HMS: A Predictive Text Entry Method Using Bigrams

Jon Hasselgren **Erik Montnemery** **Pierre Nugues** **Markus Svensson**
Lund Institute of Technology
Department of Computer Science
Box 118
S-221 00 Lund, Sweden
{d99jh, d99em, d99msv}@efd.lth.se, Pierre.Nugues@cs.lth.se

Abstract

Due to the emergence of SMS messages, the significance of effective text entry on limited-size keyboards has increased. In this paper, we describe and discuss a new method to enter text more efficiently using a mobile telephone keyboard. This method, which we called HMS, predicts words from a sequence of keystrokes using a dictionary and a function combining bigram frequencies and word length.

We implemented the HMS text entry method on a software-simulated mobile telephone keyboard and we compared it to a widely available commercial system. We trained the language model on a corpus of Swedish news and we evaluated the method. Although the training corpus does not reflect the language used in SMS messages, the results show a decrease by 7 to 13 percent in the number of keystrokes needed to enter a text. These figures are very encouraging even though the implementation can be optimized in several ways. The HMS text entry method can easily be transferred to other languages.

1 Introduction

The entry of text in computer applications has traditionally been carried out using a 102-key keyboard. These keyboards allow to input characters in a completely unambiguous way using single keys or sometimes key combinations.

However, in the last few years, mobile telephones have introduced a new demand for text entry methods. Mobile telephones are usually optimized in size and weight. As a result, the keyboard is reduced to a minimal 12-button keyboard (Figure 1).

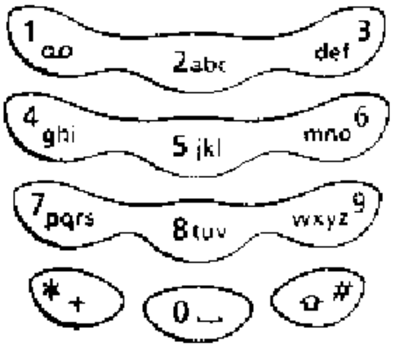


Figure 1: The 12-button keyboard of a Nokia 3410.

The reduced keyboard makes it hard for the user to enter text in an efficient way because s/he has to use multiple tapping or long key combinations to display and disambiguate the characters. Albeit tedious, the multiple tapping method was the most commonly implemented in mobile telephones until some time ago. To spare the user these elements of frustration, a new class of text entry methods has appeared. It uses dictionaries in an attempt to resolve the word ambiguity and requires, in most cases, only one keystroke per character.

This paper proposes a method that supplements the dictionary with word and bigram probabilities. The method uses the last written word to improve the prediction of the current word and to decrease the number of needed keystrokes even further. This method that we refer to as HMS in the rest of the text, uses the frequencies of common bigrams that we extracted from a corpus of texts.

2 Current Text Entry Methods

In this section, we summarize the text entry methods currently in use and some methods under development. All the mentioned methods use a keyboard with 12 buttons.

As a measurement of the efficiency of the different text entry methods, we will use the number of keystrokes per character or $KSPC$ (MacKenzie, 2002). A completely unambiguous keyboard enables a $KSPC$ of 1, text prediction methods may reduce this number even further.

2.1 Multi-Press Methods

The multi-press methods require more than one keystroke to enter a character. These methods allows for unambigous typing of characters. They can be used alone or as a fallback for systems using more complex text entry methods. The multi-press methods are well suited to type words not contained in the dictionary.

2.1.1 The Multi-Tap Method

The first and still most common way to enter text on a mobile telephone is the multi-tap method. Since 'a', 'b' and 'c' share the same key, the user presses it once to enter an 'a', twice to enter a 'b', and three times to enter a 'c'. To enter the word *dog*, the user presses the sequence of keys "36664".

As two consecutive characters of a word can share a same key, as for example the word "no" where both 'n' and 'o' are assigned to 6, a timeout is needed to determine when to stop shifting the letters and display a new character.

This method results in a $KSPC$ of 2.0342 if English text is entered (MacKenzie, 2002).

2.1.2 Remapped Keyboard

On current mobile telephone keyboards, characters are assigned alphabetically to keys. This is not optimal given that, for instance, the most frequent character in English, 'e', is displayed using two taps. Remapped keyboards assign a single key to the most frequent characters. The remaining characters are grouped into sets that share a same key. This method decreases the $KSPC$ because frequent characters are entered with only one keystroke.

The program MessagEase (Saied, 2001) of EXideas uses the idea of the remapped keyboard technique. MessagEase results in a $KSPC$ at 1.8210 (MacKenzie, 2002).

