, extrapolation, extrapolating, excellently, exploiting, explosions, maximilian, exploitation, exaltation, explains, explanatory, exclamation, exhaling, sextillion, explanation, exploding, explain, exemplification, explained, explainer, excluding, exiling, exclusiveness, exemplifying, xylophones, exclusionary, exclamations, excelling, excellent, exoskeleton, expulsion, unexplained, exploring, explainable, explicitness, exalting, exploration, xylophone, explaining, exclusions, explorations, exclaiming, expelling, exclusion, explainers, excellence, excellency, maxwellian, explanations, excellences, exploitations, exultation.

For example, the first word ”extrapolations” has a match in the 2nd, 8th and 2nd to last positions. The string “xln” does not make up a valid prefix and the nine alternatives are therefore selected according to length:

explain, exiling, explains, exhaling, exalting, explosion, dixieland, exploding, explained

Next, the user enters “t” giving “xlnt” and the LCS algorithm return the following 5 words which are all presented as valid alternatives:

excellently, explanatory, explanation, excellent, explanations

a) **_**
[]

b) **her_**
[1:her 2:heri 3:herk 4:hero 5:herr 6:herd 7:heros 8:heron 9:herme ...]

c) her **_**
[]

d) her **falr_**
[1:fallér 2:famler 3:faller 4:falmer 5:falker 6:falder 7:fakler 8:fabler 9:fenalår ...]

e) her faller **ljprn_**
[1:oljeprisen 2:bilkjøperen 3:bilkjøperne 4:oljeprisene 5:oljeprisens 6:miljøprisen 7:miljøparken 8:loddkjøperne 9:oljeprisenes ...]

f) her faller oljeprisen **_**
[]

g) her faller oljeprisen og **krnkrs_**
[1:kronekurs 2:kronekursen 3:kronekursens 4:kjernekraftens 5:omskoleringskurs 6:kjernekraftsaken 7:skrankepersonale 8:skrivekonkurranse 9:elektronmikroskop ...]

h) her faller oljeprisen og kronekursen **_**
[]

Fig. 2. Entering a sentence in Norwegian.

Here, the user could choose the fourth alternative by pressing "4", instead the user enters the space delimiter giving "xlnt ". The system determines that the user intends to input "excellent" since this is the shortest word (9 characters) while the other words have 11 and 12 characters.

Imagine that the user entered a word not in the English dictionary such the Norwegian word "hjelp" meaning "help" in English. The LCS algorithm does not return any words as there are not subsequences with the characters "hjelp" in the English language. The system thus interprets "hjelp" as the intended word.

Finally, imagine that the user enters the four character sequence "over" resulting in a list of 335 words using LCS including:

overtaking, impoverish, overtone, overtook, override, whomever, overhanging, coventry, overpowered, nongovernmental, overcomes, discovered, discoverer, undiscovered, overworks, overcrowding, overviews, contriver, overhauling, overtakers, overcoming, lovelorn, overstating, stopover, overwhelmingly, governance, overtime, overstatement, oversized, …

The list is sorted into increasing word length giving

over, lover, mover, cover, overt, dover, rover, hover, govern, clover, movers, grover, lovers, covert, covers, oliver, overly, solver, prover, hoover, drover, hovers, glover, whoever, solvers, convert, provers, proverb, governs, hovered, andover, olivers, hanover, remover, olivier, forever, recover, covered, overjoy, drovers, cutover, bouvier, uncover, overdue, glovers, overall, poverty, grovers, overuse, however, …

The nine shortest words which prefix matches the inputted string "over" are then displayed to the user, namely:

over, overt, overly, overjoy, overdue, overall, overuse, overthy, oversee, …

The rationale behind this approach that strings making up valid prefixes trumps general reflective query strings. Ellipses (…) are used to indicate that there are more alternatives than the ones displayed.

