An Error Tolerant Memory Aid for Reduced Cognitive Load in Number Copying Tasks

Frode Eika Sandnes

Institute of Information Technology, Faculty of Technology, Art and Design
Oslo and Akershus University, College of Applied Sciences, Norway

Frode-Eika.Sandnes@hioa.no

Abstract. Number copying tasks are still common despite increased digitalization of services. Number copying tasks are cognitively and visually demanding, errors are easily introduced and the process is often perceived as laborious. This study proposes an alternative scheme based on dictionary coding that reduces the cognitive load on the user by a factor of five. The strategy has several levels of error detection and error correction characteristics and is easy to implement.

1 Introduction

Number input and number copying tasks are common despite the fact that services are increasingly being digitized. At the very basic level users set their microwave timer or alarm clock. This study focuses on sequential interfaces which rely on numbered keys, for example numeric keypads. One example of number input using numeric keypads includes entering pin-codes to access buildings, or entering pin-codes to finalize debit card transactions in shops.

Most calculators are equipped with numeric keypads and are often complex to use [1]. Making phone calls involves entering digit sequences. Often placing a call involves a number copying task where the number is read from some source such as a phone directory and input using the numeric keypad of the phone [2].

Many commercial software packages require the user to copy printed serial numbers to authenticate installations. Online shops frequently operate with discount codes issued on printed vouchers (see Fig. 1). Airlines frequently operate with booking references printed on the reservation that the users enter on self-service terminals in the airport to obtain their boarding cards (see Fig. 2). Such codes often use higher base numbers such as 36, that is, combinations of letters and numbers or even base 62 with combinations of numbers and upper and lower case letters. Shannon [3] was the first to identify that a decimal digit is approximately equal to 3 1/3 bits. Similarly base 36 and base 62 numbers are equal to 5.2 bits and 6.0 bits, respectively. One may argue that the increased information capacity gained through large base-digits does not justify the increased complexity of the copying tasks from a usability perspective. One reason is that certain letters and digits look similar such as the following digit-letter pairs, 0 - O, 1 - l, 2 - Z, 5 - S, 6 - G and 8 – B (see Fig. 2). Differentiating such sym-

bols is even more difficult if the user has reduced visual acuity and certain fonts are used. When numbers are coded with a high base these digits appear random to the user. The lack of internal structure means that users are unable to resolve ambiguous looking characters. The problem of visually confusing characters is well known in the optical character recognition literature [4].

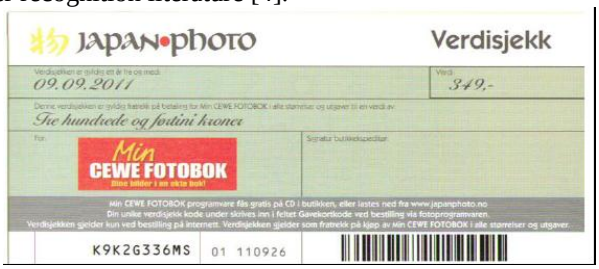

Fig. 1. Special offer voucher with 10 symbol reference code and barcode for barcode readers..

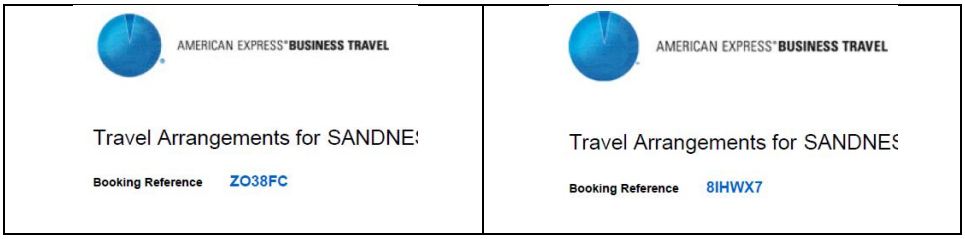

Fig. 2. Booking references with confusing characters: o or zero (left), i or one (right).

