

Improved Word List Ordering for Text Entry on Ambiguous Keypads

Jun Gong
Google Inc.
1600 Amphitheatre Pkwy.
Mountain View, CA USA 94043
1-650-214-5957
jungong@google.com

Peter Tarasewich
Sawyer Business School, Suffolk Univ.
8 Ashburton Place
Boston, MA, USA 02108
1-617-994-6841
tarase@suffolk.edu

I. Scott MacKenzie
York University
4700 Keele St.
Toronto, Ontario, Canada M3J 1P3
1-416-736-2100 ext. 40631
mack@cse.yorku.ca

ABSTRACT

We present a design methodology for small ambiguous keypads, where input often produces a list of candidate words for a given desired word. The methodology uses context through semantic relatedness and a part-of-speech language model to improve the order of candidate words and, thus, reduce the overall number of keystrokes per character entered. Simulations yield improvements in text entry speed of about 10% and reductions in errors of about 20%, depending on the keypad size. We describe a user study with 32 participants entering text on a keypad with letters arranged on three keys. Entry speed was 9.6% faster, and error rates 21.2% lower, compared with standard disambiguation, as found on mobile phones.

Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation]: User Interfaces – *Input devices and strategies*.

General Terms

Measurement, Performance, Design, Experimentation, Human Factors, Standardization, Languages, Theory, Verification.

Keywords

Mobile devices, text entry, keypad, disambiguation, prediction

1 INTRODUCTION

Handheld mobile devices vary widely in their text entry interfaces. Devices like the *Blackberry* have miniaturized full keyboards. PDAs often use a stylus with a virtual keyboard. While some mobile phones have integrated full keyboards, most employ a 12-key keypad (Figure 1). The result is ambiguous when used for text entry because multiple letters reside on each key. Therefore, mobile phone text entry requires more than one keystroke on average for each character entered [18].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

NordiCHI 2008: Using Bridges, 18-22 October, Lund, Sweden
Copyright 2008 ACM ISBN 978-1-59593-704-9. \$5.00

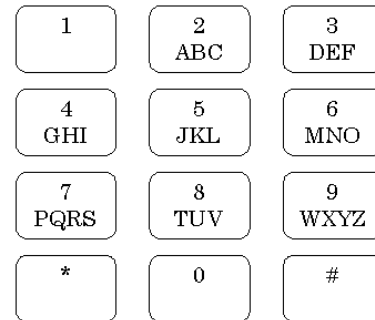


Figure 1: Standard 12-key mobile phone keypad. [22]

Dictionary-based predictive disambiguation requires one initial keystroke per character entered. Stored linguistic knowledge then determines candidate words for a given key sequence. For instance, 8TUV-4GHI-3DEF might produce "the" from the possible letter combinations. Where collisions occur, a rank-ordered list is produced. For example, 3DEF-6MNO-4GHI matches "dog" and "fog". List ordering occurs through a combination of word or n -gram (sequence of n characters) frequency lists, user preferences, or prior word usage. Examples of commercial dictionary-based predictive disambiguation text entry systems are *T9* by Tegic, *iTap* by Motorola, and *eZiText* by Zi Corp. Improved word list ordering – the goal of this research – can reduce the number of extra keystrokes for selecting a desired word, potentially improving text entry performance. The improvement will be more pronounced on very small keyboards (e.g., 3 or 4 keys) since the word lists are often large. Devices with very few buttons are often used for text entry in scenarios including text entry for disabled users, text entry with game controllers, and text entry with wearable computing devices. These are our primary use contexts.

Previous research sought to improve predictive disambiguation using, for example, character-level digrams (two-letter sequences) [17], grammatical and linguistic information [25], and a commonsense knowledge base [29].

In this paper, we present an improved predictive disambiguation method using a novel combination of semantic relatedness and a part-of-speech language model. The result is better ordering of candidate words through contextual information. Simulations using various sizes of keypads show that the likelihood of the desired word appearing at the top of the list is significantly

improved, and that the average number of extra keystrokes to cycle through a word list is reduced.

Existing mobile devices have relatively limited hardware resources, such as processing power and memory. Therefore a large language model is problematic and of limited practical use. However, our language model requires limited memory and provides real-time processing capabilities, resulting in good disambiguation performance. After running simulations that showed expected performance improvements with our method, a prototype system was implemented on a Dell Pocket PC for practical study. Results of a usability experiment confirm these expected improvements in text entry performance.

2 BACKGROUND

Research in predictive disambiguation text entry started with the work of Shannon in 1950s [27]. In 1976, Rabiner [24] analyzed the ambiguity problem of the telephone keypad to enter text. Later research looked at improving keypad designs and disambiguation algorithms. Levine et al. [15] described a technique with a reduced computer keyboard containing more than one letter on a single key. The technique determined which letter was the most likely using both an English dictionary and letter trigram statistics. The authors also used a genetic algorithm to alter the keyboard layout to minimize word ambiguity. Two text entry methods were compared. One used a standard QWERTY keyboard and another used predictive disambiguation text entry with a reduced keyboard. Results showed that although predictive disambiguation text entry required greater mental load, keyboard designs that put multiple characters on a single key still offered advantages, such as minimizing the size of the device as well as the physical movement required by motor-impaired users.

When an ambiguous key sequence is entered, predictive disambiguation algorithms determine the most likely desired word. Most algorithms choose the word with the highest frequency of occurrence among matching words based on a given corpus. Both trigram- and digram-based disambiguation have proven effective at the character or word level.

