Power Email: Efficient Email Entry on Pen-Based Handheld Devices

Orkut Buyukkokten, Hector Garcia-Molina, Andreas Paepcke, Terry Winograd
Digital Libraries Lab (InfoLab), Stanford University, Stanford, CA, 94305
{orkut, hector, paepcke, winograd}@cs.stanford.edu

ABSTRACT
Email access from handheld devices is cumbersome, partly because of the considerable effort in entering text using pen-based devices. Even though much research has been done on improving text entry speed on handheld devices, little research has been done on text entry for e-mail messages. In this paper, we utilize the user knowledge and application semantics of email to enable easy and fast email entry on pen-based devices. We introduce several text entry techniques that combine Graffiti with a word selection interface where the user can complete words by choosing from a menu. We report several experiments that evaluate these techniques. These experiments revealed that the two most effective methods were personalized text entry and message in context. We measured a significant improvement in text entry speed, reduction in input effort and reduction in input error rate.

Keywords
Handheld, Email, Graffiti, Palm Pilot, Pen-Based, PDA, Word Completion

INTRODUCTION
Palm Pilot and Windows CE handheld digital information appliances are now widely used to fill daily information needs, such as calendar and address management. As these devices are beginning to be connected to the Internet through wireless networks, communication applications are being added to the typical personal information management facilities. Unfortunately, email, the traditional 'killer app' for networked environments, has been slow to spread onto handheld devices. One major obstacle to blame for this delay is the inconvenience of text input on handhelds.

Creating an email message on a handheld device includes entry of the recipient, a subject line, the message body, and a signature. For most handheld devices, users enter these text portions with a pen on a pressure-sensitive surface. In some cases, specialized character sets, such as Palm Inc.'s Graffiti, are used to make character recognition easier, and to speed text entry. Even with such optimizations, text entry remains slow and error-prone. Our measurements on test subjects show a Graffiti input speed of about 0.7 characters per second. An informal survey of wireless email users confirmed the "slow and cumbersome" construction of messages as a major source of user dissatisfaction.

LEAVE BLANK THE LAST 2.5 cm (1") OF THE LEFT COLUMN ON THE FIRST PAGE FOR THE COPYRIGHT NOTICE.

Our goal is to make the pen-based entry of email messages on handheld devices as easy as possible. We have explored several word-completion techniques that allow the user to select among likely words from a list, based on what has been entered so far and on the message context. We exploit application semantics and knowledge about individual users in deciding what words to display. A simple tap by the user then completes the word. We have implemented and tested several alternative designs.

One question that arises immediately with a text completion approach is how many candidate words the device should suggest on the screen at a time. Handheld screens are very small, scanning a long list of words can take longer than entering additional characters. We conducted user experiments to find the optimal number of candidate words, which we report in this paper.

The success of our word completion technique further hinges on our ability to predict words the user wishes to write. The system's prediction method therefore must be very accurate. We conducted user experiments to learn, for example, whether it is better to base predictions on a user's own past email messages, or on a larger pool of messages that were created by multiple users, and whether in generating a reply we should favor words that appear in the incoming message. We were able to achieve a 133% increase in text input speed by picking the best prediction methods.

While our implementation includes support for entering salutations and subject lines, our focus is on speeding up the entry of email message bodies. In the remainder of the paper we first introduce our implementation with a walkthrough. We then present design alternatives and user studies for the word table presentations. This is followed by an exploration of the word prediction alternatives, related work and summarizing conclusions.

COMPOSING AN EMAIL MESSAGE
We initially implemented a system we call *Power Email* on the Palm operating system. Figure 1 shows a sample screenshot. The uppermost area of the screen consists of a

toolbar. The toolbar buttons from left-to-right are used for looking at the message list, composing a message, reading a message, editing the address book, sending and receiving messages, setting up the preferences and the account, deleting a message, and quitting the application.

Before we go into more detail about the technical challenges and our solutions, we will start with a sample walkthrough that will demonstrate how the user interacts with the system. Let us assume the user wants to write a message to 'Tyler' and invite him out for dinner. We will separate this task into five phases and examine the user interaction in each of the phases.

a. *Entering the recipient:* The user taps on the *To* button directly under the toolbar (Figure 1) to go to the *Contact Selection Form* (Figure 2). This form displays the names in the contact list. The user taps on the entry 'Tyler Ziemann' from the list instead of writing out the email address. (The button with the arrows is used to navigate through the contact list.)