In several countries users receive printed invoices with information that are input in order to initiate online transactions [5, 6]. In Norway, a transaction can often comprise the copying of up to 50 digits including the recipients account number, the transaction date, the amount to pay, an invoice identification code and an authorization code obtained from a separate code generating device. Input errors of bank account numbers has led to severe problems for banking customers such as losing large sums of money by incorrect transfers due to input error [5].

Number input in hospitals has also received much attention [7]. For example, intravenous drug administration involves the input of rate, dose, time and volume of drugs and input errors can be lethal for obvious reasons. Studies have shown that error rates in intravenous drug administration can be as high as 50-80% [8, 9]. A study of the frequency of digits in drug infusion pumps showed that 0 is the most frequent digit, followed by the digits 1 and 5, while 3, 4, 6, 7 and 8 are comparatively rare [10]. Numbers ranged in 1 to 5 digits in length with 3 digits being the most common.

It has been found that number input errors are caused by either motor slips where the fingers do not perform as expected, recall slips where numbers are remembered incorrectly, or perception slips where the source number is read incorrectly [11]. These errors can result in digit substitutions, inserting or omissions [11]. Most digit and text input studies operates with typical error rates of 5-10%, which practically means

that up one in every 10 digits are likely to be incorrect. Even worse, one class termed "out by 10 errors" are caused by incorrectly entering a decimal point or a zero [12].

Several studies have looked at various number input interfaces including touch based [13] numeric keypads, displays with individual incremental up-down buttons, incremental left-right/up-down buttons or 5-button interfaces [14] that are often attached to mobile equipment in hospital wards, etc, and results show that the slower incremental up-down interfaces leads to fewer errors than the faster serial numeric keypad [15].

2 Number copying challenges

Number copying tasks are problematic for several reasons. First, copying digits is cognitively demanding. Miller's limit on humans' short term memory of 7+/-2 pieces of information at one time [16] is often cited and subsequent studies have narrowed this limit down to a maximum of 5-7 pieces in short term memory [17]. Studies involving memorizing phone numbers have shown that recall performance degrades rapidly beyond 6 digits [2], and we could therefore chose to operate with the magic number 5 as the upper limit to increase the probability that users with reduced memory are included. A digit copying task therefore has to be split up into several read-input cycles. For example, to copy 25 digits 10 read-input cycles are needed, each cycle requiring shifting the visual attention from the source to the target. The Target is usually a form input field. Matters are complicated further if the source digits are not chunked, that is, strings of digits presented without separators. It is common practice to chunk strings of symbols into groups of say 4-5 symbols separated by blank space. For example, most credit cards numbers comprise 16 digits that are usually presented as four chunks with four digits each. A lack of chunking means that additional cognitive and visual effort is required for users to remember and locate where in the string they were when shifting attention between the source and the target. Another issue that can contribute to the cognitive complexity is a lack of standard form layout. If there is a mismatch in both information sequence and information position at the source and the target it will be cognitively and visually harder for users to pair data from the source to the target.

Second, the digit copying task is error-prone and error rates of 5-10% are not uncommon. Some number sequences contain parity checking symbols such as credit card numbers. Simple parity checking schemes are only able to detect simple errors, and if only one digit is employed there is an approximate 10% chance that an error will go undetected. Many other numbers are not protected by any parity symbols or other error-detection schemes. Also, a lack of chunking will also add to the chances of input error. Errors can also occur if the source numbers are perceived incorrectly.

Third, the digit copying task is perceived as time-consuming and laborious. Many users use inexpensive portable laptop computers. Such portable computers often have a reduced keyboard without a numeric keypad and it has been demonstrated that the input of digits on QWERTY keyboards without numeric keypads are significantly slower than using full keyboards with numeric keypads [6]. The lack of numeric key-

pads is also common on touch based self-service kiosks [18] such as those found in airports and train stations. Moreover, users who master touch typing are usually not able to touch type digits without a numeric keypad.

It is occasionally necessary and practical to communicate numbers orally and each digit will have to be read out one by one and often repeated to confirm the correctness of the data. This further reduces efficiency and increases the chances of error.

