An ACT-R model of SMS behavior on the optimal mobile telephone keyboard

T. W. Schaap (s1015583)

May 21, 2005

RuG

Supervised by:

Dr. H. van Rijn
Dr. N. Taatgen

Kunstmatige Intelligentie
Rijksuniversiteit Groningen

Abstract

The current alphabetical distribution of letters on a mobile telephone keyboard is suboptimal because no thought has been given to the structure of the input language. By separating the most frequent letter combinations within the Dutch language and by assigning the most frequent letters a low string number on a key, the number of keystrokes per letter is reduced and text entry on the new layout is made theoretically faster. However, because of the new letter distribution, which seems random at first sight, longer search times can be expected, which could outweigh the speed-up resulting from the optimal letter distribution. The factors that influence the moment at which the location of a letter is known are examined using a cognitive model in ACT-R/PM. The model gives us insight in the theoretical text entry speed that can be reached on the new keyboard layout and the search strategies that are developed to locate letters on an unfamiliar layout. The model initially searches systematically, but later develops strategies based on its newly gained knowledge. The main factor that influences the moment at which a letter's location on an unfamiliar layout is known is the frequency of the letter in a text. The string number of a letter or the number of other letters on its key does not have a significant influence on the difference in search times between the beginning and the end of the task. The model's results are tested in an experiment. The model results show that the new letter distribution leads to a speed-up in typing on the new keyboard layout compared to the original layout of 30 %.

Contents

List of Figures

Voorwoord

De afgelopen maanden hebben in het teken gestaan van het schrijven van mijn afstudeerscriptie. Nu het proces afgerond is en ik terug kijk op de afgelopen periode, voelt het als een mooie bekroning op mijn studietijd. Tijdens de verschillende stadia van het onderzoek heeft het regelmatig tegen gezeten, heb ik moeten ploeteren en vroeg ik me soms af waar ik het allemaal voor deed. Maar de goede momenten, de dagen dat de woorden gemakkelijk uit mijn handen vloeiden en ik wist welke kant ik op ging, die waren zeer de moeite waard. Ik heb veel geleerd: over wetenschappelijk onderzoek, over prioriteiten stellen en doorzetten, en bovenal over mijzelf. Het was een mooie tijd die ik kan afsluiten met een positief gevoel.

Een aantal mensen hebben hun licht laten schijnen over mijn scriptie en tegen hen wil ik dan ook van harte mijn dank uitspreken. In het bijzonder wil ik Hedderik van Rijn bedanken voor zijn intenstieve begeleiding bij het leerproces van de afgelopen maanden. Hij heeft me op weg geholpen als ik vast zat en op een zeer positief-kritische manier geholpen deze scriptie vorm te geven. Daarnaast ben ik hem erg dankbaar voor zijn wijze en motiverende uitspraak: "Als de uitkomst van een onderzoek niet overeenkomt met je hypothese, dan is dat geen teleurstelling: het is een ontdekking!" In de komende jaren zullen deze woorden waarschijnlijk nog vaak door mijn hoofd spelen. Ook Niels Taatgen heeft me erg geholpen met zijn begeleiding en evaluatie. Zijn positieve aanbevelingen hebben ertoe bijgedragen dat er een scriptie voor u ligt waar ik trots op ben. Ten slotte wil ik Roel en Anne heel hartelijk bedanken voor hun hulp bij de taal- en tekstproblemen die ik onvermijdelijk ook tegen kwam.

Met deze scriptie sluit ik mijn studie Technische Cognitiewetenschap/ Kunstmatige Intelligentie met een heel goed gevoel af. Stafleden, medestudenten: bedankt voor een leuke en leerzame tijd!

Chapter 1

Introduction

1.1 The mobile telephone keyboard

Mobile telephones have become an important medium for storing and sharing information. A growing number of people use mobile phones for sending textual information to others using SMS (Short Message Service). Obviously, entering and sending textual information requires text entry on the keyboards of these telephones. The keyboards used for this purpose are in many ways different from standard keyboards. This influences the way people use these keyboards. Three obvious ways in which a mobile telephone keyboard is different from the standard "QWERTY" keyboard (Sholes' layout), are:

- it is much smaller,
- it has more letters on one key,
- the letters are alphabetically ordered.

We will first discuss these three differences in turn with respect to their influence on text entry.

Prior research has shown that the size of a keyboard has little to no effect on typing speed. Sears and Zha (2003) found that size of a keyboard has little impact on the speed of text input on this keyboard. Also, Fitts' Law (Fitts, 1954) predicts that changing the size of a keyboard does not have an effect on the speed of data entry, as long as ratio between the distance between the keys and the size of the keys stays the same. For that reason, the issue of size of the keyboard will not be dealt with in this paper. The other two factors, the familiarity of the letter distribution and the number of letters on one key, do have an important effect on typing speed (Sears, Jacko, Chu & Moro, 2001).

The fact that more letters are placed on one key has an important consequence: Keys have to be pressed multiple times to enter most of the letters. A letter's position within a key is called its string number. When for instance the letter "s", a letter with string number 4 on key "pqrs" in the alphabetical layout, has to be entered, this key has to be pressed four times. Also, one has to wait a certain amount of time before a next letter from the same key can be entered, in most mobile phones around 1000 milliseconds (Silfverberg, MacKenzie & Korhonen, 2000). This means that entering two letters from the same key consecutively

adds a waiting time of around one second every time.

With respect to the alphabetical ordering of the letters, an advantage is that it is relatively easy for users to familiarize themselves with the layout. This means that a user has to search less for the locations of letters than on an unfamiliar layout, at least in the beginning. The search costs of a familiar layout are therefore lower. However, if no thought is given to the structure of the language used, the case where two letters from the same key have to be entered successively will be relatively frequent. This is the case with the alphabetical ordering of the letters. Obviously, the more frequent a user has to press a button multiple times, the slower text entry becomes. Text entry costs will therefore be higher when no thought has been given to the structure of the language used for typing.

This paper focuses on the possible contradiction between the effects of layout on search costs and on text entry costs. While a familiar layout may look easier to use at first sight, it might be ineffective in the long term because the text entry costs are too high. This has been a topic of research with regard to normal sized keyboards. The alphabetical layout has long been written off for these keyboards and the use of Sholes' layout is widely spread. Sholes' layout is more effective, because the most frequent letter couples were placed on opposite sides of the keyboard to avoid jamming (Yamada, 1980 in Zhai, Kristensson & Smith, 2004). The Dvorak layout has been proven to be even faster, with a speed increase of about 4% compared to Sholes' layout (West, 1998). Its design "fits hand-stroking skills to the sequence patterns of English words" (Dvorak, Merrick, Dealy & Ford, 1936, p. 218, in West, 1998). Because thought has been given to the language used, the letters are placed in such a way that hands alternate as much as possible and the home (middle) row is the row with the most common letters. Although the typing speed improvement of the Dvorak keyboard compared to Sholes' layout is modest, we believe that using the same method to design a mobile telephone keyboard layout will have a big effect on text entry speed on these keyboards. Small changes will have a relatively big impact, because more letters are placed on one key and therefore typing delays or speed improvements can occur easily. The language that we used for our model is Dutch. We will therefore base the layout for the mobile telephone keyboard on the structure of the Dutch language.

In Chapter 2 we will introduce a new layout for the mobile telephone keyboard that has a reduced text entry time. This layout that is based on the structure of the Dutch language will initially have as a disadvantage that letters have to be searched for more often than on an alphabetical layout. An advantage will be that the penalty resulting from multiple presses or waiting to enter the next letter will be smaller. This advantage may even get bigger with practice, because search times will be reduced once the user becomes familiar with the new layout. This paper will not only focus on the reduced entry times, but will also describe the search strategies that users develop when faced with an unfamiliar layout. We will show that the reduced entry costs will eventually outweigh the high search times, favoring the optimized keyboard layout over the original layout.

1.2 Research questions and outline of the paper

The research questions that will be addressed in this paper are the following:

(1) How long does it take before users know where a certain letter is located on an unfamiliar keyboard layout? (2) Which mechanisms can explain this learning process? (3) Which are the search strategies that people use to find letters on an unfamiliar layout? And (4) to what extend is a new layout which is faster in theory, also faster in practice, when compared to the original alphabetical layout?

The next chapter will explain how a new keyboard layout was created based on the structure of the Dutch language. Chapter 3 will then describe a cognitive model which learns to search for letters on the new layout. We have collected data in an experiment that is described in Chapter 4. A description of these results can be found in Chapter 5. Next, we will test the model's validity in Chapter 6 by comparing its results to human behavior. Finally, we will answer our research questions, based on the predictions from the cognitive model.

Chapter 2

Keyboard Design

2.1 Introduction

The process of designing an optimal telephone keyboard layout based on the structure of language can be divided into three stages. First, a corpus that is appropriate to assess the language structure must be selected. Details about language structure will be explained in Section 2.2 and we will elaborate on the process of selecting an appropriate corpus in Section 2.3. Then information about the relative frequencies of characters and bigrams pairs of characters has to be extracted from this corpus, as will be described in Section 2.4. Finally the optimal layout has to be determined using the information acquired from the corpus, which will be described in Section 2.5.

2.2 A new layout based on bigram frequencies

The new layout will be based on the relative frequencies of bigrams. A bigram is any combination of two letters that occur together in a word, such as "th" and "he" in "the". The new layout will have lower text entry costs, because letters from highly frequent bigrams will be placed on different keys. This will bring down the entry costs a great deal: a user will not need to wait as often to press the same key, which has a penalty of about 1000 milliseconds in most mobile telephones (Silfverberg et al., 2000). Furthermore, placing the letters from these frequent bigrams on buttons that are located closer together helps reduce the entry costs even more because of the smaller moving time between two letters. To accomplish this, the relative frequencies of all the bigrams in a representative corpus have to be calculated.

2.3 Selecting a representative corpus

As MacKenzie and Soukoreff (2002) discussed in the context of predictive text input (also known as T9 for SMS messages), they define a corpus to be representative if it (1) is appropriate for the type of messages (e.g., personal e-mail messages are different from formal letters) and (2) includes all the characters used in typing (including simple punctuation and spaces). We define a third constraint: a corpus is representative if it is in the language that is used for text entry. Using a different language than the one used for typing would give a suboptimal layout, because bigrams that are very frequent in one language (for instance,

"th" in English) can be rare in other languages. A keyboard layout designed to be optimal for one language should obviously not be based on the structure of another.

We therefore had to select a corpus in Dutch that is is representative of SMS messages and that includes all characters. Unfortunately, a Dutch corpus for SMS messages does not exist. However, if we can prove for the Dutch CELEX Lexicon (Baayen, Piepenbrock & Van Rijn, 1993) that it is representative of SMS messages, this corpus can be used for extracting information about the language structure for Dutch SMS messages. We determined its representativity by comparing the Short Messages Service Corpus (National University of Singapore, 2004), which is an SMS corpus in English, and the English CELEX Lexicon on the level of characters and bigrams. The NUS SMS Corpus is a set of 10.000 SMS messages sent on mobile phones in Singapore. The origins of the Dutch CELEX Lexicon and English CELEX Lexicon are similar. Therefore, if the English lexicon is representative of the English SMS language, we can assume that the Dutch lexicon is representative of Dutch SMS language.

To establish the degree of resemblance between the NUS SMS Corpus and the English lexicon, we determined the frequencies of all the characters and bigrams in both corpora. This was done using "Awk", a tool for manipulating and analyzing text. R (R Development Core Team, 2004), an environment for statistical computing, was used to compare the frequencies. Each word in the CELEX corpus was preceded and followed by a space to ensure that the starting and ending letters of a word were counted correctly. In the comparison between the two corpora, punctuation was not taken into account, because the English lexicon does not include punctuation symbols. In the final design, the frequencies for the characters in punctuation from the NUS SMS Corpus will be used. We also did not make a distinction between capitals and lowercase letters, because the use of capitals in the CELEX Lexicon is limited to names. This does not resemble the grammatical use of capital letters. The order of the bigrams in the NUS SMS COrpus and the CELEX Lexicon are shown in Figure 2.1. Spaces are denoted by a single dot.