While disambiguation works on a sequence of keystrokes mapped to a word or portion of a word, *word prediction and completion methods* use “forecasting” to add characters to a word stem to form a complete word. However, both disambiguation and prediction use similar techniques, as Masui [21] demonstrated in a pen-based text entry system using word prediction. Word prediction and completion are often used in alternative and augmentative communication (AAC) to reduce the keystrokes required by motor impaired users. Boggess [2] experimented with two simple prediction algorithms. Results showed that for a single user’s vocabulary, a prediction window of 50 words yielded about a 50% success rate in predicting the next word. When the first letter of the next word was available, a prediction window of 20 words succeeded about 82% of the time. There are also alternatives to statistical models, such as Stocky et al.’s system using semantic information derived from a “commonsense knowledge base.” [29]

The use of context is promising. There are at least two possibilities: *physical context*, such as location, time, or weather, and *textual context*, referring to the surrounding text. Physical context, while potentially useful, is rather difficult to implement.

As one example, Dominowska et al. [6] used contextual information to enhance a text communication system for disabled users by adjusting vocabularies based on user location. This was implemented, however, using preprogrammed context data rather than actual sensor data.

On the other hand, previous research suggests that better prediction is possible using textual context. Recent work by Budanitsky [4] surveyed five WordNet-based measures of lexical semantic relatedness. In Manning and Schutze [20], *semantic relatedness* is defined as entities that are likely to co-occur. Therefore, instead of using the real meanings of words, a simpler model might just use word co-occurrence data in a large corpus.

Li and Hirst [16] proposed an n -gram model combined with a semantic knowledge model to predict future words that were more appropriate than those generated by the n -gram model alone. Algorithms using text co-occurrence information were proposed for finding the semantic relatedness between English word pairs.

Various Part-of-Speech (POS) models are widely used by researchers in Natural Language Processing (NLP). There are many approaches to automatic POS tagging, such as *Stochastic Tagging*, which can incorporate frequency, statistics, or probability information. Simple stochastic tagging may be based on the probability of the occurrence of a given sequence of tags. This is like an n -gram/Hidden Markov model, where the best tag is determined by the n previous tags [20, 31].

NLP research has been applied to text entry, especially with mobile devices, for quite some time. As noted in Kukich’s 1992 survey [13], bi-gram language models, built both on words and on word POS, are useful for correcting errors in user-entered text. However, the POS models described were based solely on bi-gram information, possibly missing other valuable syntactic relationships farther away. In comparison, our language model captures relationships between words that reside far apart from each other.

Researchers from Microsoft recently proposed an n -gram language model for text entry with soft keyboards [12]. However, the model is based on characters, and therefore has limited applicability to word-level text entry systems using dictionary-based prediction.

NLP techniques are also used in industry for improving keypad text entry. Patented systems, such as those developed by Ameritech and Tegic [5, 9], all employ NLP methods for disambiguating user input. A common drawback is that the models deployed are usually built at the character-level, since a word-level language model simply would not fit in the available memory of a mobile device. Our work, by comparison, uses word-level semantic and syntactic language data to capture relationships between non-adjointing words in a model that requires realistic amounts of memory given the constraints of mobile devices.

3 PREDICTIVE DISAMBIGUATION WITH SMALL KEYPADS

Standard dictionary-based disambiguation uses word frequency information to decide the order of a list of matching words. The goal is to minimize the keystrokes for inputting more frequently used words.

Given the results of many research studies on the problem of word prediction and/or disambiguation [e.g. 29, 21, 16], it is reasonable to hypothesize that better word list ordering is achievable by considering previously entered text as contextual information. The highest payoffs are expected for keypad designs with very few keys combined with contextually aware predictive disambiguation [11].

Finding good text entry methods for keypads with very few keys remains a significant challenge for HCI researchers, although progress continues. Some recent advances include a 5-key watch-top interface from Dunlop [7] and Wobbrock et al.'s *EdgeWrite* [31], as well as other methods [e.g., 1, 8]. Sandnes et al. [26] reviews existing text entry methods on devices with only a few keys (usually 3 to 4). Such devices are particularly useful for users with special needs (e.g., those with motor or visual impairments). Examples of existing small devices with very few keys or limited interaction space are shown in Figure 2.



Figure 2. Devices with very few keys or buttons. The Dunlop [7] watch interface is shown in the lower left.

3.1 Semantic Relatedness of Word Pairs

N -gram models have been used by text entry or natural language processing systems to capture relationships between different words [20]. However, such models are ill-suited for mobile devices because of the storage space and computational power they require [28]. For example, in the sentence “The dog was really sick and barked all night”, the word “dog” is a good contextual word for disambiguating “barked”. To capture this relationship, a word level n -gram model of length six is necessary; however, this exceeds the memory limitations of most mobile devices.

To solve this problem, we used a co-occurrence-based semantic relatedness model to replace the more limited word n -gram model. The model is a modification of Li and Hirst's semantic relatedness model [16] and takes the following form:

$$SEM(w_1 | w_2) = \frac{C(Stem(w_1), Stem(w_2))}{C(Stem(w_2))}$$

where w_1 and w_2 are any two words in the dictionary. $Stem(w_1)$ and $Stem(w_2)$ are the word stems of w_1 and w_2 . A word stem is the basic form of a word plus any derivational morphemes (i.e., those prefixes and suffixes that change the meaning and/or part-of-speech of a word, such as “un” or “less”), but excluding inflectional elements (i.e., suffixes that only change the function of a word, such as “s” and “ly”). Word stems used here are

derived by a stemming algorithm [23], and therefore may not be exact. $C(Stem(w_2))$ is the number of times the stem of w_2 occurs in the training corpus. $C(Stem(w_1), Stem(w_2))$ is the number of times the stems of w_1 and w_2 occur in the same defined contexts in the training corpus. For this research, two words in the same sentence are defined as being in the same context. The choice of a sentence as the unit of context is based on results from two experiments using sentences and paragraphs as two forms of context. Finally, $SEM(w_1 | w_2)$ denotes the co-occurrence based semantic relatedness of word w_1 after seeing w_2 in the context.