Although the ideas presented herein will benefit most users, it is especially useful for users with reduced cognitive functioning. Approximately 10% of the population suffers from some form of reduced cognitive functioning or learning disorders such as dyslexia. Providing better support for number input and number copying tasks is therefore an important part of making computer systems universally accessible.

3 Memory aids

Practice is needed to move information from working memory into long term memory. Unimportant information are filtered by the brain while frequently used information, such our phone number, are gradually moved into long term memory. Students often employ strategies involving memory aids to help remember details for tests and exams. One popular memory aid is to remember number sequences associated with word input on mobile keypad using disambiguating text input [19] such as a four letter word representing the height of a mountain. Instead of recalling the individual digits where each digit counts as one piece of information, the student remembers the word which counts as one piece of information.

Index	Word
00000	a
00001	ex
00002	ai
…	…
99997	desalinizing
99998	preallotting
99999	manipulative

Fig. 3. 5-digit sequence to wordlist mapping.

This paper proposes to use a fixed dictionary on the encoding and decoding sides. The principle is to split sequences of digits into chunks, where each chunk is converted to a linguistic word. The user is presented with a word sequence instead of a number sequence and will thus be able to copy more information per copy-input cycle. Chunk sizes of 5 digits are chosen as this relies on wordlists with 100,000 entries, which can be found in many languages.

For example, take the digit sequence 01234 56789. It is first split into chunks of 5 digits namely 01234 and 56789. Each number is then looked up in the wordlist (see Table 1). The digits 01234 could correspond to the English word “stir” and the digits 56789 correspond to the word “galumphs” meaning portions. Thus, the word se-

quence “stir galumphs” is presented to the user, who has the memory capacity to simultaneously hold both words in working memory while copying the words to the target. Next, the words are identified in the wordlist on the receiving end and the indices of the two words identified, namely 01234 and 56789, respectively.

The word list can be organized such that the magnitude of the number correlates with the word length such that smaller numbers with many prefix zeros are assigned shorter words and larger number are assigned longer words. Another strategy would be to assign the most frequently used numbers shorter words to maximize the effectiveness of communication.

A key advantage of coding digit sequences into linguistic words compared to digit sequences are that experienced readers do not read words character by character, but word by word, while digit sequences are read a digit at a time. The reader recognizes the height signature of text set in lower case letters. This greatly adds to the reading speed. Obviously, the words must be presented in lower case to reap this benefit.

Moreover, the known internal structure of words means that users are manually able to spot simple motor errors. Chances of visual perception errors are reduced since words are read as one unit rather than individual unrelated units.

4 Error detection and correction

The literature on spell checking is large and usually classifies input errors as deletions, insertions or replacements [20]. This strategy proposes three levels of error detection. The first level of error detection catches misspelled words not found in the wordlist. Such mistakes and their whereabouts are easily identified and the location of the misspelled word can be reported back to the user.

To increase the chances of spotting errors the wordlist is organized such that similar words leads are associated with dissimilar digit sequences, and vice versa. The Levenshtein distance is often used to measure the distance between words [21], but it is relatively complex to compute as it allows strings of different lengths to be compared. We therefore use the Hamming distance herein as it is simpler to compute. Moreover, it is only necessary to compare pairs of words with the same lengths. The following word to number assignment is employed. The list of 100,000 words is first sorted according to length. Then words with the same lengths are grouped. The words in each group were next shuffled into random order and assigned running numbers. Finally, the groups were recombined into one list organized according to increasing word lengths while exhibiting a random internal structure. This randomization strategy was simple to realize and will ensure that neighboring words with short hamming distances will have long number distances in most cases.