b. *Entering the salutation, closing and signature:* The application automatically returns to the *Compose Form* and fills in the salutation and the closing of the message (See Figure 3). The salutation and closing is contact-specific and is specified by the user during the entry of recipients into the contact list. Power Email also appends a personal signature to the message. The signature can be changed and customized for each contact as well. The cursor is placed between the salutation and closing, at the beginning of the first sentence. At his point, the user can start entering the message or entering the subject.

c. *Entering the subject*: The user taps on the *Subject* button. This takes the user to the *Subject Selection Form* (Figure 4) where the user can enter the subject or choose from a list of common subjects that we pre-defined. The user taps on 'Dinner.' The system sets the subject and returns to the *Compose Form*.

d. *Entering the text*: The user starts writing the message, entering letters with the pen using Graffiti. As the user enters letters, s/he is presented with a menu of words to choose from. We call this menu, the *Prediction Table*. For instance, after entering the first letter 'L', the menu

Figure 1: The Compose Form

Figure 2: Contact Selection Form

Figure 3: Salutation and Closing

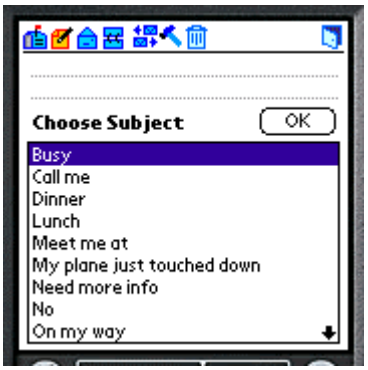

Figure 4: Subject Selection Form

Figure 5: Word Selection Menu

in Figure 5 is presented. Instead of writing the rest of the word (i.e., "let's"), the user simply taps on the corresponding entry in the menu. If the desired word is not listed in the menu, s/he enters a second character and a new prediction table is presented. The process continues, and for some 'rare' words the user has to complete the entry. The user finishes the message by entering text and selecting the words on the menus presented. This text-entering phase, the most time-consuming one, will be the main focus of this paper.

e. *Sending the message*: Finally, the user taps on the *Send* button to send out the message.

To illustrate the saving offered by Power Email, let us consider one simple example. (Detailed experiments are discussed later.) Suppose that the user entered: "Let's get together for dinner sometime. How about tonight? Let me know." as the message in our running example. For this case, Table 1 compares the input effort involved in composing the same message with Power Email and other popular email applications on the same platform. The input effort is measured in terms of the number of Graffiti characters entered (gf) and the number of taps (tp) on the screen. As can be seen from the table, even with the best competitor (OmniSky), one has to perform almost three times more actions (110) than with Power Email (39 actions).

Step:	a	b	c	d	e	Total
Power Email	2 tp	0	2 tp	16 tp 18 gf	1 tp	21 tp 18 gf
Palm Mail [12]	5 tp	1 tp 25 gf	1 tp 6 gf	71 gf	1 tp	8 tp 104 gf
OmniSky Mail [11]	2 tp	1 tp 25 gf	1 tp 6 gf	71 gf	1 tp	6 tp 104 gf
Multi Mail Pro [10]	5 tp	1 tp 25 gf	1 tp 6 gf	71 gf	1 tp	8 tp 104 gf

Table 1: Input effort for a sample email message

There are many challenges involved in coming up with the right menus and entries. Email messages are generally very short and include many abbreviations. If we look at the words used in writing emails, we see a considerable difference in the selection of words as compared to other kinds of documents. The vocabulary is much smaller and some of the words tend to occur more frequently. Power Email exploits this knowledge by selecting common email words for prediction tables. Furthermore, the selected words can also be tailored to a specific user's vocabulary or to the context (e.g., replying to a message).

In the rest of the paper, we will discuss several approaches for building good prediction tables. We also provide a detailed user study that shows performance comparisons.

CONSTRUCTING THE PREDICTION TABLES

One decision we need to make is how to choose the words that may be included in the prediction tables. In order to choose our corpus, we analyzed 3105 messages sent by 409 different people. The messages were all composed on a desktop because we wanted to study messages that users would really like to send if they had a good email application. (Initially we did not analyze emails specific to a single user. We present those results in a later section.) Since messages written on small devices are generally short, we eliminated messages that were longer than 20 lines when rendered on a regular Palm screen. After eliminating the long email messages, we were down to 1356 messages. Then we ran the words through a spell checker and eliminated the words that were misspelled or words that were special names, such as names of people and places. As we will see, some of the people and place names are re-introduced by other portions of the prediction process. We computed the word frequencies of all the remaining words. The frequencies were used to create the prediction tables. All of these computation-intensive operations were done on a desktop computer, and they only need to be performed once.