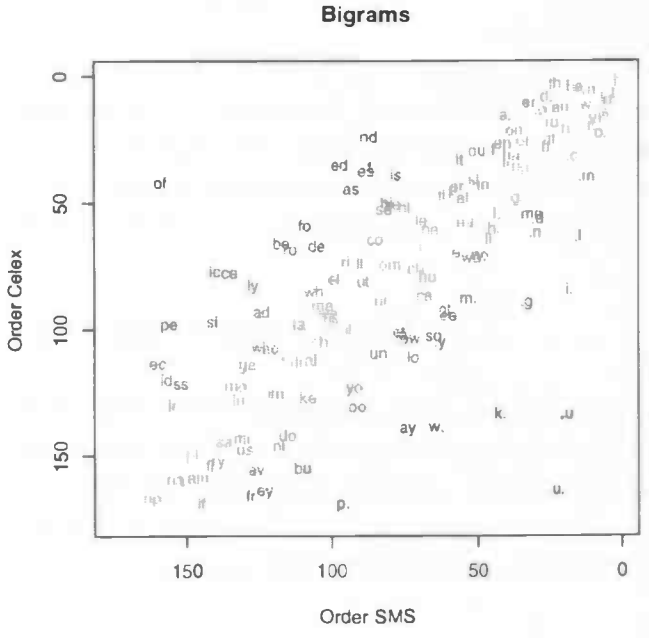

Figure 2.1: Distribution of bigrams in the NUS SMS Corpus (National University of Singapore, 2004) and the English CELEX Lexicon (Baayen et al., 1993)

From Figure 2.1 it can be seen that there are not many large deviations in the distribution of the bigrams. For some of the bigrams that do deviate, an explanation is easily found. The word "of" is usually used in a more formal context, leading to a higher frequency of this bigram in the CELEX Lexicon than in the SMS Corpus. On the other hand, since SMS messages are mostly personal messages about the sender and the receiver, the words "I" and "you", or "u" in short, will be used more often in SMS texts than in the texts the CELEX Lexicon was based on. In Dutch however, these deviations may be less apparent. The words for "I" and "u" in Dutch consist of multiple letters ("ik", "jij") instead of one. The bigrams in these words are frequent in other words as well. The bigrams in Dutch that are equivalent to the bigrams "i.", ".u" and "u." in English will therefore not deviate as much in the Dutch language. It should also be noted that the NUS SMS Corpus was in English, but with many so called "Singlish" words in it. This might have affected the comparison in a negative way, because syllables like "leh" and "yia" are very common in Singlish, but don't occur in regular English. The correlation between the two corpora is .87. This is not a perfect fit, but the correlation is high enough to consider the English CELEX Lexicon representative of SMS language.

Prior research supports this idea. Doering (2002) found that language used in SMS is not as different from normal language as is often thought. Messages do on average contain more abbreviations than normal language, and the messages have a more interpersonal content, but she found no evidence for a distinct SMS language.

We therefore draw the conclusion that the Dutch CELEX Lexicon is representative of Dutch SMS language, because the English CELEX Lexicon is representative of English SMS language and the origins of the Dutch and English corpora are similar.

2.4 Analysis of the corpus

After we had determined that the Dutch CELEX Lexicon (Baayen et al., 1993) is representative for Dutch SMS messages, we extracted from it the relative frequencies of characters and bigrams in the Dutch language. We used the same methods as we had done with the English CELEX Lexicon. The list of bigrams and their frequencies can be found in Appendix A. With these data we designed the new keyboard.

2.5 Designing the new keyboard

There are two factors that determine the speed with which text can be entered on a keyboard: the number of times one has to wait to press two letters from the same key consecutively, and the movement time between two buttons. Recall that the waiting time between two letters from the same key is 1000 milliseconds in most mobile telephones (Silfverberg et al., 2000). The movement time from one button to another is given by Fitts' Law (Fitts, 1954):

$$MT_{ij} = a + b \log_2 \left(\frac{A_{ij}}{W_j} + c \right) \tag{2.1}$$

where MT_{ij} is the movement time from starting key i to target key j, a and b are empirically determined, device dependent constants, A_{ij} is the amplitude of movement from i to j, W_j is the width of the target and c is a constant of 0, 0.5 or 1 (see MacKenzie & Buxton, 1992, for details). The equation was adapted by MacKenzie and Soukoreff (2002) for the application of single-finger or stylus entry on a small physical keyboard with a reduced key set, leading to the following equation:

$$MT_{ij} = 0.204 \log_2 \left(\frac{A_{ij}}{W_j} + 1 \right) \tag{2.2}$$

The amplitude of movement A_{ij} is measured by assigning each key an x-y coordinate and calculating their distances using the Pythagorean identity.

The time to move the finger from key 1 to key 9 is 271 milliseconds (we used a mobile telephone with a key width of 6 millimeter and a diagonal distance of 20.5 millimeter). Note that the penalty for waiting to press the same key twice exceeds this movement time by over 700 milliseconds. This gives further support for the idea that letters from frequent bigrams should be placed on different keys as much as possible.

A mobile telephone keyboard has eight or nine keys with letters and three with special characters, such as simple punctuation and the space. When a keyboard has eight keys with letters instead of nine, pressing the ninth key changes the font to capital letters. We decided

to distribute the twenty-six letters of the alphabet evenly over the nine keys to reduce the number of times that letters from the same key are typed consecutively. This means that we do not have a separate key for capital letters, but this function can be put on another key as well. Because twenty-six letters are not easily divisible for nine keys, we added an empty symbol to the letters (in this case, a capital "X"). This resulted in nine keys with each three characters. The total number of unique three letter sets, or possible keys, was 2925. As the optimal order of letters within a key is based solely on frequency, it was not necessary to generate all permutations.

The frequencies of all the bigrams on the keys were added to calculate the respective key penalties. The frequencies of double letters, such as "aa", were not included in this calculation. Their frequencies do not contribute to a better distribution of the letters on the keyboard, because placing these letters on a different key does not change the movement time or the number of times one has to wait to press a key on the same key. For example, to calculate the penalty of the key "abc", the frequencies of the six bigrams "ab", "ba", "ac", "ca", "bc" and "cb" were added (yielding a penalty of 109331). This way, the penalties for keys containing highly frequent bigrams were high and the penalties for keys with less frequent bigrams were lower.

We implemented an algorithm in R (R Development Core Team, 2004) which selected the optimal key combination by a depth-first search through all possible layouts. Its ultimate goal was to find the layout with the lowest possible combination penalty. Some heuristics were implemented to speed up the searching process and to cut off some of the branches of the search tree. The algorithm started each cycle of the search process by determining the first letter in the alphabet that was not yet included in the combination. This way, every letter would be used exctly once in the key combination. It then determined which key containing the first unused letter in the alphabet had the smallest key penalty. All keys with letters that had already been used were excluded from the list of possible keys, rendering the search tree a lot smaller. The penalties of the keys that were used in the key combination were added to calculate the total keyboard penalty. Only if a total combination penalty was less than any combination penalty calculated before, the layout with this total penalty was registered as an improvement. The smallest combination penalty could obviously not be improved and the respective key combination was therefore returned as the final combination. To speed up the search process, we rearranged the keys in increasing order of penalty. Keys with the lowest penalties were considered first for the selection, and keys with too high key penalties were automatically cut off from consideration for the new layout. This way, the first total combination penalty was already relatively low and many branches could be cut off from the search tree. The annotated source code of the algorithm can be found in Appendix B.

This algorithm returned a combination of nine keys: auy, btz, cjr, dkp, eq(X), fmn, gsv, hlw, iox. The total penalty of this combination is 56837. This is the sum of all the bigram frequencies in the three letter sets on the nine keys. For comparison: the penalty of the key "abc" alone is 109331, almost twice as high! This means that we have greatly reduced the number of times that two letters from the same key have to be typed consecutively. Note however that this does not mean that the new layout is 55 times faster than the original layout, which has a total penalty of 3,154,915 (this is the penalty for the alphabetical key-

board with eight letter keys and a separate key to change the font to capital letters). We have minimized the number of times that two letters from the same key have to be typed consecutively, but the time it takes to move to another button or to press a button multiple times is not part of this calculation. We can safely assume that the new layout will be faster, but the exact speed increase for the whole keyboard has to be calculated using Fitts' Law. This can not be done for a set of letters that are not yet finally distributed over the keyboard, because movement time is an important part of text entry time.

So at this point the three letters on each key had to be arranged in a certain order, based on their relative frequencies. Obviously, we assigned lower string numbers to the letters with higher frequencies: the lower its string number, the faster a letter can be entered!

After rearranging the letters on each key, we calculated the best distribution of the keys over the keyboard using Fitts' Law as adapted by MacKenzie and Soukoreff (2002) for the application of single-finger or stylus entry on a small physical keyboard with a reduced key set. This was done in a way that the letters from the most frequent bigrams were placed close together and the amount of movement required to type on the new keyboard layout was minimal.

The Dutch CELEX Lexicon, like the English CELEX Lexicon, does not include punctuation. As the NUS SMS Corpus does, and we have no reason to assume that the use of simple punctuation is different for English versus Dutch, we used the punctuation from the NUS SMS Corpus to place these special characters on their keys. Again we used the respective frequencies, assigning the most frequent characters the lowest string numbers.

The keyboard that we designed has the following layout:

GVS	TBZ	AUY
DKP	EQ	NMF
OIX	RJC	LWH
.) (;	␣ -	? , ! :

Entering all the bigrams from the entire Dutch CELEX Lexicon is 32.0% faster on this new layout than on the original keyboard with the alphabetical layout. The text entry time for each bigram was calculated by multiplying the time to enter that bigram, given by Fitts' Law (as adapted by MacKenzie & Soukoreff, 2002), by the total frequency of the bigram in the lexicon. If a bigram consisted of two letters from the same key, a penalty of 1000 milliseconds was added to the entry time. This way, typing all the bigrams from the total Dutch Lexicon takes 4.67×10^{10} milliseconds on the original keyboard and 3.17×10^{10} milliseconds on the new keyboard. Note that this might be different than entering the actual CELEX Lexicon because we used a fixed starting point for the finger (the middle button on the middle row) for each bigram.

In the remainder of this paper we will compare the performances of people and a cognitive model on the optimal layout and the alphabetical layout. As described before, mobile telephone keyboards often have eight letter keys and one separate key for changing the font to capital letters. Because our new layout has nine keys with letters, we will also use nine of

the keys for letters with the alphabetical letter distribution. A function for changing the font to capital letters can be implemented under either of the three keys with special characters. This will ensure that the differences between the two layouts do not result from the different number of keys that are used for typing. The alphabetical layout that we will use from hereon will therefore have the following letter distribution:

ABC	DEF	GHI
JKL	MNO	PQR
STU	VW	XYZ
:;()	␣-	.,?!

The theoretical speed improvement of the new layout compared to this alphabetical layout is 31.6 %.

Chapter 3

Modeling the search process

3.1 Introduction

In Chapter 2 we have described how a new optimal keyboard layout was designed based on the frequencies of letters and bigrams in the Dutch CELEX Corpus. Typing on this keyboard layout is in theory much faster (about 32% increase in typing speed). However, a negative consequence of the new layout might be that it is less familiar to users than the original, alphabetical layout and thus might cause users to search for letters more and longer, especially in the beginning. These two consequences of changing the keyboard layout can be contradictory: improving the layout of a keyboard to increase the speed of typing has little use if the user needs to search longer than the time he can gain with the entry speed improvement. After some time however, the user will learn where specific letters can be found and might change to other search strategies in accordance with the new layout. This contrast between typing speed improvement and search delay leads to the following research questions (copied from Section 1.2):

(1) How long does it take before users know where a certain letter is located on an unfamiliar keyboard layout? (2) Which mechanisms can explain this learning process? (3) Which are the search strategies that people use to find letters on an unfamiliar layout? And (4) to what extend is a new layout which is faster in theory, also faster in practice, when compared to the original alphabetical layout?