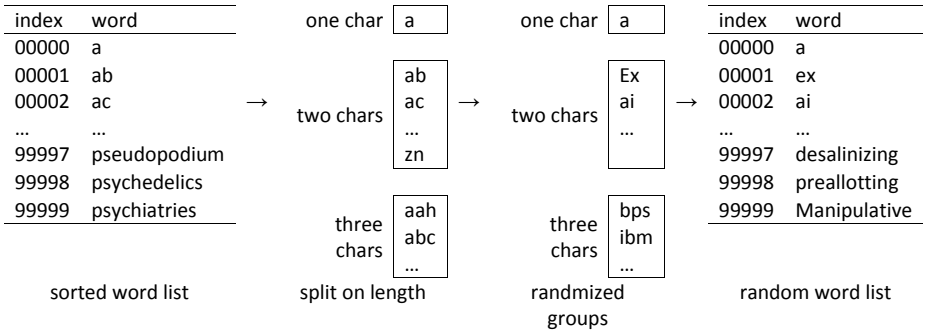

Fig. 4. Composing the error tolerant wordlist.

For example, the word “buys” can easily be mistyped as “buts” by substituting the character y with t by erroneously pressing the neighboring key. The Hamming distance between these words are 1. However, the corresponding numbers for the two words are 1225 and 3022. These numbers have a Hamming distance of 3 and a numeric distance of 1797. If one had used running number instead of random numbers the two numbers for buys and buts would have been 1352 and 1354, respectively, yielding a Hamming distance of 1 and a numeric distance of 2.

A second level of error detection is achieved by introducing a parity check, such as modulus 10 [22]. This parity check can be performed by summing each 5-digit chunk and computing the desired modulus M of the total. An alternative, and perhaps more robust strategy, to computing the sum is to compute a hash value and then take the desired modulus from this hash. If there are any mistakes in one or several of the words then this is likely to be caught since the parity will not match. For example, the misspelled words “form” and “from” that both are valid words in the dictionary are represented by the digits 01825 and 01249, respectively. This difference would flag a modulus-10 parity error. Note also that the words form and from are easily mistaken by humans, but the digits 01825 and 01249 are unmistakable.

A modulus 10 scheme will miss 10% of the errors, while modulus 100 and 1000 will only miss 1% and 0.1% of errors respectively.

A third level of error detection can be introduced computing some modulo of each number chunk and representing the result as sequence of modulus values. The modulus values for each chunk will allow the system to also report the location of the error. The ratio of errors detected depends on the size of the modulo. Thus, 50% of errors are detected with one bit per word, 24% with two bits per word, etc.

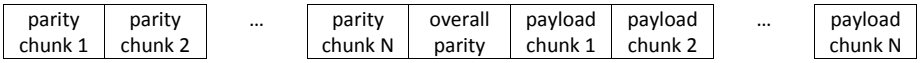

Fig. 5. The overall modulus test detects the presence of errors, the set of individual modulus tests indicates the location of the errors.

The linguistic representation of digits can also assist detecting mistakes when comparing numbers. For instance imagine a customer comparing the dates 23-06-2012 and 23-08-2012, coded as 23062 and 23082, respectively. Each date comprises three information parts, day, month and year. If the user focuses on the day he or she might overlook the difference in month, that is, June versus August since the shape of 6 is similar to 8. However, when comparing the linguistic representations, that is, the words "thrives" and "marxist", it is obvious that the dates are different.

5 Information coding schemes

This section illustrates how various numbers can be fitted to the proposed scheme. It is recommended that various units are fitted to chunk segments such that they do not overlap. This allows various information parts to be assessed independently of each other.

Large numbers with more than 5 digits are simply split into several chunks. For example, Norwegian account numbers comprise 11 digits. Such numbers can be presented as two 5 digit numbers by discarding the 11th digit which is a parity check digit that can be omitted and recalculated at the receiving end since the proposed scheme has a more powerful error-detection mechanism than this simple modulus-11 parity check. Consequently, an account number can be represented using two-word phrases, which is easier to work with than 11 digit numbers. Invoice identification codes represent another example. Invoice identification codes, which in Norway can consists of 2 to 25 digits, are simply split into as many 5-digit chunks as required.