Note that Li and Hirst's semantic relatedness model [16] is symmetric, meaning that $SEM(w_1 | w_2)$ equals $SEM(w_2 | w_1)$. Our model is asymmetric since it is conditional-probability-based. Also note that the semantic relatedness model is built on word stems instead of the words themselves.

3.2 Word Part-of-Speech Validity

To reduce the number of POS tags in the algorithm and minimize computation, we used 19 closely related POS tags from the British National Corpus (BNC) tag set. These were used for tagging the BNC itself.

The POS validity of a word, based on words preceding it, is defined as follows:

$$\begin{aligned} POS(w_i | S = t_0 w_1 w_2 \dots w_{i-1}) \\ &= \max_{1 \leq j \leq n} \{\delta_j(t_j)\} \\ &= \max_{1 \leq j \leq n} \{\max_{1 \leq k \leq n} \{\delta_{i-1}(t_k) \times P(t_j | t_k) \times P(w_i | t_j)\}\} \end{aligned}$$

Which uses the definitions provided earlier, except that $t_i \in \{t_1 \dots t_n\}$ now refers to one of the 19 POS groupings instead of individual POS tags. Furthermore, $POS(w_i | S = t_0 w_1 w_2 \dots w_{i-1})$ denotes the POS validity of a word w_i with the context of the preceding input sentence $S = t_0 w_1 w_2 \dots w_{i-1}$.

The POS validity of word w_i is defined as follows. Assume $S = t_0 w_1 w_2 \dots w_{i-1}$ is entered and w_i is a candidate word derived from the current keystroke sequence. According to the Viterbi algorithm, we calculate n paths from the sentence delimiter to each t_j of the n possible POS categories of word w_i . The scores of these n paths are $\delta_j(t_1)$ to $\delta_j(t_n)$, which represent the probabilities of the most likely tagging for the sentence $S = t_0 w_1 w_2 \dots w_{i-1} w_i$ such that w_i has the POS tags t_1 to t_n . However, for dictionary-based predictive disambiguation, the primary concern is not which POS tag is the most likely tag for word w_i . Instead, we want to know, in the case of word w_i being the user-desired word, the POS validity for w_i in the i^{th} place in the current sentence. Therefore, where w_i is the user desired word, the greatest probability among $\delta_j(t_1)$ to $\delta_j(t_n)$ is chosen to represent the POS validity for w_i . This follows the definition of word POS validity given above.

In the following two sections, our context-aware predictive disambiguation algorithm is described. The objective is to improve the ordering of candidate words using semantic and syntactic information in the sentence context. This is followed with the results of a simulation (Section 6) and a user study (Section 7).

4 ALGORITHM DESCRIPTION

We begin with a few definitions:

w_i – a word in the dictionary

ks_i – the keystroke encoding of w_i

$match(ks_i) = \{w_1^{w_i}, w_2^{w_i}, \dots, w_n^{w_i}\}$ – the set of words sharing the same keystroke sequence with w_i . Note that $w_i \in match(ks_i)$.

$w_j^{w_i} \in match(ks_i)$ – a word that shares the same keystroke encoding with w_i .

$Freq(w_j^{w_i})$ – the normalized frequency of $w_j^{w_i}$ among all the words in the set of $match(ks_i)$. Note that

$$\sum_{w_j^{w_i} \in match(ks_i)} Freq(w_j^{w_i}) = 1$$

If w_i is not ambiguous, $match(ks_i)$ has one element $w_1^{w_i} = w_i$, therefore $Freq(w_1^{w_i}) = 1$.

The inputs are the keystroke encoding ks_i of word w_i and sequence H_i of context words. H_i contains all the input words preceding w_i and in the same sentence. The output is a list of English words w_1, w_2, \dots, w_m ordered so that the most probable list is given by ks_i and H_i .

First, we define the validity of word w_i given its history H_i of context words with the following components:

1. The estimated semantic validity $SEMV(w_i | H_i)$ of w_i given H_i :

$$SEMV(w_i | H_i) = \sum_{w \in H_i} SEM(w_i, w)$$

2. The normalized estimated semantic validity $NSEMV(w_i | H_i)$ of w_i given H_i :

$$NSEMV(w_i | H_i) = \frac{SEMV(w_i | H_i)}{\sum_{w_j \in match(ks_i)} SEMV(w_j | H_i)}$$

3. The estimated POS validity $POSV(w_i | H_i)$ of w_i given H_i :

$$POSV(w_i | H_i) = POS(w_i | H_i)$$

4. The normalized estimated POS validity $NPOSV(w_i | H_i)$ of w_i given H_i :

$$NPOSV(w_i | H_i) = \frac{POSV(w_i | H_i)}{\sum_{w_j \in match(ks_i)} POSV(w_j | H_i)}$$

Therefore, the estimated validity $EV(w_i | H_i)$ of w_i given H_i is the linear combination of its normalized semantic validity, its normalized POS validity, and its normalized frequency:

$$EV(w_i | H_i) = \alpha \times Freq(w_i) + \beta \times NSEMV(w_i | H_i) + \gamma \times NPOSV(w_i | H_i)$$

where α , β , and γ (with values between 0 and 1) are coefficients specifying how certain we are in the frequency, semantic, or POS validities.