A prediction table contains the high frequency words for a given prefix. In particular, $PT_{n,i}(p)$ is a prediction table with *n* entries, for the first *i* characters of prefix *p*. We refer to *i* as the level and *n* as the size of $PT_{n,i}(p)$. For example, Figure 6 shows the tables $PT_{12,1}(s)$, $PT_{12,2}(sh)$ and $PT_{12,3}(sho)$. (Even though the size of $PT_{12,3}(sho)$ is 12, only 2 rows are displayed since there are only 7 words in the table.) Table $PT_{12,1}(s)$ will be displayed after the user enters 's', table $PT_{12,2}(sh)$ will be displayed after the user enters 'sh', and so on.

saturday	should	somethin	stanford
see	so	soon	still
sensitivit	some	sorry	sure

shall	she	shopping	shouldn't
share	she's	short	show
sharp	shoes	should	showed

shoes	short	shouldn't	showed
shopping	should	show	

Figure 6: Prediction Tables

The processor on a small device, such as the Palm Pilot, has the power of a desktop machine in the mid 80's. Creating the prediction tables as the user types the characters takes too much time if we compute the tables on the fly from the corpus. Therefore we pre-computed the prediction tables, and for each prefix we created a separate table of words. A *Lookup Table* was generated to get the appropriate prediction table for a given prefix. Finally the lookup table and the prediction tables were downloaded on to the handheld device. For the rest of the paper, we will refer to the set of all the prediction tables used as the *Dictionary*.

There are three important decisions to make regarding prediction tables: the number of levels to use, the size and the actual entries. We discuss size and entry selection in the upcoming sections. In the rest of this section we discuss the number of levels.

We do not wish to provide prediction tables that are too sparse, since computing and storing them is too expensive given the low benefits. The number of possible words that start with two letters can be as many as 80, based on our corpus. Thus, unless we use huge prediction tables at level 2, a third level is needed. On the other hand, if we look at three letters, there are only three prefixes (i.e., 'sta', 'com' and 'pro') with more than 20 words. Thus, level 4 tables would have very few new words to display. Therefore we decided to use three levels of prediction tables. If the user does not see the desired word after entering three characters, s/he will have to enter the full word.

OPTIMIZING THE TABLE SIZE

Since a table $PT_{n,i}(p)$ will be displayed as *x* rows by *y* columns, we have to select the appropriate number of rows and columns. The number of columns is determined by our display. In our case, the width of the Palm Pilot screen is *160* pixels, so we will have *160/y* pixels per word. If a word does not fit in the table cell, only the beginning of the word is displayed.

We analyzed the corpus and computed the number of words that fit in a cell for a given number of columns. Table 2 shows the results. Even though 3 columns seems to be a better option, by inspection, we determined that with 4 columns most of the truncated words were still identifiable. On the other hand, with five columns, many more words are not identifiable. Therefore we decided to use four columns in the tables (*y=4*), which gives 40 pixels per cell. This means that the choices for the number of entries *n = 4*x* will be 4, 8, 12...

Columns	Words that Fit
6	52.46%
5	65.77%
4	82.11%
3	96.6%

Table 2: Words that fit for a given number of columns

The other factor to consider is the number of rows, *x*. A table with too many rows becomes too cumbersome to scan. On the other hand, a table with too few rows does not help much with the selection either since the user has to write additional letters to see the word on the table. In order to figure out the optimum number of rows for the prediction tables, we performed a user study.

User Study 1 – Table Size

We chose 10 subjects with Graffiti experience and asked them to enter email messages from our corpus using Power Email. In the study the users copied all the messages word by word from a paper handout to the handheld devices. We experimented with 5 different methods (i.e., table sizes). These methods sized the tables respectively at 0 rows, 1 row, 2 rows, 4 rows and 6 rows. The table size with 0 rows translates to plain Graffiti entry with no prediction tables. The messages were chosen randomly from the 1356 messages mentioned earlier. We randomly chose 5 short (66-134 words), 5 medium (156-180 words) and 5 long (221-300 words) messages among the 1356 messages. Each subject was asked to write 3 messages (one of each size) with each method for a total of 15 messages. The order of the email messages was the same for each user but we changed the order in which the users were exposed to the methods. The user input consisted of two repeated actions: i) entering a character using Graffiti and ii) tapping on the prediction table to choose a word. Our Power Email is instrumented to record all the actions and the time elapsed between these actions. We measured the text entry speed for each message as the number of characters in the message divided by the time in seconds to enter the message.