2.2 Single-Press Methods

The single-press methods try to reduce the $KSPC$ to roughly one. They resort to a dictionary as a means of resolving the ambiguity of the input.

2.2.1 The Predictive Text Entry Method

With the predictive text entry method, the user presses one key per character and the program matches the key sequence to words in a dictionary (Haestrup, 2001). In that sense, although its name suggests otherwise, this method merely aims at disambiguating the words rather than predicting them. Even if several characters are mapped to the same key, in many cases, only one word is possible given the sequence. This method makes it possible to reduce the $KSPC$ to roughly 1. If the key sequence corresponds to two or more words, the user can browse through the resulting word list and choose the word s/he intended to write.

The user, for example, enters the word *come*, by first pressing 2. The program will then propose the word *a* because it matches the entered sequence. When the user presses 6, 6, and 3, the program might propose the words *an*, *con* and finally *come*. The words *bone*, *bond*, and *anod* (and some more), also fit the given sequence. The user can access these words by pressing a next-key.

Many new mobile telephones use this method. The most widely used implementation is T9 by Tegic (Grover et al., 1998). Other implementations are eZiText by Zi Corporation (Zi Corporation, 2002) and iTAP by Motorola (Lexicus Division, 2002). Most implementations only match words with the same length as the key sequence, resulting in a $KSPC$ of slightly greater than 1 when the user types words that are contained in the dictionary.

Some implementations propose words longer than the tapped sequence based on probability information for the words. These implementations can reach a $KSPC < 1$.

2.2.2 WordWise

WordWise developed by Eatoni Ergonomics uses an auxiliary key. A character on a key is selected explicitly by simultaneously pressing the key corresponding to the character and the auxiliary key indicating the position of the character on the key. This decreases the number of matching words for a key sequence considerably because the user explicitly disambiguates some characters in the sequence.

A drawback is that two keys must be pressed concurrently. With a limited space keyboard, this can prove difficult to some users.

2.2.3 LetterWise

LetterWise (MacKenzie et al., 2001), also by Eatoni Ergonomics, is a different approach, which eliminates the need for a large dictionary. It only considers the letter digram probabilities. In English, the letter 't' is often followed by 'h' and hardly ever by 'g'. The program selects the most probable letter knowing the previous one. The user can browse and change the characters by pressing a 'Next' key.

The LetterWise method has a $KSPC$ of 1.1500 (MacKenzie, 2002). One of its main advantages is the small amount of memory needed. Another advantage is the fact that it is just as easy to enter words, which are not in a dictionary. Therefore this could be a suitable fallback method instead of the multi-tap methods, to produce faster text input.

3 Predictive Text Entry Using Bigrams

Prediction may further improve the performance of text entry with a limited keyboard. With it, the suggested words may be longer than the currently typed input.

We propose to use word bigrams, i.e. two consecutive words, to give a better text prediction, see *inter alia* (Shannon, 1948), (Jelinek, 1997), and (Manning and Schütze, 1999). The list of bigrams is stored in memory together with their frequency of occurrence and it is accessed simultaneously with the character input.

Given a previously written word, the most probable subsequent words are extracted from the bigram list. Using the maximum of likelihood, the probability of the bigram w_{n-1}, w_n given the word w_{n-1} is computed as:

$$P_{MLE}(w_n|w_{n-1}) = \frac{C(w_{n-1}, w_n)}{C(w_{n-1})} \quad (1)$$

Since the previously written word w_{n-1} is always known and constant, it is sufficient to use the frequency of the bigrams and set aside $C(w_{n-1})$.

In practice, bigrams must be combined with a dictionary. Sparse data from the development corpus and memory constrains make it impossible to store an exhaustive list of bigrams. To choose the words to propose, we used a variation of the Katz model (Katz, 1987). The Katz model takes the longest available N-gram and uses correction terms to normalize the probabilities. In the case of bigrams, the probabilities can be expressed as:

$$\begin{gathered} P(w_n|w_{n-1}) \\ = \\ \begin{cases} P(w_n|w_{n-1}) & \text{if } C(w_{n-1}, w_n) \neq 0 \\ \alpha P(w_n) & \text{if } C(w_{n-1}, w_n) = 0 \end{cases} \end{gathered} \quad (2)$$

where α is the correction term.

In our implementation, the bigrams are always prioritized over the unigrams. The Katz back-off model is well suited for our implementation as it allows for a small memory footprint of the bigrams list, while still ensuring that the system will support entering of all words in the dictionary.