Given the definitions above, context-aware predictive disambiguation is straightforward. It first finds the list of matching words, ks_i , and the estimated validity of each matching word. Next, it sorts the list by validity values and returns the sorted list. This is shown as pseudocode in Figure 3.

```

Disambiguate ( $ks_i, H_i$ )
   $m_i = match(ks_i)$ 
  For each  $w_k \in m_i$ , find  $EV(w_k | H_i)$ 
  Sort  $m_i$  based on  $EV(w_k | H_i)$ , and return sorted  $m_i$ 
End Disambiguate
  
```

Figure 3. Pseudocode for the context-aware predictive disambiguation method.

5 ALGORITHM COEFFICIENTS

An important step is to linearly combine the frequency, semantic relatedness, and POS validities of each matching candidate word into overall estimated validities, so that all the candidate words can be properly sorted. This section describes the method to find the optimal values for the linear combination coefficients α , β , and γ .

5.1 The Equivalent Geometry Problem

Assume the current ambiguous keystroke sequence is ks_i . Then

$$match(ks_i) = \{w_1, w_2, \dots, w_l\}$$

is the list of l matching words (among which w_k is the desired word), and

$$\{Freq(w_1), \dots, Freq(w_l)\}, \{SEMV(w_1), \dots, SEMV(w_l)\},$$

and

$$\{POSV(w_1), \dots, POSV(w_l)\}$$

are the frequency, semantic relatedness, and POS validities for each matching word.

To make w_k the first disambiguation choice, we pick any α , β , γ such that for any $1 \leq i \leq l$ and $i \neq k$:

$$\begin{aligned} &\alpha \times (Freq(w_k) - Freq(w_i)) + \\ &\beta \times (SEMV(w_k) - SEMV(w_i)) + \\ &\gamma \times (POSV(w_k) - POSV(w_i)) > 0 \end{aligned}$$

If we consider

$$(Freq(w_k) - Freq(w_i), SEMV(w_k) - SEMV(w_i), POSV(w_k) - POSV(w_i))$$

as $l - 1$ data points in a 3-dimensional space corresponding to the $l - 1$ matching but undesired words, the above equation is simply the basic geometry equation with all the $l - 1$ data points in the positive plane of a surface determined by the normal vector (α, β, γ) . For a testing corpus of many ambiguous keystroke sequences, each is disambiguated to matching candidate words for the data points in the 3D space. Using these data points, the problem is to find a surface (α, β, γ) that contains the greatest number of data points in the positive plane of this surface.

5.2 Solving the Coefficient Optimization Problem

With a transformed but equivalent problem, finding the best coefficients for linearly combining estimated validities is much easier. It is known that any surface containing the origin can be determined by specifying two angles θ and φ , such that

$$\alpha = \text{Cos}(\theta)\text{Sin}(\varphi)$$

$$\beta = \text{Cos}(\theta)\text{Cos}(\varphi)$$

$$\gamma = \text{Sin}(\theta)$$

Therefore, the optimization algorithm simply collects the data points and progressively varies θ and φ from 0° to 90° in steps of 1° . The parameters α , β , and γ are chosen based on the best angles reported; that is, those that place the most data points in the positive plane.

6 SIMULATION

To test our context-aware predictive disambiguation method, computer simulations were conducted. We compared the performance of our method against a standard (unmodified) predictive disambiguation method, similar to the methods implemented on many current cell phones.

6.1 Corpus

We used the BNC Baby corpus [3] for these simulations. This is a four million word sampling of the entire British National Corpus (BNC). BNC Baby was chosen because of its XML formatting and because it is entirely POS tagged. This provides a solid foundation to build our semantic relatedness and POS models while providing the convenience of having XML and POS information available. Because of the relatively small size of this corpus, the predictive disambiguation performance might further improve if larger text corpora were used in the future.

6.2 Simulation Setup

The simulation used a ten-fold cross validation design. The corpus was randomly divided into ten equally sized data sets. For every set of simulation settings (e.g., models used, keypad size), the simulation was repeated ten times, so that each data set could be used as a testing set. The frequency, semantic relatedness, and POS models were trained using the remaining nine data sets. Using this design, the effect of biased training and testing data sets was minimized. The results of the ten runs were evaluated statistically to check the significance of any differences.

Pair-wise relationships between words in the same sentence were all captured. However, the number of tokens is no larger than a

typical word level bi-gram model. The sizes of the semantic and syntactic language models were approximately 28MB and 700KB, respectively, which are practical for current mobile devices.

Simulations were also run for different sized keypad designs to examine theoretical performance differences based on the number of keys. The keypad designs tested were the optimal constrained keypad designs from previous research [10].

6.3 Simulation Results

For each run of the ten-fold cross validation simulation, nine-tenths of the BNC Baby corpus was used for building the vocabulary and training various models. The resulting training corpus and testing corpus contain about 3.6 million and 0.4 million words, respectively. As a practical consideration, words with less than 10 occurrences were removed. In the end, vocabularies of about 16,170 word tokens were used in the simulations.

As stated, the semantic relatedness model was built on word stems instead of entire words. The Porter stemming algorithm [23] was applied to the vocabulary to get the word stems. This resulted in ten lists of 9909 word stems on average. The stem lists were then used to build the semantic relatedness models.

Table 1 shows the theoretical maximum predictive disambiguation performances (in DA and KSPC metrics) for our context-aware method and the standard (unmodified) method for select keypad sizes. DA is disambiguation accuracy, defined as the probability that after any keystroke sequence is entered, the desired word will be displayed. KSPC is keystrokes per character, defined as the number of keystrokes required per character of text entered, averaged and normalized over the entire corpus.

Note that since the simulation used a 10-fold cross validation design, each entry is the average of the results from ten separate runs using the same parameters (number of keys and performance metric), but each has different training and testing sets.