The text entry speed is a function of the method, the subject and the order of the method. The mathematical model we used for the design of the experiment can be written as:

$$speed_{ijk} = \mu + \alpha_i + \beta_j + \phi_k + error_{ijk}$$

where μ devotes the overall mean, α_i denotes the effect of method *i* (*i*=1, 2, 3, 4, 5 for rows 0, 1, 2, 4 and 6 respectively) on the speed, β_j denotes the effect of the subject (*j*=1, 2, …10), ϕ_k denotes the effect of the order in which the method was presented to the subject (*k*=1,2, …5) and $error_{ijk}$ denotes the random effect on the i^{th} method on the j^{th} subject in the k^{th} order. The order of a method is the position of the method in which the method was presented to the subject. (For instance, if the subject completed Method 3 first, the order of Method 3 would be 1.) We ran a univariate analysis of variance (Univariate ANOVA) over the resulting data. Figure 7 gives the mean speed for methods, statistically adjusted for the effects of order and subject. We see in Figure 7 that entering messages with Graffiti only yielded an average entry speed of 0.69 characters per second, while, for example, the help of prediction tables with 2 rows improved the speed to 0.89 characters per second.

We also performed additional analysis on our data to assure that the difference between methods is significant. In order to verify whether a method *X* is significantly different than method *Y*, we need to test the hypothesis:

H_0 = the effect of method *X* and method *Y* are the same

Our regression analysis (Univariate ANOVA) adjusts the effect of the subjects and orders and computes the *p* values. Our analysis showed that the differences between the first three methods were significant (*p* at most 0.0382). Similarly the difference between 4 and 6 rows was also significant (*p*= 0.0441). Thus changing the number of rows has a direct effect on the text entry speed.

As can be seen in Figure 7, the maximum speed is measured when there are 2 rows. However, if we look at the graph, we can see that the speed improves as the rows increase from 0 to 2 and the speed decreases for more than 4 rows. This confirms our earlier intuition that, as the number of rows increases, the text entry speed increases since desired words are more likely to be in the table.

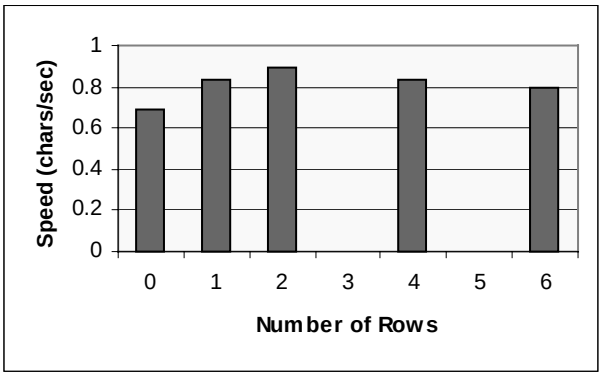

Figure 7: Text entry speed for different table sizes

However, as the number of rows increases beyond some point, the table becomes too large and scanning the table requires too much effort. By extrapolation from Figure 7, 3 seems to be the optimum number of rows for maximizing entry speed. Unfortunately, our initial experiment did not include a 3-row method.

User Study 2 – 2 Rows vs. 3 Rows

In order to make sure that 3 was indeed the optimum, we ran an additional study with 4 subjects who were asked to write messages using 2 rows and 3 rows. The text entry speed with 3 rows was 7.2% faster compared to 2 rows. This result showed that 3 rows was indeed the optimum number of rows for the table. In the rest of the paper, we will refer to the method of using three rows as the *Twelve Method* since the prediction tables display up to *n=12* words.

EVALUATING THE WORD SELECTION OPTIONS

Based on user studies 1 and 2, we understand that tables with 3 rows work best. But is the choice of selecting these words by usage frequency ranking optimal as well? In order to find out, we experimented with several different design alternatives for the prediction tables.

User Study 3 – Word Prediction

We performed another user study to explore the effect of these different word choice methods on the selection speed. We had 12 subjects. Each of them completed 3 messages (one short, one medium, and one long) for each method. We changed the order in which the users were exposed to the methods.

a. *Distinct Method:* The tables we used in studies 1 and 2 had overlapping entries across levels. For instance when the user types the letter 'c', 'could' is one of the words given in $PT_{12,1}(c)$. After the user types 'o' as the second letter, the word 'could' was still visible in $PT_{12,2}(co)$. For the 'Distinct' method, we instead eliminated all words the user saw but did not pick at a previous level. The eliminated words are replaced by new words further down the rank list. This way we can present more options to the user.