The original alphabetical layout may have led to the development of certain search strategies in the task of finding a letter on a keyboard, which is a big part of typing text messages. Lovelace and Snodgrass (1971) found that people have a good idea of how a certain letter is positioned in relation to others in the alphabet. When given two letters from the alphabet, people know quickly and accurately which letter comes before the other. Furthermore, Klahr, Chase and Lovelace (1983) found that people access letters in the alphabet in their memory through chunks of two to seven letters. These chunks are ordered and when people are asked where a letter can be found in the alphabet, the position of the chunk that contains the letter gives information about the position of the letter in the alphabet. This obviously leads to an idea of where to search on an alphabetically ordered keyboard: for instance, the "a", from the first chunk, can be found on the left upper corner, the "x", a letter from the last chunk, on the bottom row of the keyboard and the "k", a letter from one of the middle chunks, on a button in the middle row. This way, knowledge of the alphabet directly

gives information about the distribution of the letters on the keyboard and search strategies are based on knowledge retrieval. On the other hand, for a totally unfamiliar layout, this knowledge will not be of any help and other search strategies will be more efficient. Random search or starting from the top and working towards the bottom will therefore become better ways of searching though the letters. Once the user gets some experience in using the new layout, he might change his search strategies to searching in a specific region on the keyboard or trying to recall the exact location of the letter.

This chapter will describe an ACT-R model for the task of sending text messages on both the original and the new keyboard. The model starts out as a novice, without any knowledge of the two layouts. It does however have basic knowledge about the way text messages have to be typed on a keyboard. After several runs, it will learn where it can find the letters and how it can improve its speed and it will develop one or more search strategies to find these letters faster. This will ultimately tell us whether typing on the new layout, which is theoretically faster, is faster in practice as well and which factors influence the speed of learning the locations of letters on an unfamiliar keyboard. In the following section we will first give a short description of ACT-R. Also the task of sending a text message will be examined in more detail. The actual model and its results will be described later in this chapter.

3.2 ACT-R

ACT-R is a production system for modeling human cognition (Anderson & Lebiere, 1998). In ACT-R, two types of knowledge are represented. The first type is procedural knowledge, knowledge in the form of rules we follow and actions we take in trying to achieve a certain goal. These rules are called production rules and they are implicit, meaning that one cannot verbalize the knowledge represented in them. The second type of knowledge is declarative knowledge, knowledge about facts. ACT-R represents this declarative, explicit knowledge in chunks. Where ACT-R focuses on central cognition, ACT-R/PM contains additional modules, representing visual and motor attention and action. This gives ACT-R the possibility to not only reason about the environment, but to also interact with it. Through perceptual requests a visual object can be looked at; motor actions, like movement of a mouse or hand, are also possible.

Procedural learning is made possible in ACT-R by the process of production compilation (Taatgen & Lee, 2003). This is the process of building new rules from the combination of already existing rules and chunks that are fired and retrieved in sequence. Through this process ACT-R learns new rules and can thus access its declarative memory more effectively, or handle a task faster or more efficiently.

ACT-R/PM combines declarative and procedural learning with visual and motor capacities. Therefore ACT-R/PM can facilitate models that learn about complex tasks, like text messaging. This is why we chose this environment to model our task.

3.3 The task

Sending a text message on a mobile telephone consists of several parts. At first, the message has to be defined. Assume the sender wants to send the text "hi". Then the letter to be entered has to be selected. In this example, this is the letter "h". This letter then has to be located on a button on the keyboard. A letter's string number can be derived from its position on the button. In the case of the alphabetical layout, the letter "h" has the second position (string number 2) on key "ghi". When the correct button is located, this button has to be pressed the appropriate number of times. A letter's string number denotes the number of times the button has to be pressed to enter the letter on the keyboard, so in this case button 3 has to be pressed twice. Finally, the letter shown on the display has to be checked, after which the cycle starts over again. This means that the sender has to search for the letter "i" on the keyboard, press the appropriate button the correct number of times, and check the display to see if the correct letter was entered. This is a complex task that involves cognitive skills, visual attention and manual movement. It also requires memory, setting priorities and keeping track of goals.

3.4 The ACT-R Model

The model starts with the goal of selecting the first letter of a given message. It then starts looking at the window shown in the simulation environment. This environment shows a mobile telephone keyboard as depicted below in Figure 3.1.

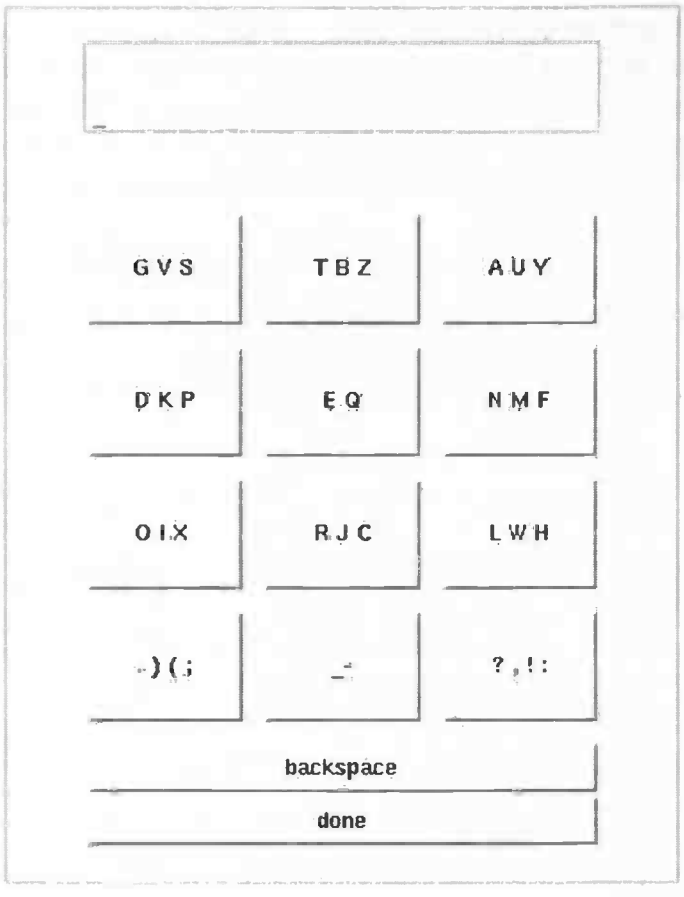

Figure 3.1: The keyboard in the simulation environment

The model initiates a visual action, trying to look at a certain item on the keyboard. Once found, the item is attended and the value of the text becomes available to the model. The model examines the button text to check whether this button contains the letter. If the letter is not found in the button text, memory is searched to determine whether the letter can be found in declarative memory. Simultaneously the search process starts over again, starting with putting another visual-location in the visual buffer and ending with the check if the desired letter is on the button looked at. This cycle continues until the model either retrieves a chunk from declarative memory with the location of the letter, or finds a button that contains the desired letter. Once the correct button is found, a new goal is triggered to move the hand to the location of that button. It then presses the button a certain number of times, depending on the string number of the letter. Once a button is pressed this many times, a new goal is initiated to check whether the letter was entered correctly. This is done by searching for the new letter in the display and comparing this newly appearing letter to the goal letter. User errors, such as pushing an incorrect button or pressing the button a wrong number of times, are possible, because of a motor-related noise parameter. If the display shows the wrong letter, the model searches for the Backspace and presses it, after which it tries to type the original letter again. Finally, when the correct letter is shown in the display, the process repeats from the beginning with the remainder of the message.

As has been explained above, the task of sending a text message consists of different

parts, called subtasks. The model handles these subtasks in a mostly serial way. It initiates new (sub)goals for every (sub)task of the process. As can be seen from Figure 3.2, these goals return to their parent to return the information gained once the goal has been reached. They also initiate new (sub)goals if it turns out they can not reach the ultimate goal themselves. Retrieval goals can exist at the same time as these other goals, leading to situations where to model can try to retrieve chunks from declarative memory while the rest of the process continues.

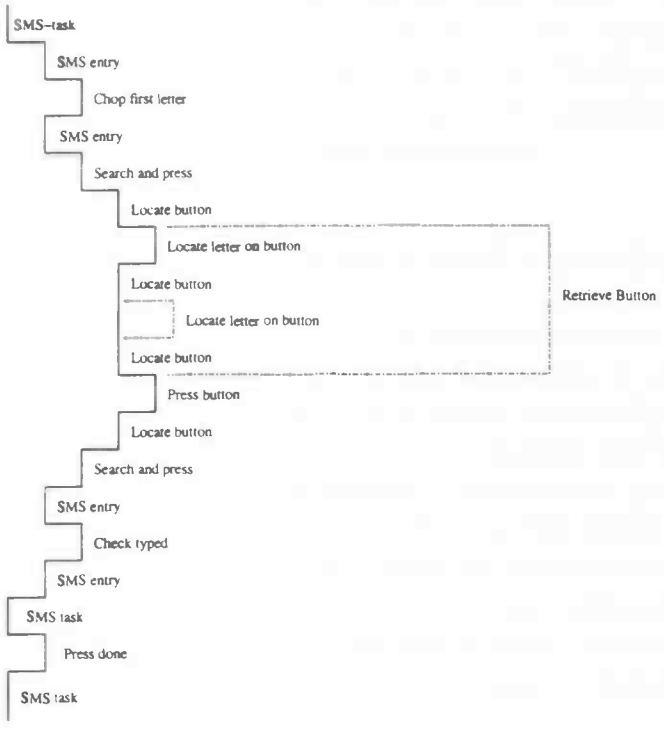

Figure 3.2: Flowchart of goals of one cycle in the process of typing a text message (see text for extensive description)

The goals have the following content:"SMS-task" is the ultimate goal, that is, the goal to type and send a certain message. This means entering the message and pressing the button "Done" at the end. "SMS entry" is the first subtask: entering the message. Within this task, three subtasks can be defined, starting with chopping the first letter off the message to be typed. This goal is called "chop first letter". After this task has been fulfilled, it returns its value to its parent. This parent in return initiates its second subtask. This goal is called "Search and press" and its task is to locate and press the button that contains the desired letter. This initiates a subtask to find the letter ("Locate button"). It finds a visual-location and focuses on it. It then looks for the desired letter in the string found on the button. This goal is depicted as "Locate letter on button" in Figure 3.2. The outcome of this task determines what the next move is. There are two possibilities: the outcome is either an integer (the string number of the letter) and the desired letter is found, or the outcome is "FALSE" and the desired letter is not yet found. In the first case, a new task is initiated to press the button as many times as the value of the integer. In the second case, the cycle starts over again from"Locate Button" and the next button is looked at, until the correct button is found. This is reflected by the dotted line with the text "Locate letter on button".

Another possibility to find the position letter is to retrieve the location of the button from declarative memory. This retrieval is initiated the moment the subtask "Locate button" is started. It can return its information at any time during the cycle of searching for the appropriate button.

Once the appropriate button has been located, the button is pressed the appropriate number of times ("Press button") and the information is returned to a higher task, "SMS entry". This initiates the third subtask of "SMS entry", checking the results of "Search and press". This is the task called "Check typed". If the correct character is shown on the display, all information is returned to the parent task, "SMS-task". Once all the letters have been entered, the main task initiates one last goal, to press the button "Done". After this, the program returns to the main goal and terminates.