4. Text entry system example

Fig. 2 shows the system in practice with the input Norwegian phrase "her faller oljeprisen og kronekursen" (here falls the oil-price and crown currency rate). First, the system displays the cursor and empty brackets symbolizes that there currently are no suggested word alternatives (Fig. 2a). Fig. 2b shows the system state after the user has entered "her" (here) giving several alternatives, including "her" as the first alternative. The user simply enters the space character and the word "her" is chosen (see Fig 2c). Next, Fig. 2d shows that the user has entered "falr" as an abbreviation for the word "faller" (falling). This abbreviation comprises the prefix of the word "fal" and the end syllable avoiding the double syllable l. This input gives several alternatives including "faller" which is the third alternative. The user selects the third alternative by pressing the "3"-key. The word "faller" is then expanded and a space is added. The user has thus saved two keystrokes for this six character word.

Fig. 2e shows that the user has entered the abbreviation "ljprsn" for the word "oljeprisen" (oil price). These are all the consonants of the word. The first suggested alternative is the suggested word and it is selected by inputting space (see Fig. 2f). Thus, this 10 character word is entered with just 6 keystrokes.

Next, the user inputs "og " ("and"+space) and the abbreviation "krnkrs" for the word "kronekursen" (crown currency rate) which are the most distinct consonants of the word (see Fig. 2g). There are several alternatives including "kronekursen" appearing as the 2nd alternative which the user selects pressing "2". The word is expanded and a space added (see Fig. 2h). Here, the 11 character word is entered with 6 keystrokes.

5. Performance evaluation

The value of abbreviation expansion has been addressed in previous studies [17, 18, 19] and this initial study does therefore not involve any text input experiments involving users. The objective of the experimental evaluation herein was to identify whether the proposed approach is computationally able to function in real time. The continuous scanning of large dictionaries using the LCS algorithm is a concern in terms of computational efficiency. If the computational effort is beyond a certain level the user will experience a lag. Consequently, if the lag is too long it may not be tolerable and the system will be perceived as unusable [21]. Although, the target group of the system are characterized as slower than average computer users, their cognitive abilities are unaffected and thus expect the same perceived immediate response as other user groups.

The current implementation uses an English word list comprising 45,374 entries and a Norwegian word list comprising 181,731 entries. The words in the English word list are shorter than the many long compound words in the Norwegian word list that also take on various grammatical forms. It was therefore expected that the computational time will differ significantly for the two languages since the word list is scanned for each keystroke entered.

To assess the computation delays an experiment was set up where the full text from a Norwegian news story was used as input. Each word of the story was fed one character at a time into the system and the processing delay and the number of suggested alternatives were recorded.

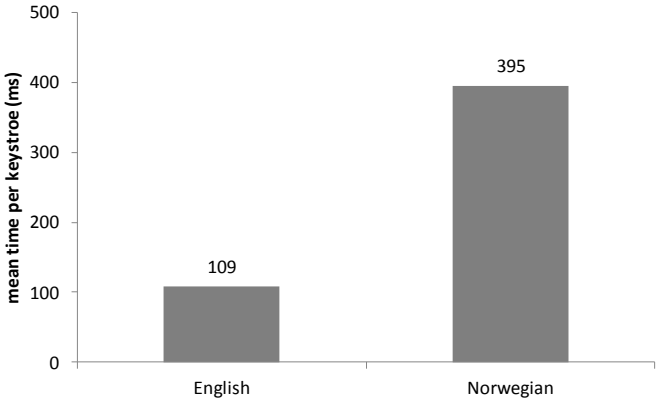

Fig. 3. The mean response time per keystroke shown in milliseconds.