Some number sequences are longer than 5-digits, but can easily be reduced to 5-digit chunks without much loss of generality, such as dates. Dates usually indicate the day, month and year suing eight digits. An alternative 5-digit representation is to specify the day and month using four digits and one digit for the year. This will work as most applications operate within a limited scope of time – usually less than a year. This scheme will work for with a 10-year window. However, if a longer time window is needed, the day and month can be reduced to a 3-digit format using the day of year, which is less readable. However, such a scheme will allow to digits for the year giving a time window of 100 years. The two date formats illustrated allows dates to be specified using a single linguistic word which is useful when making comparisons. Comparisons across years are simplified further if the year is dropped altogether.

Numbers comprising of 5 digits or more are simply represented using heading zeros such as small quantities, prices, drug doses, etc.

6 Implementation

An English wordlist published by the SIL International Linguistics Department was used as starting point. This wordlist contains 109,582 entries with lengths ranging from 1 to 28 characters. The entries were sorted according to increasing word length and the 100,000 shortest words were kept and the other words discarded. The average length of the remaining words is 8 with the longest words being 12 characters long.

The size of the final wordlist is less than 1 Mb. Words of equal length were shuffled into random order to increase the distances between number chunks.

A proof of concept coder and decoder were implemented in a Microsoft Excel. Excel was chosen in order to prove the simplicity of the approach demonstrating that it is simple to implement irrespective of platform.

The strategy can be illustrated using one of the flight reference number shown in Fig. 2, namely ZO38FC, which is hard to perceive because of the o/0 ambiguity, hard to remember and hard to input as it involves a mixture of numeric and letter keys. It is assumed that this reference uses base 36 comprising letters and digits with the rightmost symbol being the least significant. When converted to decimals this becomes the 10 digit number 2218923661 which is split into two chunks, namely 22189 and 23661. These numbers correspond to the two linguistic words "relets sheaves" which are easier to read, remember and type. This example does not employ any explicit party checks.

Another example is the payment of a TV-license invoice issued twice a year by the Norwegian Broadcasting Corporation issued to each household with a television set. The details printed on the invoice are as follows:

Accout no.	70410542100
Date	31 Jul, 2012
Amount	1290,06
Invoice ID	500330570102919

This is first converted to the digit sequence with the first chunk used for error correction with N=1 and M=10 giving

01206 70410 54210 31072 01291 50033 05701 02919

which would have to be copied in eight cycles if done manually. This sequence gives the following eight words

vive islanding sockeyes hautboy pint limberer puled hors

This sequence is easily remembered in two chunks, that is "vive islanding sockeyes" and "hautboy pint limberer puled hors". When this phrase is input to the decoder the original invoice information is detected successfully.

Imagine that one of the words in incorrectly spelled, for instance "viwe" instead of "vive". The mistake and the location of this mistake would easily be detected as this word is not present in the dictionary. Next, imagine we misspell "hors" as the valid word "horn". This mistake is both detected and located since both the modulus-10 and modulus-2 tests fire. Instead, imagine the word "hors" is misspelled as "hops". This mistake is detected as the modulus-10 test fires, while the location is not detected since hops is a valid word and the modulus 2 test does not fire. Finally, imagine that we misspell "hors" as the valid word "hers" – this mistake is not detected with the following setup as the parity tests pass despite the error.

Only a simple linguistic parity word is used in the example. The error rate would be very robust by detecting nearly every error if error correction is compromised for error detection by using the entire parity word with a modulus 100,000 test. If error correction is more important the number of bits per chunks can be improved, doubling its capacity with each bit added. By dropping the overall modulus check and using to bits per chunk one could survive with one parity word while doubling the capability to detect the location of the errors.

7 Conclusions

A memory aid for simplifying manual number copying tasks is proposed. The strategy converts the digits sequences to sequence of linguistic words, reducing the number of read-input cycles by a factor of five, meaning that certain transactions can be remembered in one go. In addition the strategy can be used as an alternative modality allowing users to easily verify the correctness of information as it is easier to compare linguistic words than number sequences. The approach is capable of detecting and correcting multiple errors and is more powerful than simple parity symbol methods. Errors can be detected and corrected at word level as incorrectly spelled words or through two levels of parity checks allowing the position of the error to be reported to the user. The wordlist is constructed to ensure sufficiently large linguistic word distances between neighboring number sequences increasing the chances of detecting errors. The approach is applicable to all areas where users have to copy number sequences into a computer or electronic device.