3.5 Model behavior and learning

The model starts as a novice with knowledge of only the most basic aspects of the task: it can look at different items on the keyboard, it can move its finger from one location to another and it can determine how many times it has to press a button when the desired letter is found on it. It also knows how to identify the first character to be typed and how to check whether the letter in the display is the desired letter. It does not have any information about the ordering of the letters on either mobile telephone keyboard. An unfamiliar distribution of the letters on a keyboard forces the user to search in a systematic way (Sears & Zha, 2003). In line with this, the model is given procedural knowledge that guides the search process from top to bottom and from left to right. In cases where this does not lead to success, the model can try to find the letter with a random search.

The model stores the locations of certain letters in chunks in declarative memory (i.e. it learns their locations) and uses this knowledge to find the letters faster. It can access and use this information through direct retrieval. Every time a chunk is accessed or recreated, its activation increases, but it also decays with time when it is not accessed. The activation of a chunk has to be higher than a certain value, otherwise the information stored in the chunk is unavailable to the model. In the case of declarative memory, this minimum activation level is called the retrieval threshold. We set the threshold to a value which ensures that chunks can not be retrieved immediately, but have to be requested and updated a number of times before their information can be accessed.

At creation, a chunk's activation value is high enough to be accessed and retrieved directly. However, this value drops immediately to a value below the threshold after only 50 milliseconds. The chunk has to be requested and updated a number of times before its activation becomes high enough to be retrieved directly. Noise in the activation evaluation leads to differences in the chunk activation. The higher the activation of a chunk, the easier and faster it can be retrieved from declarative memory.

Direct retrieval is a way of speeding up the model's performance. However, if the model keeps using the same production rules, it will not show the highest possible speed-up: it does not learn to do the task itself in a more efficient way. Therefore, the model also starts

to take two rules together to speed up its performance, or learns to link a certain chunk from declarative memory with a rule that often calls for its retrieval. This is called production compilation (Taatgen & Lee, 2003). The process adds new rules to procedural memory, which function as shortcuts. For example, the model can return the first letter of a message to its parent "SMS entry" and at the same time initiate a new goal to "Search and Press", instead of waiting for "SMS entry" to initiate this goal after processing the first letter. The model could also initiate a motor action ("Press Button") at the same time as a retrieval (for example, the parent goal), instead of waiting for the result of this retrieval before moving on to the motor action.

New production rules are constructed immediately after a new combination of two successive rules has fired. However, the model cannot use these rules immediately. A production rule can only fire if its utility value is higher than the utility threshold. Its utility value is calculated from four factors: the chance that the production rule will lead to success, the value of the goal that it tries to reach, the time it needs to reach that goal and a noise factor. At creation, the utility of the new production rule is lower than the utility values of the two original rules. Every time the two original rules fire in sequence, the utility for the new production rule is updated, together with the values for the two original rules. At a certain point, the new rule may get a higher utility value than the original rules. If the new rule is fired and leads to success faster, its utility value becomes structurally higher than that of the two parent rules. It will then be chosen over its parents.

We define search time in our model as the time between the identification of the letter that is to be typed and the moment the model has identified the button with the letter. The ACT-R production trace shows at exactly which moment in time the model has identified the letter on the button and initiates a motor action. Our model's motor actions take a relatively long time, so a letter is located on the keyboard much earlier than the first button is pressed. For people we used a different definition for search time, because they do not have a trace of their memory use and the rules they follow. The definition of search time for people will be described in the next chapter.

Through the learning mechanisms described above, search time in the model is highly reduced. While at first the model only knows how to move its eyes from left to right and from top to bottom, the model later learns to look directly to the letter's location. Table 3.1 shows the model's average search times in milliseconds for the ten most common letters in the Dutch language at the beginning of the task and at the end. These letters are the "a", "d", "e", "g", "i", "l", "o", "r" and "t". Together, they account for 75.2% of the letters in the Dutch CELEX Lexicon. The times for the beginning of the task vary a lot because of the strict way of searching (from left to right and top to bottom). The search times for some letters, such as the "r", are very long compared to others ("a", "e", "n"). The relative position of the letters is a possible explanation for these deviations in search times. Because we read from left to right and from top to bottom, some letters will be viewed consecutively very often. The combination of "e" and "n" is a good example of this: if the bigram "en" (the most frequent bigram in the Dutch language) has to be typed, the letter "n" will be found very quickly after the "e". On the other hand, some letters are far apart on the keyboard and will thus have long search times between them. The search times at the end of the task are close together but not exactly the same. This is due to the fact that retrievals take an uncer-

tain amount of time, depending on their activation: the higher the activation, the faster the retrieval.

Table 3.1: Search times in the model (in milliseconds), see text for details

Letter	begin	end
a	3920	460
d	8060	500
e	3330	410
g	12110	630
i	5380	470
l	9440	550
n	3180	500
o	15550	490
r	17090	620
t	10940	510

The minimum search time was 400 milliseconds. This time was shown for most letters, and with sufficient practice probably every letter can be found at this speed. We based our first hypothesis (see Section 3.6) on this minimum search time. During these 400 milliseconds, the model retrieves the location of the button with the letter, moves its vision to this location and confirms that it is looking at the correct button.

We adjusted the retrieval threshold so that only the most active chunks are available for direct retrieval. Table 3.2 shows how many times (n) the ten most common letters in the Dutch language had to be typed before their locations were known. To avoid ambiguity, we define a letter's location to be known if it has been retrieved from declarative memory three or more times in a row. Noise in the activation evaluation can lead to rapidly changing activation values and therefore chunks may become temporarily available for retrieval, but we believe that this temporary availability is not the same as definitely knowing what a letter's location is. For example, in the case of the "n", the letter's location was retrieved the second time it had to be typed but was never retrieved again until the twelfth time. After that, it took another four times of typing and updating the activation value before the activation of the letter's location was high enough to be retrieved more than once. We believe that this is the moment at which the letter's location was really "known" for the first time.

Table 3.2 also shows the elapsed time (ET, in milliseconds) between the first occurrence of the letter and the moment the location of the letter was known.

As can be seen from Table 3.2, the average number of occurrences of a letter that is needed to learn its location is 13 for the ten most common letters. There was a period of on average 2,450 seconds between the first occurrence of a letter and the moment at which that letter's location was learned. This is the base for our second hypothesis (see Section 3.6).

The model typed two rounds of twenty-five sentences and five alphabetical sequences on both keyboards. The first round served as a practice round, because the model had no prior experience with typing. In this round, the model had to learn everything about text

entry and the locations of the letters. Therefore, the search times between the letters varied a lot (which can be seen in Table 3.1 as well). The second round was a better representation of the differences between the keyboards in text entry time, because the model could type more easily. After the second round on the alphabetical keyboard, the model was reseted and started on the new keyboard without any recollection of earlier experience. The total time to type the sentences on the original keyboard was 8,122 seconds the first round and 4,977 seconds the second round. On the new keyboard, time for the first round was reduced to 7,453 seconds and for the second round even further, to 3,661 seconds. The typing speed increase for the first round was 12%. As described above, the model's rigid search strategy in the beginning of the task lead to large and greatly varying search times for this first round, which made this round not a good representation of average search and entry times. The speed-up for the second, more representative round was 26%. Overall, typing on the new layout was 15% faster than on the alphabetical layout after these two rounds. This speed-up may even increase further with practice, until it reaches the theoretical speed-up of 30% that was calculated using Fitts' Law (MacKenzie & Soukoreff, 2002). The sentences that were typed are listed in Appendix C.1.

The activation of a chunk, and therefore the chance of a successful retrieval, depends solely on the number of times the chunk has been requested and updated and the time between those updates. Therefore, we believe that only the number of prior occurrences of a letter in a text determines the speed with which its location on the keyboard will be learned. All other factors, such as the number of letters on a key, or a letter's string number, do not have an effect on the speed with which search times for a letter decrease with practice. Our Hypotheses 3, 4 and 5 (see Section 3.6) are in line with this.

In Section 3.6 we formulate the six hypotheses that are based on the results of our model. In these hypotheses, the terms "goal button", "search time" and "entry time" are used. The goal button is the button with the correct letter on it. Search time in the model can be read directly from the trace. However, in the experimental data, search time can be measured the most accurately by defining it as the time between the last press on the previous button and the first time the goal button is pressed. Unfortunately, participants have no output trace

Table 3.2: Learning moments in the model (number of occurrences (n) and elapsed time (ET), in milliseconds)

Letter	n	ET
a	16	2,955,000
d	20	3,950,000
e	5	368,000
g	11	1,729,000
i	8	1,274,000
l	15	2,980,000
n	14	1,988,000
o	15	2,256,000
r	16	4,615,000
t	13	2,403,000
$\tilde{X}$	13	2,452,000

and therefore this is the most accurate way of measuring search times. We believe that the results will not be very distorted by the different definitions for search times, because the ratio of the differences between the two keyboards will stay the same. Finally, entry time is defined as the time between the last press on the previous button and the moment the correct letter is shown in the display.

3.6 Hypotheses

Evaluating the results of the ACT-R model leads to the formulation of the following hypotheses about the speed and search strategies of typing on the original and the new keyboard:

Hypothesis 1 on the minimum search time: *The minimum search time is 400 milliseconds.*

Hypothesis 2 on the speed of learning the letters on the new keyboard layout: *It takes on average thirteen occurrences of a letter to learn the position of that particular letter on the keyboard.*

Hypothesis 3 on the factors that influence search time: *Letters that have been typed more often than other letters will be found more quickly, resulting in smaller search times for next occurrences.*

Hypothesis 4 on the factors that influence search time: *The string number of a letter on a key has no impact on the search time for that particular letter.*

Hypothesis 5 on the factors that influence search time: *The number of letters on a key has no impact on the search times for the letters on that key.*

Hypothesis 6 on the comparison of the original and the new keyboard layout: *After enough practice, users type about 30% faster on the new keyboard than on the original keyboard.*

The next chapter will describe the experiment method.

Chapter 4

Experiment method

We tested the hypotheses drawn from the model's results. By testing the assumptions about the factors that have an effect on the search times for specific letters, the minimum search time and the comparison of the two layouts are correct, we will assess the validity of our model.

4.1 Participants

Eight participants with ages ranging from nineteen to twenty-two were recruited, all from the Rijksuniversiteit Groningen. All participants had some experience in text messaging on a standard mobile telephone keyboard. The data from the first two participants were not used for analysis because of problems with the input device.

4.2 Apparatus

A similar interface to the one used for our ACT-R model was used in this experiment. The only adjustment was the box on the bottom of the screen showing the sentence or alphabet sequence to be typed. This box remained visible throughout the task to reduce dual-task interference from having to memorize the sentences. Input was given by mouse clicks.

4.3 Procedure

Participants had no knowledge about the methods used to construct the new telephone keyboard layout, but they were told that the new layout was designed using a "smart way" of placing the letters on a keyboard. The participants were asked to type four sentences on the alphabetically ordered telephone keyboard and twenty-seven sentences on the new telephone keyboard. The sentences were taken from a short children's story. Five alphabet sequences were presented at fixed points between the twenty-seven sentences that participants had to type on the new layout. These sequences were placed after the seventh, the twelfth, the sixteenth, the twenty-fourth and the twenty-seventh sentence. The full set of sentences and sequences is listed in Appendix C.1. The moment at which a key was pressed was recorded and written to an output file, along with the appropriate participant number, the key number, the sentence to be typed, the letters entered so far in the sentence and the

length of the string in the display at that moment. After completing the task, the participants were asked to fill in a questionnaire about their experiences. This questionnaire is added to this paper as Appendix D.1. The total experiment time per participant was about one hour, including instructions and post-test questionnaire. We believe that testing for one hour gives enough information about the learning rate and search strategies in people to test the model's validity. Furthermore, stretching the experiment time will probably have a negative effect on the participants' concentration and could therefore lead to biased data.