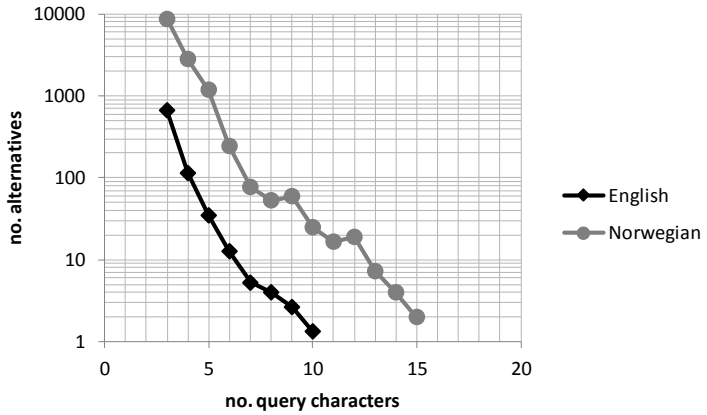

Fig. 4. Mean number of alternatives as a function of number of query characters.

The results confirm difference in processing time for the two languages. Fig. 3 shows the mean response time per keystroke. Note that only keystrokes triggering LCS-searches were included in these means. The mean processing time per keystroke for English was 109 ms (SD=67.4) which was barely noticeable. However, the mean processing time per keystroke for Norwegian was 395 ms (SD=379.5), which was noticeable, yet did not disturb the interaction flow [21]. Several simple implementation optimizations are possible that will greatly reduce the delay.

Next, Fig. 4 shows a log plot of the mean number of alternatives plotted as a function of the number of characters in the query string. One distinct characteristic is that Norwegian has more alternative suggestions than English. For 3 query characters there are a vast number of alternatives, but the number of alternatives quickly decreases as the query strings become longer. English converges at one alternative for words with 10 characters, while Norwegian converges for words with 15 characters. This is as expected due to the longer words in the Norwegian language.

6. Conclusions

An improved abbreviation expansion based text input strategy was proposed. The strategy is intended for users with physical disabilities such as ALS by allowing text to be input with low physical effort. The user enters an abbreviated form of a word and the system determines the most fitting alternatives using a dictionary. Unlike prefix based predictive text input techniques, the proposed strategy allows the user to use characters from the entire word as long as they are input in the correct order. The approach is especially suitable for languages with long compound words such as German and Norwegian where prefix based predictive text input is less effective than for English. The method is termed reflective as it is looking back at the entire word, while prefix based predictive text input is only looking forwards. However, the proposed reflective text input strategy is a more general input strategy for which prefix based predictive text input is a special case. Hence, the approach implicitly also provides the functionality of prefix based predictive text input.

The algorithm is simple to implement by relying on the well known LCS algorithm. The LCS algorithm has a substantial time-complexity. However, the a real time implementation is feasible with today's relatively fast processor and memory capacity and our experiments demonstrate that the algorithm can offer alternatives in real time with large dictionaries and relatively modest hardware without much software optimizations. It should be possible to make huge memory and processing saving by adjusting the algorithm thus making the system suitable for resource constrained devices.

A general challenge with dictionary approaches is that the entered words need to be in the dictionary. However, the proposed method will output most words as is if not in the dictionary. The method is therefore not disruptive to ordinary users. At initial use they will not notice the functionally, but may discover and make use of the functionality over time. It is therefore also possible that the method could be relevant to ordinary users for applications such as software keyboard based mobile text entry. Assistive technologies often tend also to eventually become assets to ordinary users. One could for instance imagine a mobile keyboard with only consonants for simplified text input or some other layout that encourages the use of low frequent characters as low frequent characters are more likely to yield short distinctive abbreviations.

Future work involves testing the input strategy on users to evaluate how well users learn strategies for making abbreviated forms of the word. In particular, experiments are needed to shed light on how users with dyslexia respond to abbreviation based text entry. Additionally, other future work includes the visualization of abbreviations that could accelerate the learning of abbreviations of the words the user is most frequently using. Another improvement is to employ word frequencies to further improve the recommended word alternatives as well as improving he abbreviation expansion according to the users' text entry habits.