In addition to the bigram frequencies, the word length is a useful criterion to present the matching words to the user. This additional parameter is justified by the navigation through a list of words with the keys available on mobile telephones.

Bigram probabilities used alone produce a list of possible words and rank them without regard to the effort needed to select the intended word. Since browsing the list is carried out using one scrolling key, it may take a couple of keystrokes to reach the word. Even, if corpus frequencies suggest a longer word being preferred to a shorter one, a presentation by decreasing frequencies may be inadequate.

The list navigation is in fact easier in some cases using character input keys. A single keystroke can resolve a great deal of ambiguity because there is a total of 8 keys to choose compared to the unique scrolling key to cycle the list of suggested words. That's why the list of proposed words is rescored and short words are given an additional weight.

4 Implementation

We implemented a software prototype of the HMS method we described in this paper. We chose the Java programming language because of its extensive packages that allow for rapid development. Another advantage is Java's platform independence, which should, in theory, make it possible to run the program on any modern mobile telephone.

The program was designed to run on a handheld device i.e. on the client side of the mobile network. The memory of a mobile telephone is very limited and a disadvantage of this strategy is the memory footprint of the language models we use. A possible workaround would be to implement the HMS software on an application server. All the users would then share the language models with possible customizations. Modern mobile telephone infrastructures enable a real-time round trip of the typed characters and thus the interactive suggestion of matching words.

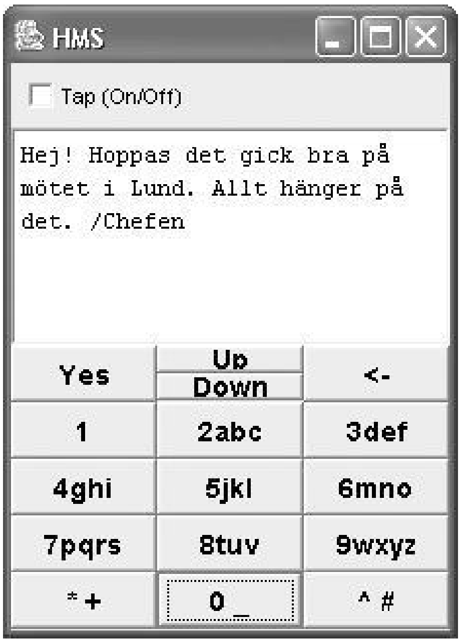
The program computes a list of word suggestions every time a key is pressed and the best suggestion is displayed simultaneously on the screen: The top white window in our Java program (Figure 2). The user can browse the list of suggestions using the up and down keys.

4.1 Program Design

The program is divided into two parts: a user interaction module and a lexical database module.

The user interaction module currently consists of a Graphical User Interface (GUI) whose layout closely resembles that of a mobile telephone. The simulated keyboard layout makes it possible to compare the HMS prototype with software running on mobile telephones.

The lexical database module contains the core of the program. It is responsible for the generation of a list of suggested words given the user input so far. The modules communicate with each other using an interface. Thus, the two parts are independent and one may modify the user interaction module in particular to fit different platforms without having to modify the module concerning the word guessing algorithm.



☐ Tap (On/Off)

```
Hej! Hoppas det gick bra på
mötet i Lund. Allt hänger på
det. /Chefen
```

Figure 2: Screenshot of the HMS Java prototype.

4.2 Data Structures

A compact encoding structure of the bigram and unigram lists has a significant impact to achieve an efficient word proposal.

The data structure we used is comparable to that of a letter tree or *trie* (de la Briandais, 1959). However, the nodes of the new tree structure correspond to an input key instead of a character as in the classical tries. For instance, the characters $(a, b, c, 2)$ are associated to a single node. Thus, the tree structure enables to represent the keystroke ambiguity and makes it easier to traverse the tree. It also introduces the need to store a complete list of words that match a keystroke sequence in the leaves resulting in a somewhat higher memory overhead.

Searching this type of tree is straightforward. The keys pressed so far by the user are used as input and the tree is traversed one level down based on every key pressed. When the traversal is completed the resulting sub-tree includes all possible suggested words for the typed key combination.

For the bigrams, a slightly different structure is needed. Since the previously written word has been chosen from the list of suggested words, it can no longer be considered ambiguous. One cannot simply build a tree of bigrams using the proposed structure because the tree itself is ambiguous. A collection of trees, one tree for each preceding word, was used. For performance reasons, a hash table was used to manage the collection.