Participants received instructions to type the sentences as fast as possible but with as few mistakes as possible. They were instructed to correct errors immediately, but if an error was discovered after some new letters had been typed, they could leave it in the text. We chose not to give these mistakes a major role in the analysis, because a pilot experiment showed that the percentage of mistakes was small. This was also true for the final experiment, where 1.2 % of the total of typed letters was an error of which only 0.2 % was caused by pressing the wrong button when trying to type a letter. The rest of the errors was caused by pressing the correct button a wrong number of times or pressing the Backspace key when the Space key was intended. Because the number of mistakes was so small, we believe that these mistakes would not alter the structure of the typed text to such a point that it was no longer representative for the structure of the Dutch language. Therefore, the speed with which letters were found throughout the experiment was probably not affected by these mistakes. Furthermore, as long as the intention is to locate and type a particular letter, the search time for the next occurrence of that letter will most probably not be affected by the mistakes.

4.4 Analyses for testing the hypotheses

The data from six participants were analyzed to test the hypotheses presented in Section 3.6. Below we describe how these hypotheses were tested.

Hypothesis 1: *The minimum search time is 400 milliseconds.*
To test this hypothesis, we selected the smallest search time for each participant from all the search times.

Hypothesis 2: *It takes on average thirteen occurrences of a letter to learn the position of that particular letter.*
To test this hypothesis, first the minimum search time for every user had to be established. This time reflects the fact that the user knows directly where the letter is positioned on the keyboard. Then the data were searched for all times within a predefined range of this minimum search time. We set this range to 368 milliseconds: this is the time that Fitts' Law gives for moving the finger from the upper left corner to the lower right corner. This difference in movement time has to be taken into account to ensure that no specific combinations of keys are selected over others because of their relative (closer) positions. The data were analyzed for the first entry time within this range, to find the moment at which a letter was no longer searched for but was located directly. After that, all prior occurrences of the letter were counted. This is the number of occurrences needed to learn the position of a specific letter on the keyboard.

Hypothesis 3: *Letters that have been typed more often than other letters will be found more quickly, resulting in smaller search times for next occurrences.*
To test this hypothesis, two letters with the same string number on a key but different frequencies in the experimental text were compared. We chose the letters "n" (frequency 99, key "nmf") and "g" (frequency 35, key "gvs") for this comparison. Both letters have string number 1 on a key with three letters. The analysis was conducted using a paired t-test, comparing the search times for both letters at the beginning and end of the task.

To make a valid comparison between the search times for the letter "n" and the letter "g", we had to determine the search times at fixed points within the experimental sentences. The alphabetical sequences are not right for this, because the fixed order of the letters in the alphabet influences the search times. For instance, the "n" is always preceded by the "m", which is located on the same key. This means that the user always has to wait a second before pressing the same button, but searching is most probably not necessary. We therefore decided to use only the sentences and not the alphabetical sequences for the comparison between these two letters.

We calculated the average of the first two and the last two occurrences of the two letters. The variance in search times can become high as a result of only a few extreme values. These extreme values can arise when, for example, the participant looks at the sentence at the bottom of the screen instead of searching for the next letter on the keyboard. By averaging two search times per letter, these extreme values are evened out.

Hypothesis 4: *The string number of a letter on a key has no impact on the search time for that particular letter.*
To test this hypothesis, two letters with the same frequency in the text but with different string numbers were compared. We chose the letters "u" (string number 2 on "auy") and "f" (string number 3 on"nmf"), with respective frequencies of 20 and 19, and the letters "r" (string number 1 on "rjc") and "s" (string number 3 on "gvs"), with respective frequencies of 47 and 46. The analysis was done with a paired t-test, like for the comparison for Hypothesis 3. We again used the average of the first two and the last two occurrences of the letters.

Hypothesis 5: *The number of letters on a key has no impact on the search times for the letters on that key.*
To test this hypothesis, two letters with the same frequency but with a different number of letters on their respective keys were compared. For this comparison we have chosen the letters "x" (key "oix") and "q" (key "eq"), which both have zero occurrences in the text. Both letters occur only in the alphabetic sequences. We compared the search times for these two letters in the alphabetical sequences throughout the task. The analysis was again conducted using a paired t-test.

Hypothesis 6: *Typing on the new keyboard is 30% faster after practice than typing on the original keyboard.*
To test this hypothesis, the search times of a set of letters at the end of the task on the new layout were compared to the search times of these letters at the end of the task on the alphabetical layout. Because some letters might not have been learned, we decided to compare the typing speeds for the ten most common letters in the Dutch language: "a", "d", "e",

"g", "i", "l", "n", "o", "r" and "t". These letters together account for 75% of the letters in the Dutch CELEX Lexicon and 69% of the letters in the sentences that participants had to type on the original and the new keyboard layout. Their respective search times on the two layouts and the average of their differences will give us good insight in the differences in typing speed on the two layouts.

Chapter 5

Experiment results

This chapter reports the results of the experiment: the minimum search times, the speed with which letters are learned, the factors that influence search time and the relative speed-up compared to the alphabetically ordered keyboard. The participants' answers to the questionnaire are also reported. Explanations for differences that may arise between the model's results and the participants' behavior will be given in Chapter 6 on the comparison between the model and the participants.

5.1 Minimum search time

The average minimum search time for the participants was close to 300 milliseconds. The minimum search times ranged from 200 milliseconds to 380 milliseconds (M= 285.2, sd= 73.2). Four out of the six participants showed a minimum search time when typing the bigram "de", the third most frequent bigram in the Dutch language, and the other two when typing the bigram "en", the most frequent bigram. "De" and "een" are Dutch articles and the three letters "d", "e" and "n" are placed directly next to each other on the middle row of the keyboard. This may explain why these bigrams were typed the fastest. Appendix E.1 shows the minimum search time per letter for each participant with the elapsed time in the experiment when they had their minimum search time, together with the average search time and standard deviation for that letter. The absolute minimum search times are printed in bold.

5.2 Learning the location of a letter

Using the operationalization as described in Section 2.4, we selected the moment at which participants had learned the locations of different letters. The letters that the participants learned the quickest (measured from the first occurrence of the letter in the text) were the "e", after on average 200.0 seconds (sd= 137.2) and sixteen prior occurrences (sd= 7.7) and the "n", after on average 490 seconds (sd= 408.0) and fourteen prior occurrences (sd= 6.3). The difference between the number of occurrences that is needed could be explained by looking at the relative positions of these letters on the keyboard. The most frequent bigram (when bigrams of one letter and the space are not taken into account) in the Dutch language is "en". The letter "n" is placed directly right to the letter "e" on the keyboard. Because

we read from left to right, the first button we see after looking away from the button with the "e" is the button with the "n". When an "e" is typed, the button with the letter "n" is logically the next button to look at. This fact might account for reduced search times between the "e" and the "n". Appendix E.2 shows for the ten most common letters in the Dutch language the elapsed time and the number of occurrences before the location of that letter had been learned. The participants learned the locations of these letters after on average 15.4 occurrences in the text (sd= 10.0).

5.3 Factors that influence the speed-up for search times

Our hypotheses are that the frequency of a letter in a text does have an influence on the search times for that letter, and that its string number or the number of letters on its key do not. This section will describe the results of our experiment data with regard to these hypotheses.

5.3.1 Frequency of a letter in a text

We expected to find that highly frequent letters showed a bigger overall speed-up in search times than less frequent letters, because retrieval time depends on activation and the activation of more frequent letters is updated more often. Our experiment data showed that the overall speed-up in search times for the letter "n" (frequency 99, M= 855.2, sd= 595.4) was significantly different from that for the letter "g" (frequency 35, M= 1776.4, sd= 1114.3), $t(5)= 4.37$, $p= .01$. However, contrary to our expectations, not the search times for the letter "n" but those for the letter "g" showed a significantly greater overall speed-up. The search times for the letter "n" were faster than those for the letter "g" throughout the whole experiment for all participants, as can be viewed from Figure 5.1 below.

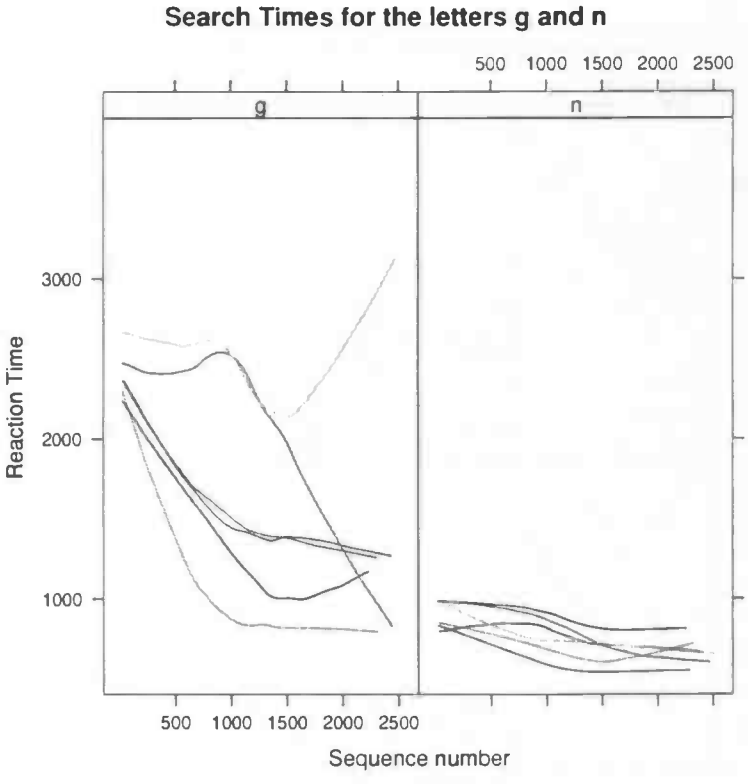

Figure 5.1: Search times for the letters "g" and "n"

The low overall speed-up seen for the letter "n" might be due to a floor effect. We expected that the search times in the beginning of the experiment would be similar for all letters. Obviously this is not the case, and the location of the letter "n" was learned very quickly by all participants. Figure 5.2 shows the search times for the nine most frequent letters of the nine keys (string number 1) on the new keyboard throughout the experiment.

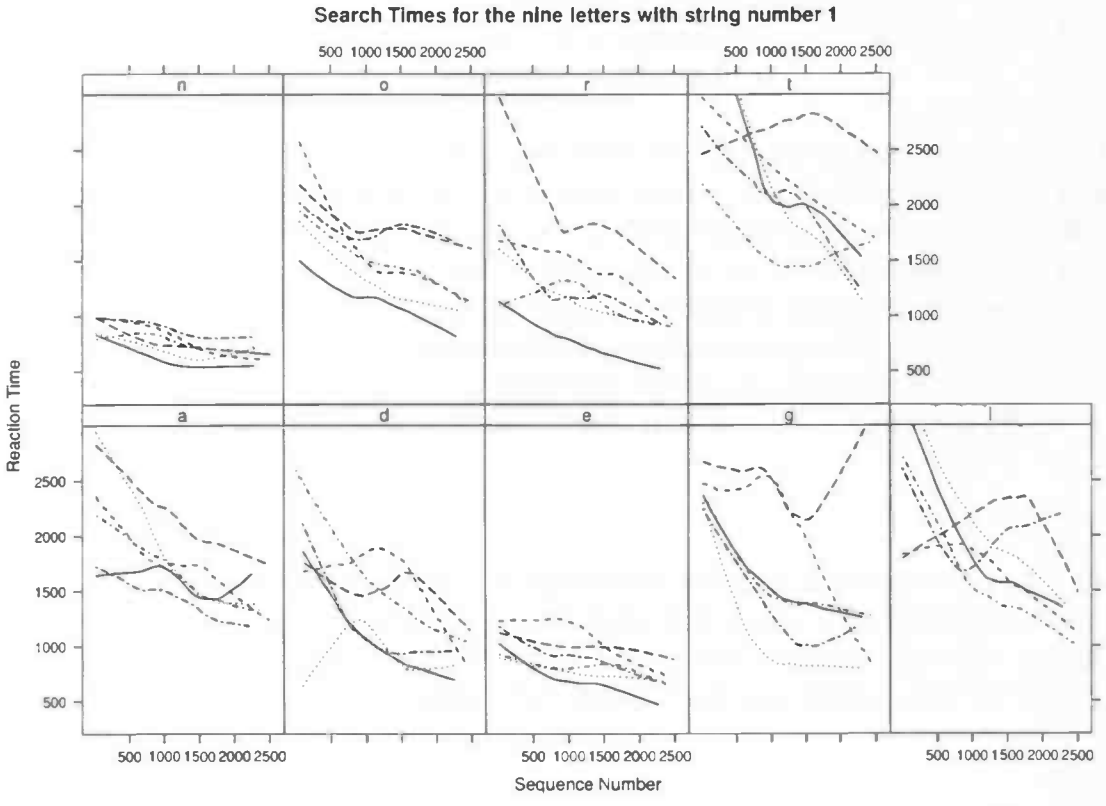

Figure 5.2: Search times for the nine letters with string number 1 on the new keyboard

From Figure 5.2 the floor effect for the letter "n" is very clear. The search times for the beginning of the task are already close to the minimum search time, so they could not decrease as much as those for the other letters. Another letter that seems to show this floor effect is the "e". We expect that the letter "e", which is the most frequent letter in the text, will show an even bigger floor effect and therefore a smaller speed-up than the "n". However, the letter "e" (M= 1055.1, sd= 633.1) has a significantly greater overall speed-up than the "n" (M= 855.2, sd= 595.4), t(5)= 6.76, p= .00. This suggests that more frequent letters, even when they have a floor effect, show a greater overall speed-up in search times throughout the task.