References

1. Sandnes FE, Aubert A. Bimanual Text Entry using Game Controllers: Relying on Users' Spatial Familiarity with QWERTY. Interact Comput 2007; 19(2): 140-150.
2. Sandnes FE. Directional bias in scrolling tasks: A study of users' scrolling behaviour using a mobile text-entry strategy. Behav Inform Technol 2008; 27(5): 387-393.
3. Sandnes FE. Can Spatial Mnemonics Accelerate the Learning of Text Input Chords? In: Proceedings of the Working Conference on Advanced Visual Interfaces (AVI 2006), ACM; 2006, p. 245-249.
4. Sandnes FE, Thorkildssen HW, Arvei A, Buverad JO. (2004). Techniques for fast and easy mobile text-entry with three-keys. In: Proceedings of the 37th Annual Hawaii International Conference on System Sciences, IEEE; 2004.
5. Sandnes, FE, Jian HL. Pair-wise variability index: Evaluating the cognitive difficulty of using mobile text entry systems. In: Mobile Human-Computer Interaction-MobileHCI 2004, LNCS 2004; 3160: 347-350.
6. Hutchinson TE, White Jr KP, Martin WN, Reichert KC, Frey LA. Human-computer interaction using eye-gaze input. IEEE T Syst Man Cyb 1989; 19(6): 1527-1534.
7. Majaranta P, Ahola UK, Špakov O. Fast gaze typing with an adjustable dwell time. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM; 2009, p. 357-360.
8. Gregor P, Dickinson A, Macaffer A, Andreasen P. SeeWord: A personal word processing environment for dyslexic computer users. Brit J Educ Technol 2003; 34 (3): 341-355.
9. Garay-Vitoria N, Abascal J. Text prediction systems: a survey. Universal Access in the Information Society 2006; 4(3): 188-203.

10. Darragh JJ, Witten IH, James ML. The reactive keyboard: a predictive typing aid. Computer 1990; 23(11):41–49.
11. Higginbotham D. Evaluation of keystroke savings across five assistive communication technologies. Augmentative Alternative Commun 1992; 8:258-272.
12. Matiasek J, Baroni M, Trost H. FASTY—A multi-lingual approach to text prediction. In: Computers Helping People with Special Needs, LNCS 2002; 2398: 243-250.
13. Beck C, Seisenbacher G, Edelmayer G, Zagler WL. First user test results with the predictive typing system FASTY, In: Computers Helping People with Special Needs, LNCS 2004; 3118: 813-819.
14. Garay-Vitoria N, Abascal J. Intelligent word-prediction to enhance text input rate. In: Proceedings of the intelligent user interfaces 97 congress, ACM; 1997, p. 241-244.
15. Cirasella J. Google Sets®, Google Suggest®, and Google Search History®: Three more tools for the reference librarian's bag of tricks. The Reference Librarian 2007; 48(1): 57-65.
16. Arnott J, Javed M. Probabilistic character disambiguation for reduced keyboards using small text samples. Augmentative and Alternative Communication 1992; 8(3): 215-223.
17. Willis T, Pain H, Trewin S. A probabilistic flexible abbreviation expansion system for users with motor disabilities. In: Proceedings of the 2005 international conference on Accessible Design in the Digital World, British Computer Society; 2005.
18. Pini S, Han S, Wallace DR. Text entry for mobile devices using ad-hoc abbreviation. In: Proceedings of the International Conference on Advanced Visual Interfaces, ACM; 2010, p. 181-188.
19. Han S, Wallace DR, Miller RC. Code completion of multiple keywords from abbreviated input. Automat Softw Eng 2011; 18(3-4): 363-398.
20. Hunt JW, Szymanski TG. A fast algorithm for computing longest common subsequences. Commun ACM 1977; 20(5): 350-353.
21. Seow, SC. Designing and engineering time: the psychology of time perception in software. Addison-Wesley Professional;W 2008.