b. *Short Method:* Word completion by selection is not very useful for short words. For instance, assume that we wish to write 'car', and the word does not appear in the table until we write both 'c' and 'a'. In that case, it might be faster just to write the remaining 'r' than to switch attention to the table. In order to make it easier to enter short words, this method favors the words that are shorter in size. In particular, $PT_{12,1}(\alpha)$ shows the most common words, starting with α, that are 2 or 3 characters in length. If there are empty cells left in the prediction table, these cells are filled with the remaining popular words.

c. *Launch Method:* In all the methods so far, the prediction table is introduced after the user enters at least one character. In this method, the user is presented with a prediction table, $PT_{12,0}$ at the beginning of the message and after any punctuation. We call $PT_{12,0}$ the *Launch Table*. We analyzed the 1356 messages mentioned earlier and found the most frequent 12 words that show up at the beginning of a sentence or after a punctuation character. These words are shown in Figure 8, and constitute the Launch Table. We used only one launch table, that is, the table that the user sees after entering any punctuation character or starting a new sentence is always the same. This makes it faster for the user to get accustomed to the words and their locations and makes the selection easier.

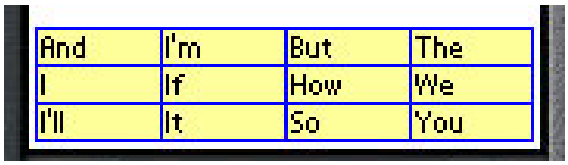

Figure 8: The Launch Table

We ran a univariate analysis of variance (Univariate ANOVA) to compute the adjusted mean speed for each method and compared all the methods against Graffiti and the Twelve methods since all of them were variations of that method. The result is summarized in Figure 9.

a. *Distinct Method:* The Distinct Method is 1.17% worse than the Twelve Method. The p-value is 0.0691, which tells us that the effect of these two methods are significantly different. Some of the subjects had a hard time with the Distinct Method for two reasons: i) The subject failed to notice the word after s/he entered the first letter. The subject entered another letter, causing the word to disappear. Consequently s/he had to complete the whole word without the benefit of selection. ii) The

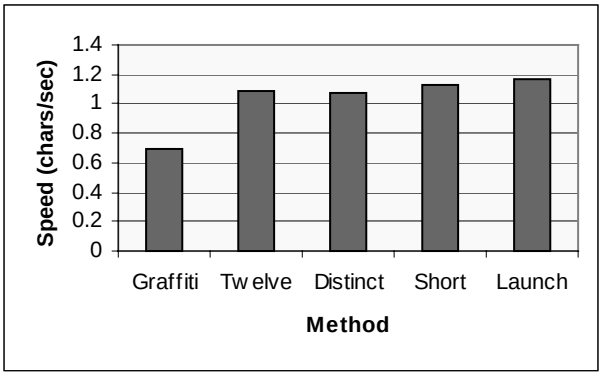

Figure 9: Text entry speed for word prediction methods

subject noticed the word just as s/he finished the first letter, but his/her hand 'kept writing' before s/he could stop him/herself to make the selection. Depending on the length of the word, sometimes the subject preferred to erase the last letter so that the word would show up again in the table for selection. Obviously, in both cases, showing distinct words wasted a significant amount of time.

b. *Short Method:* If we look at the adjusted mean of the speed, we see a 3.33% improvement. The effect of this method is statistically insignificant in the analysis.

c. *Launch Method:* We observed a 7.15% improvement in text entry speed using the Launch Method over the Twelve Method. Here the difference of these methods is statistically insignificant. However, if we look at the data set, we notice that only 5 out of 12 people used the Launch Table at all. This is probably because users were not expecting a prediction table before they entered a character. In all other cases, the table appears after a character is entered. Note that the Launch Table may or may not improve the text entry speed depending on the user. However, launch tables will not make performance worse.

To summarize, the above experiments suggest that prediction tables should not erase any matching words and that launch tables work well only for some users, but at least do no harm for others.

MESSAGE IN CONTEXT AND PERSONALIZED DICTIONARY

In this section we study how to get further improvements by exploiting the context in which the email is created and by personalizing the Prediction Tables.

a. *Message in Context:* We used an algorithm that dynamically changes the tables if the user is responding to a message. In this case all the words with the appropriate prefix that occur in the message that the user is responding to automatically occur in the prediction tables. To illustrate, consider the messages in Table 3. Under normal circumstances, the words 'proceeding' and 'citation' would not occur in the prediction tables since they are not among the most frequent 12 words. However, since the message we are responding to includes these words, in this method these words would come up if the user enters 'p' or 'c' respectively, speeding up the response.