To determine whether the frequency of a letter in a text influences the speed-up in search times for that letter when there is no floor effect, we compared another set of letters. These again have string number 1 on a key with three letters, and they are both located on a corner of the keyboard. These two letters are the "g" (frequency 35) and the "l" (frequency 50). The letter "l" (M= 1956.5, sd= 1051.7) was indeed learned significantly faster than the letter "g" (M= 1776.4, sd= 1114.3), t(5)= 4.20, p= .01. Therefore, frequency does have an effect on search times when there is no floor effect.

A possible explanation for the floor effect of the the letters "e" and "n" is their location on the middle row. However, in Figure 5.2 the letter "d" does not show an apparent floor effect. Also, the speed-up for the letter "d" (M= 1518.4, sd= 1059.7) is significantly greater than that for the letter "n" (M= 855.2, sd= 595.4), t(5)= 4.01, p= .01, despite its lower frequency. Another explanation could be the relative locations of the letters. The letters "e" and "n" are easy to find, on the middle row and directly next to letters that often precede

them. The bigram "en" is the most frequent bigram in Dutch language and "de" is the third most frequent bigram. Because we read from left to right, the first place one would normally look after typing a letter is to the button directly right to it.

Appendix E.3 lists the absolute speed-up per letter for each participant and Appendix E.4 lists the relative speed-up compared to the search time at the beginning of the task. As can be seen from these tables, the letters show big differences in speed-up within the task. Less frequent letters mostly show a great average speed-up but also have high standard deviations, indicating that the differences between participants were big. Some participants apparently learned different letter locations than other participants. On the other hand, some of the more frequent letters show a overall low speed-up. These letters were all learned very fast in the beginning of the task and their low speed-up is due to a floor effect.

5.3.2 String number

The string number of a letter does not have a significant effect on the search times for that letter: the decrease in search times for the letter "r" (M= 1332.7, sd= 878.2) did not show a significant difference from that of the letter "s" (M= 1941.4, sd= 1120.2), t(5)= 1.01, p= .36. The effect on the search times for the two letters "u" (M= 2176.2, sd= 1553.8) and "f" (M= 1547.7, sd= 1231.9) was not significantly different, t(5)= .03.

5.3.3 Number of letters on a key

We also found that the number of letters on a key does not influence the speed-up in search times for the letters on that key. The difference in search times between the beginning and end of the task for the location of the letter "q" (M= 1063.0, sd= 871.9) was not significantly different from this difference in the search times for the location of the letter "x" (M= 1031.4, sd= 747.6), t(5)= .40. The locations of the letters on two-letter key were thus not learned significantly faster or slower than the locations of the letters on the three-letter key.

5.4 Typing speed on the original layout versus the new layout

The participants typed faster on the new keyboard than on the original keyboard. The entry times for the ten most common letters in the Dutch language were 17.1 % shorter on the new layout. They showed this speed-up after typing twenty-six sentences. Four out of the six participants vocally reported that they had difficulties switching to the new keyboard after having worked on the original keyboard. The participants had tried to type the letters as fast as possible on the alphabetical layout and had "(re)activated" their knowledge about the locations of these letters on the original keyboard. When they had to switch, this knowledge may have interfered with learning the locations of the letters on the new layout. This might have affected their final typing speed in a negative way. In a future experiment, this setup could be changed to eliminate this type of interference.

Table 5.1 contains the average entry times for the ten most common letters on the original and the new keyboard throughout the whole experiment.

Table 5.1: Average entry times per letter

letter	alphabetical	new
a	877.8	1412.1
d	1347.3	1149.9
e	1460.8	872.8
g	1285.0	1548.4
i	2113.6	1460.3
l	1771.6	1719.3
n	1630.2	839.8
o	2010.2	1390.5
r	2229.7	1144.2
t	1681.3	2063.3
$\bar{X}$	1640.7	1360.1

5.5 Questionnaire

Participants received a post-test questionnaire. This questionnaire contained questions about their experiences with the new keyboard. None of the six participants found the unfamiliar distribution of the letters over the keyboard disturbing for the task. Also, five participants reported that the letters "d", "e" and "n" were easy to locate. Only one participant did not mention the "d" but he did describe the "e" and the "n" as very easy to locate. Furthermore, the letters "o" and "i" and the "q" were easy to locate for most participants. On the other hand, many subjects reported that the letters on the left and middle of the upper row ("gvs" and "tbz") and the right lower button ("lwh") were relatively hard to locate on the keyboard. These findings suggest that people link their representation of a letter to the information about a button string and information about this button to a location on the keyboard. The questionnaire is added as Appendix D.1. The raw data for the questionnaire responses can be found in Appendices D.2 to D.9.

Chapter 6

Comparison between model and experimental data

In Chapter 4, we formulated six hypotheses from the results of our model. The results from the experiment from Chapter 5 serve as a test for our hypotheses and thus the validity of our model. In this chapter, we will discuss the similarities and differences between the results and the hypotheses. We will then show how some parameters settings in our model can be altered to make valid predictions about human learning behavior concerning SMS. First we will list all six hypotheses and their outcomes and we will then discuss the differences in turn.

6.1 Hypotheses and results

In this section, we will compare all six hypotheses to the results from the experiment.

Hypothesis 1: *The minimum search time is 400 milliseconds.*
This hypothesis is not true. We found that the model shows a larger minimum search time than any of the participants. Table 6.1 shows the minimum search times (min ST, in milliseconds) and the letter for which this minimum search time was reached, for the six participants and the model.

Table 6.1: Minimum search times for participants and model

user	min ST	letter
participant 3	200	e
participant 4	321	e
participant 5	200	e
participant 6	330	e
participant 7	391	n
participant 8	200	n
model	400	all letters

Recall that the time to type the whole set of sentences once was 7,453 seconds (about 2 hours and 7 minutes) for the model (see Section 3.5). The participants did the same task

within one hour. Both the minimum search time and the total typing time were therefore approximately twice as long for the model as for the participants.

Hypothesis 2: *It takes on average thirteen occurrences of a letter to learn the position of that particular letter on the keyboard.*
With regard to this hypothesis, we found that the model's behavior is mostly representative for the experiment data, but with two deviations. The model learned eight out of the ten tested letters at a moment that is within a 95% confidence interval for our experiment data. Figure 6.1 shows these confidence intervals for the experiment results, together with the model's data.

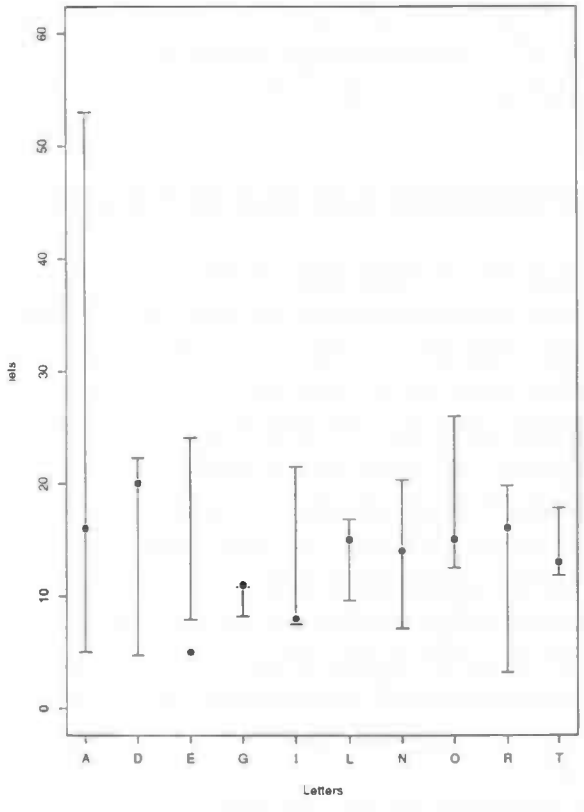

Figure 6.1: Confidence intervals for the ten most common letters in the experiment data

The number of occurrences for eight of the ten letters were within the range of the 95% confidence intervals that were constructed from the experiment data. The letter "e" was learned a little faster and the letter "g" a little slower.

Hypothesis 3: *Letters that have been typed more often than other letters will be found more quickly, resulting in shorter search times for next occurrences.*
The effect of frequency of a letter on the difference in search times between the beginning and the end of a task is in line with what we expected. However, there were some exceptions. The "e" and "n" showed a significantly smaller overall speed-up than other less frequent letters. In these cases a floor effect can be seen: the location of the letter is learned

so fast that the overall speed-up is small.

Hypothesis 4 and 5 on the factors that influence search time: *The string number of a letter on a key has no impact on the search time for that particular letter* and *The number of letters on a key has no impact on the search times for the letters on that key.*
The string number of a letter or the number of letters on a key does not have a significant influence on the search times in either the model or the participants. With regard to these hypotheses, the experimental data are in line with the data from our model.

Hypothesis 6: *After enough practice, users type about 30% faster on the new keyboard than on the original keyboard.*
We found that the speed-up for the new keyboard layout compared to the original layout was 17.1% for the participants. The model only showed a speed-up of 12 % the first round of typing, but the second round was more representative. The speed-up in this round was 26 %, leading to an overall speed-up of 15 %. This is close to the participants' speed-up.

6.2 Possible explanations for the differences

With regard to the moment at which the model learned the location of a letter, there were two deviations: the "e" (learned too soon by the model) and the "g" (learned too late by the model). For the "e" we can say that the time between the first occurrence and the fifth occurrence, at which its location was learned, was very small (only 368 ms) and that the activation value did not decay as fast as it was updated. We gave the model another set of sentences to type, in which the first ten occurrences of the letter "e"' were farther apart. With this set of sentences (which are listed in Appendix C.2), the model learned the location of the letter "e" after 9 occurrences. However, the letter "k", which occurs often in the first sentence, was now learned very quickly. This shows that the model does not make a qualitative distinction between letters. We do not know for sure whether people make this qualitative distinction, but the floor effect for the two most frequent letters "e" and "n" does suggest this. These distinctions may result from different factors than the distinction made by the model, which is based on occurrence intervals.

The differences in total typing time for the set of sentences are possibly due to the fact that our model still handles the subtasks mostly serially, whereas people might learn sooner to do the different subtasks in a more parallel way. For example, some participants reported that their finger moved to the location of a letter on the keyboard, although they had not found the location with their eyes yet. This points to a more parallel processing in human cognition. One of the learning processes in ACT-R, production compilation, leads to more parallel handling of subtasks. However, the degree to which (sub)tasks are handled at the same time is not the same in our model as in people. Motor actions and attention are still processed more serially in our model. This might be due to the fact that the model has not had enough practice with typing and with searching on the keyboard: the participants did not have to learn to press a button, whereas our model had to start as a total novice. After a certain amount of practice these differences in search and entry time may disappear.