4.3 Training the Language Model

We used a dictionary of 118,000 inflected Swedish words and we trained the language model – unigrams and bigrams – on the Stockholm-Umeå (SU) Corpus (Ejerhed et al., 1992). The SU corpus is a POS annotated, balanced corpus of Swedish news reports, novels, etc. To keep the memory size of the program under 100 megabytes, we retained only 195,000 bigrams from the corpus. The SU corpus does not reflect the language of SMS messages that differs greatly from that of the "classical" written Swedish. This results in a non-optimal language model. We chose it because of the unavailability of a large-enough public SMS corpus.

When the input of a single word is completed, its corresponding bigram and unigram probabilities are updated. It results in a learning system, which adapts to every user's style of writing. To increase the speed of adaptation, language frequencies derived from the user input have higher priorities than what has been learned from the training corpus. If the system were implemented as a server application, the personal adaptation would become more complex. However, a separate table with user customized parameters can be saved either locally or on the server.

All corpora and dictionaries used with the software have been in Swedish so far. However, the HMS program does not carry out any language-specific parsing or semantic analysis. Hence, the method could be transferred to any language provided that a sufficient corpus exists.

5 Evaluation

As an evaluation of the efficiency of our implementation, we made an initial comparative test between the HMS program and the Nokia 3410, which uses the T9 system.

As we said in the previous section, we could not train a language model optimized for an SMS application. This certainly biased the evaluation of the entry methods in our disfavor. Therefore, we chose to evaluate both programs with a test set consisting of a sample of SMS messages and short texts from newspapers.

A total of nine testers entered the texts. They first had the possibility to get accustomed to both the HMS and the T9 methods. The testers were encouraged to compose a short arbitrary SMS message of 50-100 characters containing everyday language. They also chose an excerpt of a newspaper article of approximately the same length as the typed SMS message from the *Aftonbladet* Swedish newspaper website. The keystroke count was recorded and used to calculate the $KSPC$ parameter.

The entry of new words, i.e. missing from the dictionary, uses the same technique in the HMS and T9 methods. If the text selected by a tester contained words not present in at least one of the dictionaries, the tester was asked to choose a new text.

Table 1 shows the results we obtained in keystrokes per character.

Table 1: Test results.

Method	Type of text	$KSPC$
T9	SMS	1.0806
HMS Bigrams	SMS	1.0108
T9	News	1.0088
HMS Bigrams	News	0.8807

In this relatively limited evaluation the HMS entry method shows a $KSPC$ smaller than that of the T9 system in both tests: news and SMS texts. The improvement is of, respectively, 7 and 13 percent. The better result for the bigram method is mainly due to two reasons. First, the utilization of the previously written word to predict the next word results in an improvement of the prediction compared to the methods relying only on dictionaries such as T9. Secondly, the fact that words are actually predicted before all characters are entered improves even further the performance of HMS over T9.

6 Discussion

The difference in $KSPC$ between the SMS and news text with our method is to a large extent due to the corpus, which does not fit the more casual language of the typical SMS texts. The T9 method, on the other hand, is optimized for typing SMS texts.

Another reason that may contribute to the observed difference is that the news texts in general contain longer words. The mean word length in our test is about 4 characters for the SMS texts and 5 characters for the news texts. At a given state of a word input, a few more keystrokes, often one or two, reduce dramatically the selection ambiguity and the identification of a longer word will need less keystrokes in proportion to its length than a shorter one. This corresponds a smaller $KSPC$ for longer words. Figure 3 shows the $KSPC$ according to the word length and the falling curve for longer words.

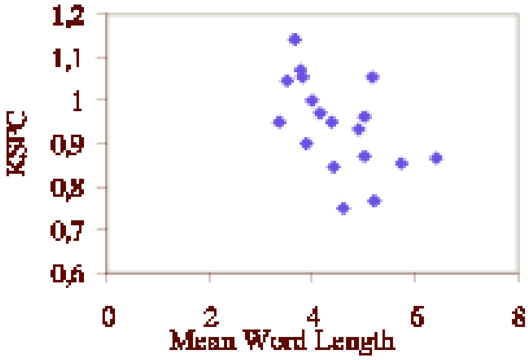


Figure 3: $KSPC$ versus the mean word length in the HMS bigram method.

Ambiguity is also reduced for longer words when the length of the key sequence is exactly that of the word. The possible words for a given longer key sequence are often fewer than for a shorter one. This explains why the T9 system also shows a result better for the news text than for the SMS messages. However, the T9 can never reach a $KSPC$ less than 1 since it doesn't predict words longer than the given sequence.