Message:
Could I borrow your copy of the '90 CHI *Proceedings* for a day or two? I need them for a couple of *citations*.
Response:
Sure, I'll bring the *proceedings* tomorrow. What *citation* do you need?

Table 3: Message in context.

b. *Personal Dictionary:* Another possible way of improving the text entry speed is to create a dictionary personalized for the user. To accomplish this, the messages from the sent-items folder of the users' mailboxes are used to pre-compute the prediction table and the lookup table.

User Study 4 – Message in Context and Personalized Dictionaries

We conducted another user study in order to test the effectiveness of these two additional methods. For message in context we randomly chose three email messages (Msg_A, Msg_B and Msg_C) and one response for each message ($Reply_A$, $Reply_B$, and $Reply_C$ respectively) from the 1365 messages. When a subject entered $Reply_A$, we added the words in Msg_A to the appropriate prediction tables, and so on. For the personal dictionaries, we obtained the sent-items folders from three users ($User_X$, $User_Y$, and $User_Z$). (These users were not used as subjects.) We used 1001 short messages from each user. We randomly chose one message from each user and then created the dictionary using the remaining 1000 messages. We will refer to the single messages we chose as Msg_X, Msg_Y and Msg_Z respectively. When the subject writes Msg_X, the dictionary that was created from the messages of $User_X$ was the basis for predictions, and so on. Note that in this experiment the dictionary was not personalized to the test subject, but rather to the user who originally wrote the test message. The experiment is realistic because the subject simply copies the messages, and the predictions are personalized to the original creator of the test message.

For our experiment we used 12 subjects. Every subject wrote all six messages ($Reply_A$, $Reply_B$, $Reply_C$, Msg_X, Msg_Y and Msg_Z). Each subject wrote two of $Reply_A$, $Reply_B$ and $Reply_C$ using message in context, and two of Msg_X, Msg_Y and Msg_Z using the personalized dictionary. The subject entered the remaining two messages using the Twelve Method without the message in context or the personalized dictionary support. We changed the order in which the subjects were exposed to the messages and the methods. The mathematical model we used for the design considered in the experiment is the same as in user studies 1 and 3. We used a Univariate ANOVA procedure to compute the adjusted mean values of the speed for the methods. The results are summarized in Figures 10 and 11.

We ran a regression analysis to test whether the effect of any of these methods was significant. In this case we had to consider two groups, the methods that have the messages $Reply_A$, $Reply_B$, and $Reply_C$, and the methods that have the messages Msg_X, Msg_Y , and Msg_Z.

a. *Message in Context:* The improvement in text entry we measured was 18.31% beyond the improvement that the Twelve Method gave us over Graffiti. In other words, if the user types 100 character using Graffiti, s/he can type in the same amount of time 157 characters using the Twelve Method, and 176 characters using Message in Context. The p-value is 0.0691, which tells us that the

effect of using a dictionary that is context specific is significant.

b. *Personalized Dictionary:* The improvement in text entry we measured was 12.27% beyond the improvement the Twelve Method gave us over Graffiti. In other words, if the user types 100 character using Graffiti, s/he can type 173 characters using the Twelve Method and 186 characters using a Personalized Dictionary in the same amount of time. The p-value is 0.1231, which tells us that the effect of using a context sensitive dictionary is notable but not quite statistically significant. The effect of using a personalized dictionary is not as strong as using a message in context.

LEARNING

In our experiments we observed that user performance improved as they gained experience with Power Email. (This improvement is 'masked out,' in our previous results due to the way we compute average speed-ups.) In our final experiment we focused on how users improve over time.

User Study 5 – Learning Effect

In this study we had 12 subjects. They were asked to write 6 messages using the Twelve Method described earlier. We changed the order in which the subjects were exposed to the tasks, but the methods remained the same.

Figure 12 shows the adjusted mean speed for the i^{th} task (using the same type of statistical analysis as before). The improvement of the speed between the tasks is summarized in Figure 13. For example the speed on Task 2 was 0.067 characters per second faster than for Task 1. As can be seen, as the subject writes more messages, the change in the text entry speed becomes less and less significant.

One would expect an analogous improvement when users learn Graffiti. However, our users were all experienced Graffiti users (on the average they had used Graffiti for 22 months), so we do not expect their performance to improve over the length of our experiments. Comparing the text entry speed we gain at the final task over the Graffiti speed, we measure a 115.16% improvement in text entry speed. That is, if the user enters 100 characters using only Graffiti, a trained Power Email user can enter 215 characters in the same amount of time.