References

1. Thimbleby, H. (2000). Calculators are needlessly bad, International Journal of Human-Computer Studies 52 (6), pp. 1031–1069.
2. Raanaas, R. K., Nordby, K. and Magnussen, S. (2002). The expanding telephone number Part 1: Keying briefly presented multiple-digit numbers, Behaviour & Information Technology 21 (1), 27-38.
3. Shannon, C. E. (2001). A mathematical theory of communication. SIGMOBILE Mobile Computing and Communication Review 5 (1), 3-55.
4. Kahan, S., Pavlidis, T. and Baird, H. S. (1987). On the Recognition of Printed Characters of Any Font and Size. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-9 (2), pp. 274 – 288.
5. Olsen, K. (2008). The $100000 Keying Error. Computer 41 (4), 108, 106-107.
6. Sandnes, F. E. (2010). Effects of common keyboard layouts on physical effort: Implications for kiosks and Internet banking. In Proceedings of Unitech 2010. Trondheim: Tapir Akademisk Forlag, pp. 91–100.
7. Oladimeji, P. (2012). Towards safer number entry in interactive medical systems. In Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems (EICS '12), ACM, New York, NY, USA, pp. 329-332.
8. Barber, N., and Taxis, K. (2004). Incidence and severity of intravenous drug errors in a german hospital. European Journal of Clinical Pharmacology 59 (11), 815–817.

9. Taxis, K., and Barber, N. (2003). Ethnographic study of incidence and severity of intravenous drug errors. BMJ 326 (7391), 684. CHECK PP
10. Wiseman, S. (2011). Digit distributions, What digits are really being used in hospitals? Proceedings of the Fourth York Doctoral Symposium on Computer Science, The University of York, pp 61-68.
11. Wiseman, S., Cairns, P. and Cox, A. (2011). A taxonomy of number entry error. In Proceedings of the 25th BCS Conference on Human-Computer Interaction (BCS-HCI '11). British Computer Society, Swinton, UK, UK, pp. 187-196.
12. Thimbleby, H. and Cairns, P. (2010). Reducing number entry errors: solving a widespread, serious problem, Journal of the Royal Society Interface 7 (51), 1429-1439.
13. Isokoski, P. and Koki, M. (2002). Comparison of two touchpad-based methods for numeric entry. In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '02). ACM, New York, NY, USA, pp. 25-32.
14. Cauchi, A. (2012). Differential Formal Analysis: Evaluating safer 5-key number entry user interface designs, EICS'12 Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems, 2012, pp. 317-320.
15. Oladimeji, P., Thimbleby. H. and Cox, A. (2011). Number Entry Interfaces and Their Effects on Error Detection, Lecture Notes in Computer Science 6949, pp 178-185.
16. Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review, 63, 81–97.
17. Simon, H. A. (1974). How Big Is a Chunk? Science 183 (4124), 482-488.
18. Sandnes, F. E., Jian, H.-L., Huang, Y.-P. and Huang, Y.-M. (2010). User Interface Design for Public Kiosks: An Evaluation of the Taiwan High Speed Rail Ticket Vending Machine, Journal of Information Science and Engineering 26 (1), 307-321.
19. Sandnes, F. E., Thorkildssen, H. W., Arvei, A, Buverud, J. O. (2003). Techniques for fast and easy mobile text-entry with three keys, In Proceedings Norsk Informatikkonferanse 2003, Trondheim: Tapir Utrykk, pp. 205-216.
20. Kukich, K. (1992). Techniques for automatically correcting words in text. ACM Computing Surveys, 24(4), 377–437.
21. Navarro, G. (2001). A guided tour to approximate string matching. ACM Computing Surveys 33 (1), 31–88.
22. Wagner, N. R. and Putter, P. S. (1989). Error Detection Decimal Digits, Communications of the ACM, 32(1), 106-110.