After running two-tailed paired *t*-tests, we found that for all keypad sizes, the performances of the context-aware predictive disambiguation methods was consistently and significantly better than the corresponding performances of the standard predictive disambiguation method for equal keypad sizes. In fact, the

Table 1. Maximum predictive disambiguation performances for different predictive disambiguation methods and selected keypad sizes. DA = disambiguation accuracy; KSPC = keystrokes per character (see text for discussion).

Number of Character Keys		3-Key	5-Key	7-Key	8-Key
Keypad Design		ABCDEFGH	ABCD	ABCD EFG	ABCD EFG
		HIJKLMNO	EFGHIJ	HIJKL MN	
		PQRSTUVWXYZ	KLMNO PQRS	MNO PQRS	O PQRS
		-	TUVWXYZ	TUV WXYZ	TUV WXYZ
Frequency Only	DA	67.58%	87.46%	94.54%	95.98%
	KSPC	1.2124	1.0449	1.0163	1.0118
Context-Aware	DA	71.82%	89.80%	95.39%	96.49%
	KSPC	1.1789	1.0358	1.0136	1.0102

performance of the context-aware method always exceeded the performance of the standard method in each individual run of the ten-fold cross validation simulations. It is worth noting that the improvements are most pronounced for small keypad designs, but apply to standard 12-key mobile phones as well (with no changes required to the physical keypad layout or to the way text is entered).

It is also worth noting that the semantic relatedness and POS models are independent of each other, and contribute to different aspects of the improvements to the predictive disambiguation performance. The corresponding performances of the normal predictive disambiguation method, when enhanced with either the semantic relatedness or POS models, are 68.5% and 1.2027, and 71.2% and 1.1856 respectively for DA and KSPC metrics for 3-Key keypad design. In addition, the total improvements are close to the sum of the two corresponding performance improvements when one of the two models is applied.

6.4 Discussion of the Simulation Results

From Table 1, it is clear that the semantic relatedness and POS models consistently helped the dictionary-based predictive disambiguation method to achieve better disambiguation performances in terms of both the DA and KSPC metrics. Performance improves with all the selected keypads of different sizes (from three character keys to eight character keys). The most significant improvement was gained with the three-key keypad using the context-aware predictive disambiguation method. The DA value improved 4.24% from the original 67.58% to 71.82%, and the KSPC value improved 0.0335, dropping from 1.2124 to 1.1789. Note that a 4.24% gain in DA actually accounts for 6.3% of the original disambiguation performance, and the improvement of 0.0335 in KSPC accounts for almost 15.8% of all the extra keystrokes. Again, since the total improvements resulting from the two combined language models is greater than those when either of them is used separately, it should be reasonable to use both models to achieve maximum possible benefits if resource limitations are not primary concerns.

Predictive disambiguation text entry methods on devices with very few keys still remain a significant and important challenge, because of their potential applications for handicapped users with motor or visual impairments. The predictive disambiguation improvements in DA and KSPC metrics become meaningful to these users, as they reduce by about 15% the number of extra keystrokes otherwise required. And about 4% more often, the desired word is found immediately.

With increased keypad sizes, DA and KSPC performance increases rapidly, which leaves less room for improvements from the semantic relatedness and POS models. However, even with eight character keys, the semantic relatedness and POS models still managed to improve DA and KSPC.

7 USABILITY EXPERIMENT

To test the empirical performance of context-aware dictionary-based predictive disambiguation on small mobile devices with very few keys, usability testing was conducted using a PDA implementation of the method and a three-key keypad design. Ideally, testing of the improved predictive disambiguation would be carried out on different mobile devices with different size and

input/output restrictions. However, to more quickly test our concepts, we chose a Pocket PC device with a virtual keypad design as an initial test platform. While we feel this is suitable for initial testing, we will move to other devices as research continues.

7.1 Participants

Thirty-two students (22 male, 10 female; mean age 26.1 yrs) voluntarily participated in the experiment. Twenty-six were graduate students and the rest were undergraduates. As only three used dictionary-based predictive disambiguation text entry methods on a regular basis, participants as a whole were considered novices for the purpose of this study. All participants used cell phones (mean of 4.9 years). Participants were compensated \$10 upon completing the experiment.

7.2 Apparatus

The experiment studied text entry performance using dictionary-based predictive disambiguation with a very small keypad design (three character keys + two function keys). The three-key design was chosen because of the increased interest in small mobile devices with only a few character keys, such as those in Figure 3. In addition, the greater keypad ambiguity in a small complement of keys allows for greater potential improvement in terms of predictive disambiguation performance.

The testing system was implemented on a Dell Axim X30 Pocket PC using Embedded Visual C++ (EVC++).

The participants were instructed to hold the PDA with one hand, and press virtual “keys” using the index finger of their other (dominant) hand. The text entry interface consisted of a text box for displaying input and a virtual 5-key keypad. The three character keys were arranged as per a previous optimized design [10]. The other two keys allowed cycling through candidate words and basic editing. Four functions were implemented:

- **DONE**: When a text phrase is complete, DONE is pressed to clear the text box and begin another phrase.
- **DEL**: Delete the last word entered. Because of the small key set, deletion operated on a word basis and was only allowed in editing mode (when keystroke sequences were committed).
- **NEXT**: In disambiguation mode, NEXT cycles through candidate words if a keystroke sequence is ambiguous.
- **“ ”** (SPACE): Pressing SPACE commits the currently disambiguated word. A space is also inserted after the committed word.

In predictive disambiguation mode (when a sequence of key strokes is entered but not committed), the function keys were NEXT and SPACE. After words were committed, they changed to DEL and DONE, as illustrated in Figures 4 and 5.

7.3 Procedure

The study was performed in a controlled laboratory environment. Before testing, participants completed a questionnaire on background information concerning their use of mobile devices. They were then introduced to the standard dictionary-based

predictive disambiguation text entry method and the operation of our 5-key keypad design, including inputting characters, cycling through candidate lists, committing words and phrases, and correcting errors. Two phrases were entered to practice the interface and method.