RELATED WORK

There have been many techniques proposed to improve text entry speed on pen-based devices. One common technique uses a virtual keyboard displayed on a tablet (soft keyboards) such as the Fitaly [14] and OPTI [7] keyboards. The Reactive Keyboard [5] also uses prediction for selection, but at the lower level of key presses. POBox [9] introduces an input method based on dynamic query of the dictionary and word prediction from context. POBox uses a soft keyboard instead of Graffiti and is not email specific.

Another set of related work does address the context of email. Most of the work done has been on classifying e-mail messages[4], filtering email messages and junk email [8, 13], email organization [1] and email agents [2,6].

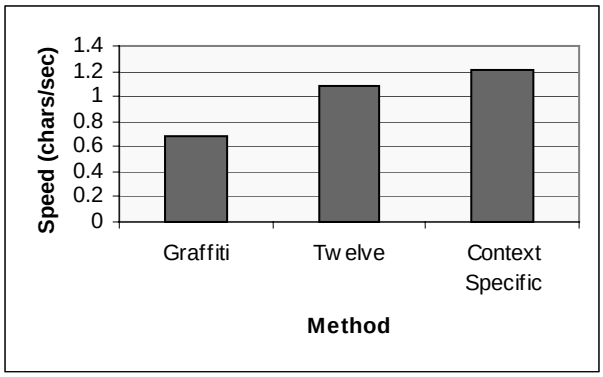

Figure 10: Text entry speed for context specific dictionary

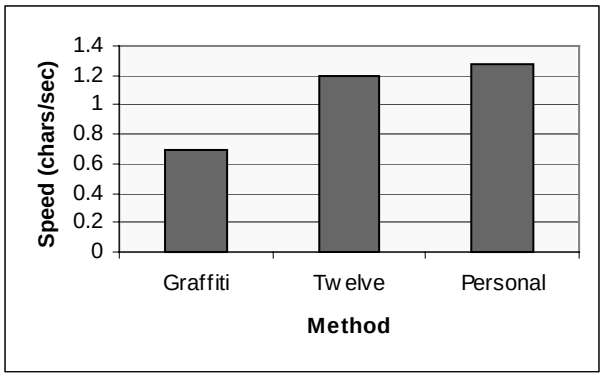

Figure 11: Text entry speed for personal dictionary

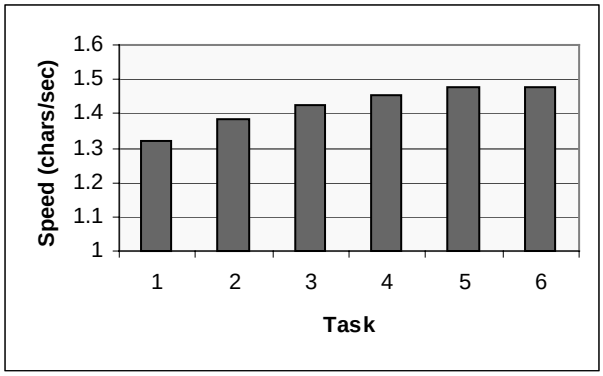

Figure 12: Text entry speed over time

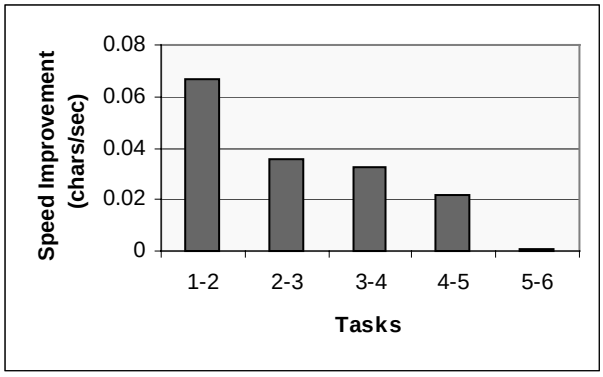

Figure 13: Improvement in text entry speed over time

There has also been research for replying messages using structured response objects instead of text [3].

CONCLUSION

This paper has introduced new techniques for efficiently composing email messages on pen-based handheld devices. Our hypothesis was that the application semantics of email, a user profile and the context of the email message could be used to improve performance. We combined Graffiti with a menu selection for entering words where we introduced prediction tables that predict the word the user is writing. We presented experimental results comparing different design alternatives.