An explanation for the delay in the model's search times can be found in the fact that

our model needs more time to process all the different subtasks, while people seem to do these things in one step. One example is the way in which our model decides whether the desired letter is located on a specific button. It initiates a new subgoal, reads the string on the button, then checks if the letter is in that string and if it is, it returns the string number to the parent task. This (sub)process probably takes longer than it does in humans. This leads to the conclusion that the implementations for some subprocesses may be too complex in our model.

With regard to the effect of the frequency of a letter on its search times, the most obvious difference is the floor effect seen in the letters "e" and "n". This is probably due to their special characters as the most common consonant and vowel in the Dutch language and their locations next to the letters that most often precede them. People will most probably use their knowledge about context in a task, but our model does not have any knowledge of the Dutch language and its most frequent letters.

The overall speed-ups in the participants and the model were almost similar. The fact that the speed-up in the second round for the model was higher than the speed-up for the participants might very well result from the fact that the participants' knowledge of the alphabet interfered with their learning process after this (relatively small) amount of practice. This could be solved by either giving the participants more practice before measuring their search times, or by showing them the new keyboard first and the original keyboard after. This could be tested in a future experiment.

6.3 Changes to the model

Our model will be able to give valid predictions after three adjustments: the model's minimum search time should to be shorter, it should need less time to type the whole set of sentences and it should either have context knowledge or practice more.

By lowering the parameter that controls the initial cost added to a newly created production rule, and by setting the speed of learning through production compilation to a higher value, a decrease in minimum search time and in the total time necessary to type the full round of sentences are achieved. These changes result in a steeper learning curve.

The effect of frequency on search times is different for some letters in the model and the experiment results. To make this difference smaller, context knowledge could be added (to create the same floor effect as in the experiment data) or the model should practice more, to ensure that entry time is more important than search time. Adding context knowledge to the model is outside the scope of this paper. We will therefore give the model more time to practice, so that it reaches the level of an expert and context knowledge is no longer important to reach a higher typing speed.

These changes together lead to a model that has a high representative value for human behavior. The minimum search time is reduced to 300 milliseconds after four rounds of practice, which is much closer to the experimental data. Also, the total typing time for a full round of sentencces is reduced, although it is still longer than was found in the experimental

data. The ratio between minimum search times in model and in people is the same as the ratio between the total typing times for the two groups. We will therefore use this model to answer our research questions in the next chapter.

Chapter 7

Discussion and conclusions

7.1 Evaluation techniques

When evaluating a new text entry device in an experimental setting, many factors have to be considered. According to Zhai et al. (2004), the evaluation task should at least consider the following factors: (1) the ultimate performance, (2) error and error handling, (3) models as performance measures, (4) initial performance vs. longitudinal learning, (5) visual and cognitive attention, (6) delight, (7) task validity and (8) linguistic fidelity. These factors are described in this section with regard to our evaluation of the new keyboard.

(1) the ultimate performance. Obviously, a layout evaluation should include the level of performance that users can reach and a clear comparison to the performance level for another layout. Our cognitive model showed that the speed-up compared to the alphabetical layout is around 30 %. This factor is therefore included in the evaluation.

(2) evaluation of error and error handling. We included error evaluation in our device evaluation task. Only few errors were made on the new layout. The total percentage of errors was 1.2 %, with the most mistakes being pressing the correct key an incorrect number of times. Only 0.2 % of the mistakes was caused by pressing the incorrect button. The mistakes can be corrected easily with the large Backspace key. However, some participants mistook the Backspace key for the Space key. This issue might require further evaluation.

(3) validity and completeness of the model as performance measure. Zhai et al. (2004) report that "we do not know enough about human perceptual motor and cognitive skills to form strong quantitative laws as a foundation to theoretical performance prediction. Understanding atomic human performance regularities is undoubtedly important." Our cognitive model does not only focus on the theoretical movement time, but also includes motor and visual actions. Obviously, the model is based on assumptions about these visual and motor actions, but the experiment showed that our model leads to valid predictions about initial user performance. We therefore believe that our model is complete and that the evaluation is representative for user performance.

(4) the effects of initial performance vs. longitudinal learning. We tested the initial performance of users in an experiment and based the predictions about longitudinal learning on the results from the cognitive model. Both initial performance and longitudinal learning

are therefore considered in the evaluation. Another important factor to include in the evaluation of a device is its initial usability. Initial usability has a huge impact on the willingness of people to start to use the device and to learn about its advantages. Most people are not willing to invest much time in learning to use a new device when a familiar alternative is available (Zhai et al., 2004). All of our participants reported that they did not find the unfamiliar distribution of the letters disturbing for the task, so the usability after less than one hour of typing was sufficient. This factor was therefore also included in the evaluation of the new layout.

(5) visual and cognitive attention. A good text input device requires as little visual and cognitive attention as possible, so all attention can be focused on the use of the device itself. This factor requires further research.

(6) delight. A new device should not only be easy but also "fun" to use. We tested this factor in the questionnaire. Our participants reported that they enjoyed using the new keyboard layout in the simulation environment. On a scale from 1 to 5, the average score on "fun of the task" was 4.2. (see Appendices D.2 to D.9 for the participants' answers on the questionnaire). However, this does not tell us much about the fun people would have with the new layout outside the experimental task, which obviously is hard to measure in an experiment.

(7) evaluation of task validity. The evaluation task that our model performed was to type pre-prepared text. These "text copy tasks" are commonly used for evaluation (Soukoreff & MacKenzie, 2002, p.11) but the actual usage of an input device is different because a user normally creates his own texts. Users may prefer difference letter combinations or specific words or abbreviations, which could lead to a different speed-up than was shown in the experiment. Furthermore, performing a text copy task adds additional attention demands to the task (MacKenzie & Soukoreff, 2002). A user has to look at the text, at the keyboard to move to the correct location and at the display to check the input. On the other hand, new difficulties arise with text creation tasks. MacKenzie and Soukoreff (2002) report that in a text creation task, text entry speed is harder to measure, mistakes are harder to identify and control over the distribution of the letters and words entered is lost. That is why we chose to use a text copy task as the evaluation task.

(8) evaluation of linguistic fidelity. The texts that the model had to type are not completely representative for the structure of the Dutch language, because some letters did not occur in their exact actual proportions. However, the occurrence rates between the letters were mostly correct.

The overall evaluation of the new layout included most of the factors as listed by Zhai et al. (2004). This evaluation was insightful about the performance and usability of the new layout. The results regarding usability and performance are positive, leading to the conclusion that the new layout is a success.

7.2 Experience and performance

In the research that we conducted, the cognitive model started as an absolute novice: it had no prior experience with either typing, the Dutch language or the alphabet. Our participants all had extensive experience with these three factors. This difference in experience may have lead to different levels of (initial) performance. Firstly, learning to move the finger to the right location and learning to press the goal key the appropriate number of times requires practice and therefore influences initial performance. Secondly, knowledge of the language might influence the way one looks at the keyboard (one could, for instance, implicitly pay more attention to more frequent letters and learn their locations faster). Finally, knowledge of the alphabet influences the way users look at the alphabetical keyboard. The model can not adjust its search strategy to the alphabetical ordering, which people can.

7.3 Use on a real telephone keyboard

Our research has mainly focused on the effects of a new keyboard layout in an experimental setting, where the keyboard was shown on a computer screen and buttons were pressed by clicking the mouse. However, real mobile telephones are not handled by mouse clicks but by thumb movements on the keyboard. This leads to one major difference between the way mobile telephones are used in real life and the way they were used in our experimental setup: on a real mobile telephone, view of the keyboard is blocked by the thumb. This might lead to a different learning strategy, particularly because it is more important to learn the letter's locations faster than when every part of the keyboard is always visible. We have shown that users can reach a 30% speed-up on the new keyboard layout, and this will most probably not change. Futhermore, we have shown that it is possible to learn how to handle the new keyboard very quickly: within 45 miutes of practice, all of our participants have learned a number of locations of letters and report no major difficulties with the new layout. Finally, our model shows that after some practice, the time that is needed to locate and press a button is no longer dominated by a search process but is almost solely devoted to the recall and check of a letter's location. These factors together lead us to the conclusion that the new layout can be succesfully used in real mobile telephones.

7.4 Future research

An interesting topic of future research is the possible time gain that predictive text input techniques, such as Tegic's T9[1], could give in addition to this new layout. T9 reduces the number of key strokes per character substantially, as does our new layout. The combination of both could possibly lead to an even larger decrease of text entry time.

Adding familiar sequences to the model's declarative memory could also lead to new insight in the search strategies that people use when confronted with keyboards with a familiar distribution of letters. For instance, the effects of the use of Sholes' layout for mobile text entry (e.g., Green, Kruger, Falgu & St. Amant, 2004) could be determined using such a

[1]Description available at http://www.t9.com

cognitive model.

The answers given in the questionnaire suggest that people link their representation of a letter to the information about a button string (i.e., they link the letter "e" to the button "eq") and information about this button to a location on the keyboard. For instance, mostly letters from the same buttons, especially those on the upper left corner and the lower right corner, were hard to find for the subjects. These letters have no obvious common attributes except their locations. Future research might give more insight in this process. This information could possibly be used to create an even faster layout, because information about search strategies can help to distribute the letters in an even more efficient way.

7.5 Conclusions

We have presented a new, optimal layout for text entry on mobile telephone keyboards. This layout was based on the structure of the Dutch language, resulting in lower text entry costs for Dutch texts. A cognitive model was constructed in ACT-R to examine the practical results of this optimal keyboard. The validity of this model was tested in an experiment, leading to small changes in the parameter settings and the amount of "practice" that is given to the model before using it for insight in human behavior. Four research questions were listed at the beginning of this paper (Section 1.2). In this chapter, we will use the results of our cognitive model with respect to these research questions.

(1) How long does it take before users know where a certain letter is located on an unfamiliar keyboard layout?

It takes on average thirteen occurrences of a letter in a text to learn to position of that particular letter on the keyboard. The locations of letters that are typed often in a short period of time will be learned faster than those of letters that stretch the same number of occurrences over a longer period of time.

(2) Which mechanisms can explain this learning process?

People can learn the locations of the letters and can recall them during their search for that letter. Furthermore, they learn to create and use shortcuts in the way that they execute the task. There are different types of short-cuts, but a common feature is their higher level of parallel handling of (sub)tasks. The following types of short-cuts can be identified: (1) motor actions, such as pressing a button, are performed at the same time as attention tasks, such as determining how many times that button still has to be pressed; (2) visual actions, such as searching for a letter on the keyboard, an motor actions, such as moving the finger to a location that the eye is looking at, are combined; and (3) two attention tasks are combined, for instance determining the next letter to be typed and trying to recall the location of that letter. These three types of sort-cuts can lead to great overall speed-up in the task execution and can eventually bring the user to an expert-level of performance.

(3) Which are the search strategies that people use to find letters on an unfamiliar layout?

An unfamiliar distribution of the letters on a keyboard forces the user to search in a systematic way (Sears & Zha, 2003). Therefore people will mostly search from left to right and from top to bottom in the beginning of the task. After some time, they have learned the locations of some letters and look directly at those locations when looking for that particular letter. When these two strategies do not lead to success fast enough, random search is a third option to find the letter's location.

(4) To what extend is a new layout which is faster in theory, also faster in practice, when compared to the original alphabetical layout?

The speed-up in text entry that can be reached by using the new layout is 30 %.

Our overall conclusion with respect to the new layout is that it is more efficient and that learning to make full use of it was done unexpectedly fast.