During testing, participants entered a set of twenty text phrases using either a standard frequency-based predictive disambiguation method or our context-aware method. Participants were then asked to enter another set of twenty text phrases in a second session using the other method. The order of administering the two methods was counterbalanced to minimize learning effects.

The short text phrases were selected from MacKenzie and Soukoreff's set of 500 phrases [19]. Examples are shown in Table 2. Computer simulation shows that the KSPC values for inputting the entire set of testing phrases using the standard predictive disambiguation and our context-aware predictive disambiguation methods are 1.4467 and 1.3273, respectively. Thus, in theory, participants should experience an 8.3% reduction in total keystrokes with our method.

After testing, participants were asked if they noticed any differences in interacting with each method, and whether they felt that one method was more efficient.

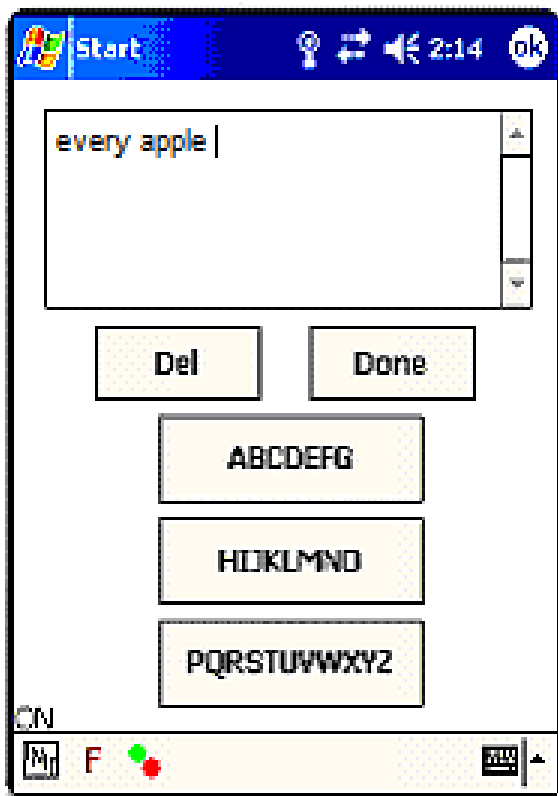


Figure 4. Testing program using a 3-key constrained keypad design in editing mode.

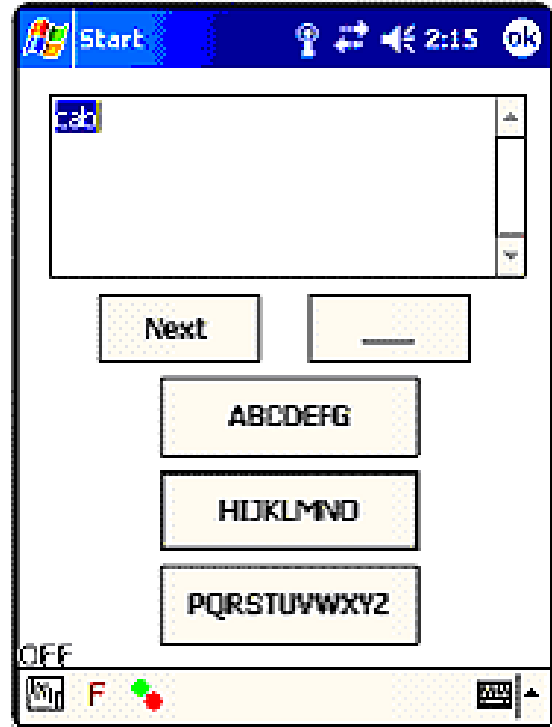


Figure 5. Testing program using a 3-key constrained keypad design in predictive disambiguation mode.

Table 2. Example testing phrases.

you want to eat your cake
every apple from every tree
the plug does not fit the socket
there will be some fog tonight
the price of gas is high

7.4 Results and Discussion

Table 3 shows the average text entry speed (in words per minute, WPM) for the frequency-based and context-based predictive disambiguation methods, along with the average number of errors per word (both assuming an average word length of five characters). Errors are defined as presses of DEL to correct an input word (no other types of error corrections were recorded). Two-tailed paired *t*-tests were used to evaluate whether differences in means were statistically significant. The associated *p* values are given in the right-hand column of Table 3.

On average, the participants achieved text entry speeds of 7.31 wpm using standard disambiguation. The entry speed, 8.01 wpm, was 9.6% faster for our context-aware method. The improvement also approximates the predicted theoretical reduction of 8.3% of the total keystrokes required. The difference was statistically significant ($p = 0.003$).

Table 3. Text entry speeds and error rates for frequency-based and context-aware predictive disambiguation methods. The last column gives the p value associated with the t test for the difference between the two values.

Dependent Variable	Frequency-Only	Context-Aware	Sig. Level
Average Text Entry Speed (WPM)	7.31	8.01	0.003
Average Error Rate (% errors per character)	2.08%	1.64%	0.016

Results also showed a 21.2% reduction in error rates, from 0.104 errors per word with the existing method to 0.082 errors per word with the context-based predictive disambiguation method. The difference was statistically significant ($p = 0.016$).

Responses to the post-study questionnaire suggest there were few perceived differences in interacting with the two interfaces. When asked how similar the two methods were without considering predictive disambiguation accuracy, the average response was 3.06 ($SD = 2.06$) on a scale of 1 to 10, with 1 being very similar and 10 being very different. On a second question concerning which method was more efficient, 15 of 32 participants identified the context-based predictive disambiguation method, stating that the method seemed “smarter” (among the other 17 participants, 10 did not differentiate between either one of the two methods, and only 7 “incorrectly” identified the standard method as “faster”).