Notice that the speed improvement with prediction tables varies among our user studies. The subjects were exposed to different table sizes in user studies 1 and 2 (i.e., the number of table rows changed). This made it harder for the subjects to get accustomed to the tables. When the user is presented with the same size tables consistently, the 'memory affect' makes him/her recognize the tables, locate words within the table more easily and make the selection more quickly. In user studies 3 and 4, the table size was fixed at 12, so the speed increases (from about 0.69 characters/sec to between 1.2 and 1.3 characters/sec.). Finally, in user study 5, we study how speed improves over time, and we see an even higher speed (about 1.48 characters/sec.) once the user is familiar with the Twelve Method. At this point, we measured an improvement of 115% compared to the Graffiti Method.

When we experimented with the Message in Context and the Personalized Dictionary Methods, we did not evaluate them after the user had been 'trained.' However, if we hypothesize that the gains we observed in studies 3 and 4 will carry over to the case where the user is trained, then we would expect even higher speeds. That is, if user types 100 characters with Graffiti, in the same amount of time a trained user can type 215 characters with the Twelve Method, and we would expect a trained user to enter about 233 characters with Message in Context (and Twelve). Thus, a message that takes the user 10 minutes to write in a traditional handheld email application would take about 4 minutes using Power Email.

Although we did not discuss it in the paper, we also compared the input effort and error ratio with and without Power Email. For example, if the number of actions (characters + taps) with the Graffiti method was 100, then the number of actions with the Twelve Method would be on the average 46.54, i.e., a reduction of 53.46%.

The error ratio tells us how many mistakes a user made. In particular, if a user types 100 characters, deletes *x* of them, and retypes *x* characters, then we say the error ratio is *x*%. Without prediction tables, the observed error ratio was 17.96%, while with the Twelve method, this ratio was down to 5.31% which gives a 12.65% improvement.

After the user studies, the subjects responded to a survey on their satisfaction. We asked them if they found our text entry approach useful. All of the subjects answered *YES*. We also asked them how much they liked our approach on a scale between 0 and 10 (10 very much, 5 indifferent, 0 not at all). The average was 8.92. The survey also included a free text field where subjects were encouraged to enter any comments they might have. One interesting comment was about the length of the words in the tables. Some subjects preferred longer words whereas some subjects preferred shorter words to be given in the prediction tables. One subject suggested sorting the words in the table by length as supposed to alphabetically. We plan to experiment with this variation in the future.

REFERENCES

1. Bälter, O. Keystroke Level Analysis of Email Message Organization, in *Proceedings of CHI '00*, 2000, ACM Press, 105-112.
2. Boone, G. Concept Features in Re:Agent, an Intelligent Email Agent, in *Proceedings of the Second International Conference on Autonomous Agents*, 1998, ACM Press, 141-148.
3. Camino B.M., Milewski, A.E., Millen, D.R., & Smith, T.M. Replying to Email with Structured Responses. In *International Journal of Human-Computer Studies*, 48, 763-776.
4. Cohen, W.W. Learning rules that classify e-mail, in *Papers from the AAAI Spring Symposium on Machine Learning in Information Access*, 1996, 18–25.
5. Darragh, J.J., Witten, I.H., & James, M.L. The Reactive Keyboard: A predictive typing aid. *IEEE Computer* 23, 11 (November 1990), 41-49.
6. Gruen, D, Sidner, C., Boettner, C., & Rich, C. A Collaborative Assistant for Email, in *Proceedings of CHI '99: Extended Abstracts*, 1999, ACM Press, 196-197.
7. MacKenzie, I.S. & Zhang, S.X. The Design and Evaluation of a High-Performance Soft Keyboard, in *Proceedings of CHI '99*, 1999, ACM Press, 25-31.
8. Marx, M. & Schmandt, C., CLUES: Dynamic Personalized Message Filtering, in *Proceedings of ACM CSCW '96* , 1996, ACM Press, 113-121.
9. Masui, T. An Efficient Text Input Method for Pen-Based Computers, in *Proceedings of CHI '98*, 1998, ACM Press, 328-335.
10. MultiMail Pro, ActualSoft: http://www.actualsoft.com/
11. Omnisky, OmniskyMail, http://www.omnisky.com/
12. Palm, Inc., PalmMail, http://www.palm.com/
13. Sahami, M., Dumais, S., Heckerman, D. & Horvitz, E. A Bayesian Approach to Filtering Junk E-Mail, in *Proceedings of AAAI '98 Workshop on Learning for Text Categorization*, Madison, WI, 1998.
14. Textware Solutions, The Fitaly Keyboard, http://www.twsolutions.com/fitaly/fitaly.htm