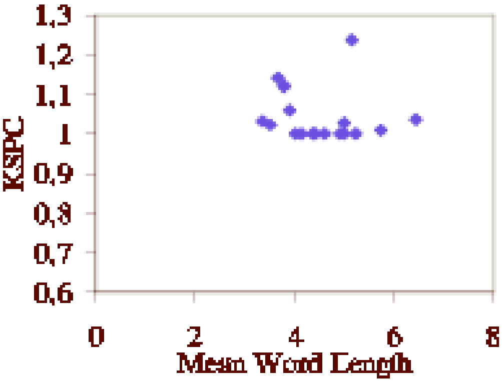


Figure 4: $KSPC$ versus mean word length in the T9 system.

Other significant differences between the SMS and news texts play a role in the final results. For example, the SMS texts show a higher frequency of certain characters such as the question marks, slashes and exclamation marks, which results in a higher $KSPC$. This fact can explain the surprisingly high $KSPC$ for some texts. This property affects both methods to the same extent though.

7 Conclusion and Perspectives

We implemented a new text entry method adapted to mobile telephone keyboards and we compared it to the T9 method widely available on commercial devices. The HMS method is based on language models that we trained on the SU corpus.

The training corpus was, to a great extent, collected from Swedish news wires and didn't fit our application very well. This is heavily related to the language used in SMS messages, which tends to include abbreviations and slang absent from the SU corpus. However, the results we obtained with the HMS method show a decrease by 7 to 13 percent in the number of keystrokes needed to enter a text. These figures are very encouraging even though the implementation can be optimized in several ways.

Further evaluation could be conducted with an automatic test system. It would produce results that would eliminate all input mistakes. It would also enable to compute the optimal solution in terms of $KSPC$ given the two options left to the user at a given point of the word entry: either type a new letter key or scroll the list of possible words.

It would also be very interesting to evaluate the $KSPC$ of the bigram method after training the system with a better-suited corpus. We expect the $KSPC$ to be significantly lower than with the present corpus. It is worth once again pointing out that even with the non-optimal corpus, the results of the bigram method are on par or superior.

We also observed that the language model adapts quicker to the users' individual ways of expressing themselves than other systems. It thus increases the gain over time.

At the time we wrote this paper, we could not gain access to a large corpus of SMS messages. However, we intend to collect texts from Internet chat rooms and message boards, where the language shows strong similarities to SMS language. We expect a better language model and an improved $KSPC$ from this new corpus.

A problem with the bigram method is its large memory footprint compared to that of dictionary-based systems. However, this should not be a problem on the next generation of mobile telephones like GPRS and 3G. The language models could be off-loaded on an application server and the low round-trip time of the network system should enable a real-time interaction between the server and the user terminal to carry out the word selection.

References

Zi Corporation. 2002. eZiText. Technical report, http://www.zicorp.com.

R. de la Briandais. 1959. File searching using variable length keys. In *Proceedings of the Western Joint Computer Conference*, volume 15, pages 285–298, New York. Institute of Radio Engineers.

Lexicus Division. 2002. iTap. Technical report, Motorola, http://www.motorola.com/lexicus, December.

Eva Ejerhed, Gunnel Källgren, Ola Wennstedt, and Magnus Åström. 1992. The linguistic annotation system of the Stockholm-Umeå project. Technical report, University of Umeå, Department of General Linguistics.

Dale L. Grover, Martin T. King, and Clifford A. Kushler. 1998. Reduced keyboard disambiguating computer. U.S. Patent no. 5,818,437.

Jan Haestrup. 2001. Communication terminal having a predictive editor application. U.S. Patent no. 6,223,059.

Frederick Jelinek. 1997. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts.

Slava M. Katz. 1987. Estimation of probabilities from sparse data for a language model component of a speech recognizer. *IEEE Transaction on Acoustics, Speech, and Signal Processing*, 35(3):400–401, March.

I. Scott MacKenzie, Hedy Kober, Derek Smith, Terry Jones, and Eugene Skepner. 2001. Letterwise: Prefix-based disambiguation for mobile text input. Technical report, Eatoni, http://www.eatoni.com/research/lw-mt.pdf.

I. Scott MacKenzie. 2002. KSPC (keystrokes per character) as a characteristic of text entry techniques. In *Proceedings of the Fourth International Symposium on Human Computer Interaction with Mobile Devices*, pages 195–210, Heidelberg, Germany. Springer-Verlag.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Language Processing*. MIT Press, Cambridge, Massachusetts.

Nesbat B Saied. 2001. Fast, full text entry using a physical or virtual 12-button keypad. Technical report, EXideas, http://www.exideas.com/ME/whitepaper.pdf.

Claude E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July-October.