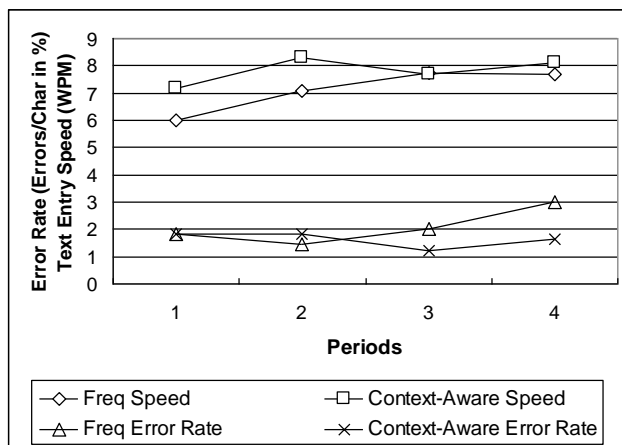


Figure 6. Learning effects on text entry speeds and error rates for frequency-based and context-aware predictive disambiguation methods.

With every ten short phrases grouped into a period, learning effects for text entry speeds and error rates for users of both frequency-based and context-aware predictive disambiguation methods are plotted in Figure 6. Periods are defined so that the first 10 short phrases from the first session are period 1, the last 10 short phrases from the first session are period 2, and so on. Average performances are plotted over each of these four periods.

An important observation is that context-aware predictive disambiguation works in the background without significantly altering the way a user interacts with the text entry method. That

is, the changes are almost transparent as far as the interaction is concerned. The only difference that users may notice is that word lists for an ambiguous keystroke sequence may be in a different order with the context-aware method. It would be interesting to conduct a longitudinal study to see if users can improve their performance by learning the way our system predicts word orderings. More performance gains may be achieved. Overall, while yielding a performance improvement, the context-aware method does not noticeably increase overall attention demands.

8.0 CONCLUSIONS AND FUTURE WORK

In this research, significant improvements were made to existing dictionary-based predictive disambiguation text entry methods. Novel semantic relatedness and part-of-speech models were developed and utilized by improved predictive disambiguation methods to achieve better disambiguation accuracy and fewer required keystrokes. Usability testing confirmed the method’s better text entry performance as was predicted initially during simulation. Lower error rates were also observed for the new context-aware method. Furthermore, there are minimal added attention costs resulting from the context-aware method.

A method for finding the best linear combination of frequency, semantic relatedness, and POS models for the KSPC metric was also introduced and used in our implementation.

Because of the relatively small size of the text corpus for training and testing our semantic relatedness and part-of-speech language models, possible future work includes training the model using larger text corpora to see if predictive disambiguation performance can be further improved.

A potential limitation of the context-aware method is that the same ambiguous keystroke sequence can produce word lists with different orderings due to the current context. This concern is most relevant to expert users, and longitudinal studies could be conducted to determine whether initial benefits in text entry performance continue to outweigh any longer-term limitations.

Finally, possible future work includes the development of predictive disambiguation methods that are more tolerant of user errors. For dictionary-based methods, a word cannot be input unless the correct sequence of keystrokes is made. However, because of the less formal style of mobile text entry, it can be difficult for users to ensure that every keystroke is correct. Therefore, devising dictionary-based predictive disambiguation methods that allow for incorrect keystrokes is a future research topic. The current system does not yet learn text patterns based on specific user history. This could be another interesting avenue for potential improvements. In addition, the method needs to be tested with character sets other than English, as word dictionaries in different languages may provide different results than those presented here.

9.0 REFERENCES

- [1] Baljko, M. and Tam, A. 2006. Indirect text entry using one or two keys. In Proceedings of the ACM Conference on Computers & Accessibility (Portland, OR, October 23 - 25, 2006). ASSETS '06, ACM Press, New York, NY, 18-25.
- [2] Boggess, L. 1988. Two simple prediction algorithms to facilitate text production. In Proceedings of the Second

- Conference on Applied Natural Language Processing (Austin, TX, February 9 - 12, 1998). Association for Computational Linguistics, Morristown, NJ, 33-40.
- [3] British National Corpus. <http://www.natcorp.ox.ac.uk>.
- [4] Budanitsky, A. and Hirst, G. 2006. Evaluating WordNet-based measures of lexical semantic relatedness. *Computational Linguistics* 32 (Mar. 2006), 13-47.
- [5] Connolly, D. and Lundy, H. D. 1999. Method and System for Intelligent Text Entry on a Numeric Keypad. US Patent Number 6005495.
- [6] Dominowska, E., Roy, D., and Patel, R. 2002. An adaptive context-sensitive communication aid. In Proceedings of the 17th Annual Technology and Persons with Disabilities Conference (Northridge, CA, 2002). Available at: www.csun.edu/cod/conf/2002/proceedings/csun02.htm#d
- [7] Dunlop, M. 2004. Watch-top text-entry: Can phone-style predictive text-entry work with only 5 buttons? In Proceedings of the 6th International Conference on Human Computer Interaction with Mobile Devices and Services (Glasgow, Scotland, September 13 - 16, 2004). *MobileHCI '04*. Springer-Verlag, Berlin, 342-346.
- [8] Evreinova, T., Evreinov, G., and Raisamo, R. 2004. Four-key text entry for physically challenged people. In Adjunct Proceedings of the 8th ERCIM Workshop "User Interfaces For All" (Vienna, Austria, June 28 - 29, 2004). Available at: www.ui4all.gr/workshop2004/publications/adjunct-proceedings.html.
- [9] Flinchem, P. E., Grover, D., Grunbock, C., King, T. M., and Kushler, A. C. 2001. Reduced Keyboard Disambiguating System. US Patent Number 6307548.
- [10] Gong, J. and Tarasewich, P. 2005. Alphabetically constrained keypad designs for text entry on mobile devices. In Proceedings of the ACM Conference on Human Factors in Computing Systems (Portland, OR, April 2 - 7, 2005). *CHI '05*. ACM Press, New York, NY, 211-220.
- [11] Gong, J., Tarasewich, P., Hafner, C., and MacKenzie, I. S. 2007. Improving dictionary-based disambiguation text entry method accuracy. In Extended Abstracts of the ACM Conference on Human Factors in Computing Systems (San Jose, CA, April 28 - May 3, 2007). *CHI '07*. ACM Press, New York, NY, 2387-2392.
- [12] Goodman, J., Venolia, G., Steury, K., and Parker, C. 2002. Language modeling for soft keyboards. In Proceedings of the Eighteenth National Conference on Artificial Intelligence (Edmonton, Canada, July 28 - August 1, 2002). *AAAI-02*. AAAI Press, Menlo Park, CA, 419-424.
- [13] Kukich, K. (1992). Technique for automatically correcting words in text. *ACM Computing Surveys* 24 (Dec. 1992), 377-439.
- [14] Levine, S. H. and Goodenough-Trepagnier, C. 1990. Customised text entry devices for motor-impaired users. *Applied Ergonomics* 21 (Mar. 1990), 55-62.
- [15] Levine, S. H., Goodenough-Trepagnier, C., Getschow, O.C., and Minneman, L.S. 1987. Multi-character key text entry using computer disambiguation. In Proceedings of the 10th Annual Conference on Rehabilitation Technology (San Jose, CA, June 19 - 23, 1987). Trace Center, University of Wisconsin-Madison, 177-178.
- [16] Li, J. and Hirst, G. 2005. Semantic knowledge in word completion. In Proceedings of the ACM SIGACCESS Conference on Computers and accessibility (Baltimore, MD, October 9 - 12, 2005). *ASSETS '05*. ACM Press, New York, NY, 121-128.
- [17] MacKenzie, I. S., Kober, H., Smith, D., Jones, T., and Skepner, E. 2001. LetterWise: Prefix-based disambiguation for mobile text input. In Proceedings of the ACM Symposium on User Interface Software and Technology (Orlando, FL, November 11 - 14, 2001). *UIST '01*. ACM Press, New York, NY, 111-120.
- [18] MacKenzie, I. S., and Soukoreff, R. W. 2002. Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction* 17. Taylor & Francis Group, London, UK, 147-198.
- [19] MacKenzie, I. S. and Soukoreff, R. W. 2003. Phrase Sets for Evaluating Text Entry Techniques. In Extended Abstracts of the ACM Conference on Human Factors in Computing Systems (Fort Lauderdale, FL, April 5 - 10, 2003). *CHI '03*. ACM Press, New York, NY, 754-755.
- [20] Manning, C. D. and Schütze, H. 2003. *Foundations of Statistical Natural Language Processing*. MIT Press.
- [21] Masui, T. 1999. POBox: An efficient text input method for handheld and ubiquitous computers. In Proceedings of the First International Symposium on Handheld and Ubiquitous Computing (Karlsruhe, Germany, September 27 - 29, 1999). *HUC '99*. Springer-Verlag, Berlin, 289-300.
- [22] Phone Key Pads. <http://dialabc.com/motion/keypads.html>.
- [23] Porter, F. M. 1980. An algorithm for suffix stripping. <http://www.tartarus.org/~martin/PorterStemmer/def.txt>.
- [24] Rabiner, L. R. and Schafer, R. W. 1976. Digital techniques for computer voice response: Implementations and applications. *Proceedings of the IEEE* 64, 4 (April 1976). *IEEE*, 416-433.
- [25] Rau, H. and Skiena, S. S. 2004. Dialing for documents: An experiment in information theory. In Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (Santa Fe, NM, October 24-27, 2004). *UIST '04*. ACM Press, New York, NY, 147-155.
- [26] Sandnes, F. E., Thorkildssen, H. W., Arvei, A. and Boverud, J. O. 2003. Techniques for fast and easy text-entry with three-keys. In Proceedings of the 2003 Norwegian Informatics Conference (Oslo, Norway, November 24-26, 2003). *NIK '03*. Tapir Academic Publishers, 205-216.
- [27] Shannon, C. E. 1951 Prediction and entropy of printed English. *Bell System Technical Journal* 30. John Wiley & Sons, Hoboken, NJ, 50-64.
- [28] Seymore, K. and Rosenfeld, R. 1996. Scalable backoff language models. In Proceedings of the Fourth International Conference on Spoken Language Processing (Philadelphia, PA, October 3-6, 1996). *ICSLP '96*. *IEEE*, 232-235.
- [29] Stocky, T., Faaborg, A., and Lieberman, H. 2004. A commonsense approach to predictive text entry. In Extended Abstracts of the ACM Conference on Human factors in

Computing Systems (Vienna, Austria, April 24-29, 2004).
CHI '04. ACM Press, New York, NY, 1163-1166.

- [30] Viterbi, A. J. 1967. Error bounds for convolutional codes and asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory* 13 (Apr. 1967). IEEE, 260-269.

- [31] Wobbrock, J. O., Myers, B. A., and Kembel, J. A. 2003. EdgeWrite: A stylus-based text entry method designed for high accuracy and stability of motion. In *Proceedings of the Annual ACM Symposium on User Interface Software and Technology (Vancouver, Canada, November 2-5, 2003)*. UIST '03. ACM Press, New York, NY, 61-70.