Appendix A

Frequencies of bigrams in the Dutch CELEX Corpus

These are the bigrams and their frequencies from the Dutch CELEX Lexicon (Baayen et al., 1993), ordered from least frequent to most frequent. Spaces are denoted by ␣

py	1	ay	15	gm	57	ow	153
xk	2	gy	16	z␣	57	tc	158
cz	2	ya	16	aj	58	kg	172
aq	4	xy	18	yn	66	mf	175
ry	4	dc	18	gj	68	ux	179
gf	4	uo	18	bd	70	v␣	180
kb	4	yd	18	sr	72	eq	185
ii	4	kf	20	wv	76	iu	186
iq	4	ao	21	pf	76	oj	190
fc	4	hl	22	xi	80	aw	200
yt	4	by	24	ye	82	ym	212
wb	6	zz	26	␣x	85	ck	213
yb	6	ox	26	ny	90	cl	225
ih	8	gc	28	gp	92	tf	236
df	8	oh	28	my	92	sq	248
ey	10	xc	30	yc	113	ah	252
km	10	hm	30	bj	114	jb	254
hn	10	fy	30	mw	115	jh	272
lc	10	xu	35	fp	134	cu	277
mn	12	jr	36	ix	138	sw	278
uq	12	␣y	38	wk	142	dm	278
wp	12	sz	39	␣q	142	ax	290
yl	12	kv	40	db	144	sy	316
wl	13	sf	40	bt	144	dz	339
mc	14	dn	50	ys	144	ct	358
wh	14	xa	52	y␣	145	uv	370
q␣	14	hy	52	oy	151	jw	384
nq	14	oq	56	fj	152	lb	392

nj	420	fk	1342	ab	3023	tz	7198
dp	423	ej	1380	sk	3086	ip	7250
x␣	484	lz	1386	su	3162	tk	7580
xe	515	pn	1418	nm	3223	kn	7623
c␣	527	ca	1430	sj	3414	av	7722
kz	536	gn	1452	fu	3528	vu	7800
ib	542	mb	1459	lk	3529	th	7846
fm	555	fv	1483	pz	3550	sm	7853
ua	592	sh	1539	au	3597	ds	8118
yz	610	np	1594	fr	3871	tb	8239
dl	617	um	1615	oa	3962	gl	8433
lw	641	h␣	1633	␣c	3973	ja	8559
ly	642	sb	1650	so	3987	fs	8683
qu	651	pk	1658	oz	4013	pu	8763
lj	652	dh	1684	jl	4018	uu	9318
gz	653	dg	1694	dj	4042	lp	9375
fn	702	wu	1786	mu	4162	nb	9412
pj	703	pw	1798	io	4319	mp	9798
ln	724	gh	1846	kh	4519	pg	9952
gw	730	tp	1870	az	4621	ug	10065
tn	746	ub	1918	fo	4684	bu	10255
dk	770	ue	1937	nl	4720	iz	10346
bs	770	fa	1946	rf	4756	ce	10614
ju	787	sv	1985	mg	4793	ob	10832
mk	802	fh	1998	iv	4845	ut	11026
fb	805	pd	2040	if	4901	js	11164
xp	810	uf	2098	wt	4920	sn	11603
cr	827	cc	2127	nw	4952	rn	11718
lh	829	nr	2173	pv	5027	ai	11747
jm	835	ml	2202	i␣	5152	ea	11807
mz	863	gu	2275	ps	5379	uc	11850
mr	887	pb	2404	mv	5483	tm	11970
gb	890	ia	2494	si	5650	jo	12107
rj	908	ex	2496	kw	5704	oc	12327
nf	932	ks	2505	lm	5745	im	12338
ty	1005	ir	2521	tl	5751	fg	13234
yp	1061	uz	2612	ci	5825	tv	13518
rc	1062	kj	2648	nh	5834	nv	13985
up	1113	ws	2714	w␣	5954	wd	14497
sg	1143	ms	2750	lf	5961	dw	14691
xt	1174	fw	2800	gs	6317	mm	14773
pm	1177	ph	2850	fi	6458	po	14781
gv	1218	ji	2897	sa	6739	dd	14955
co	1240	wr	2916	oi	6783	jt	14991
mh	1287	nc	2983	eo	6894	ff	16724
fz	1289	gk	2984	fl	7011	nz	16788
dv	1309	lv	3023	sd	7151	jp	17121

td	17153	jg	38839	id	78194	rt	146260
pi	17247	eh	39002	jk	78576	pe	153810
␣f	17393	jv	40531	tt	79167	ei	154175
fd	18150	kr	42456	sp	80381	ku	155522
jz	18672	ti	42733	br	81535	ko	156869
tg	18744	pl	42837	rg	81590	sc	158593
go	18830	rv	42954	ru	82414	ll	160643
pp	19286	rz	44406	dr	82541	ta	165497
ba	19455	vi	44541	gt	82930	ed	168894
vl	19521	lu	44730	pa	83052	ht	170968
ls	20467	ew	45609	m␣	84633	un	173598
␣j	21213	ns	46352	ac	85423	il	174492
os	22092	af	46626	ud	87895	ri	179437
rp	22404	eu	46747	nk	89096	ek	186873
o␣	22686	pt	47340	hi	90680	it	189471
of	23301	je	47391	gg	92265	za	190110
gi	23551	rb	49889	bb	92642	do	193499
rh	23778	uw	51852	gr	93651	as	193647
jf	23811	mt	52659	j␣	94526	op	194568
tj	23833	bo	53023	ep	94674	ni	196627
zu	25260	lt	53307	p␣	96253	om	196748
fe	25378	bi	53594	ts	97296	␣t	197418
a␣	25534	og	53923	to	99423	ma	202135
rr	25803	gd	53945	ot	100777	nt	202157
uk	26339	no	54875	␣p	100945	kt	206235
ec	26856	di	55466	␣r	108779	wi	211799
zw	27940	ol	57875	ig	109099	ui	212915
nu	28018	mi	57951	rk	111745	li	215274
lg	28170	rw	58075	rs	113235	vo	217747
us	28708	ic	58240	␣u	113761	ro	221719
ad	28865	ft	59654	lo	114923	nn	222200
tu	29171	is	60547	ez	117271	eb	222245
am	29478	ur	60743	ak	117277	ev	225032
tw	30053	kl	61818	g␣	117852	␣i	231179
du	30058	b␣	64553	ik	120578	es	231707
ap	30238	rm	64712	k␣	120649	da	242715
hu	32465	na	65869	ou	121531	␣a	243146
u␣	33006	vr	66210	ag	124925	jn	243462
md	33123	bl	66294	ha	128684	wo	249832
od	34132	se	67013	ld	128790	s␣	252479
hr	34605	ov	67397	dt	129284	on	255114
jd	34818	ef	70878	ga	130255	va	259430
ss	35398	pr	71759	tr	131388	ze	262464
ul	36476	zo	72199	em	135978	eg	273302
ok	37237	sl	73393	ng	137098	la	276530
f␣	37329	kk	73794	ho	141243	ar	276942
ki	37897	rl	78114	ra	141638	oo	296911

␣n	298878	ke	389966	r␣	561114	te	884148
wa	302778	zi	397416	ij	634256	␣w	901999
d␣	313169	rd	400320	␣g	659967	e␣	912982
ka	313647	␣o	408205	oe	661537	an	933133
␣e	332988	␣l	412861	␣m	681102	ge	1018634
␣s	335696	ie	418510	␣h	696921	␣v	1056939
me	338973	el	437840	be	712857	de	1145252
ch	340144	l␣	450452	␣k	719021	et	1191966
mo	349919	at	497785	␣z	740326	er	1363465
we	350515	re	498307	␣d	742524	t␣	2263978
in	363489	st	507785	␣b	748574	en	3658432
nd	375783	le	510118	ee	763032	n␣	4311805
al	381085	or	512939	aa	814110		
ne	387852	he	530259	ve	877584		

These are the letters and their frequencies from the Dutch CELEX Lexicon (Baayen et al., 1993), ordered from least to most frequent.

q	665
y	2504
x	3214
f	217944
c	363809
p	593161
j	694051
u	766259
z	975546
h	1130651
w	1147974
b	1155330
m	1169060
s	1234999
v	1493013
k	1582644
g	1658976
l	2036313
d	2241317
i	2261926
o	2697293
r	2966370
a	3616656
t	3948701
n	6184858
e	10389736

Appendix B

Annotated source code

This chapter describes the source code of the program used to calculate the optimal combination of keys. The goal of this program in R (R Development Core Team, 2004) is to obtain a combination of eight keys with three letters and one key with two letters, with the lowest total penalty.

```
## The R library "combinat" (Chasalow, 2002) is loaded, this is a library
## for combinatorics utilities.

library(combinat)

which.Min <- function(v,num=1) {
  subVec <- 1:min(length(v),num)
  order(v)[subVec]
}

## All the combinations of the twenty-six letters and the additional
## capital X are calculated.

keys <- t(combn(c(letters,"X"),3))

## the list of bigrams and their penalties.

bigrams <- read.table("nederlands.txt",as.is=T,sep=",",
                      quote="",header=F)

## Each column is given an appropriate name

bigr.corpus <- bigrams
names(bigr.corpus) <- c("bigram","frq")

## A table with the letters of each bigram in its columns:

biletters <- strsplit(bigr.corpus$bigram,"")
```

```
totPenalty <- 0

## For each entry in the list of bigrams, all the keys are assessed.
## If both the letters from the bigram are present in the key, then
## the bigram penalty (a column in the bigrams table) is added
## to the total penalty. Double letters ("aa"), are not taken into
##consideration for this calculation to prevent "pollution"
## of the values.
## All the keys and their penalties are then saved into a file.

if (0) {
  penalty.key <- rep(0,nrow(keys))
  for (i in 1:length(biletters)) {
    cat("Now working on: ",i,"\n\n")
    for (j in 1:nrow(keys)){
      if ((sum(biletters[[i]] %in% keys[j,]) == 2) &
(biletters[[i]][1] != biletters[[i]][2])) {
        penalty.key[j] <- penalty.key[j] + bigr.corpus$frq[i];
      }
    }
    save(penalty.key,file="penalty.Rdat");
  }
 } else {
   load("penalty.Rdat");
 }

## The keys and their penalties are now put into a table with four
## columns: the three characters on the key, and the key penalty.

keys <- data.frame(keys,penalty.key)

names(keys) <- c("t1","t2","t3","penalty")
keys$t1 <- as.character(key$t1)
keys$t2 <- as.character(keys$t2)
keys$t3 <- as.character(keys$t3)

## The function show.key prints the four columns to the screen.

show.key <- function(X) {
  cat(" - ",X$t1,X$t2,X$t3,sep="")
}

## This is a value that we have determined to be higher than the
## optimal score by previous calculations, but that is low enough
## to speed up the process by cutting off enough branches
## of the search tree.
```

```
bestSoFar <- 200000;

## The list of letters is extended with the capital X,
## denoting a free space.

lettersX <- c(letters,"X")

## The total number of calculations

iterations <- 0

## The keys are arranged in increasing order of penalty.

keys <- keys[order(keys[,4]),]

## Only keys with a penalty less than bestSoFar are considered
## in the calculations. Keys with higher penalties will not lead
## to an optimal layout.

keys <- keys[keys[,4]<bestSoFar,]

## The main (recursive) function:

get.best.combo <- function(okKeys=0,curPen=0,keyNum=0) {

## If no key has been assessed yet, set the number of calculations
## to zero and make the list of OKKeys equal to the list of keys.
## Then proceed to the first key in the selection.

  if (keyNum == 0) {
    cat("\n");
    iterations <<- 0
    selection <<- rep(NA,9)
    okKeys <- 1:nrow(keys)
    keyNum <- 1;
  }

## For each calculation, add one to "iterations".
## Print the current selecion plus the best score so far each time
## iterations reaches a multiple of one thousand.
## If the selection has not yet found a complete selection
## of nine keys, leave the blank keys open.

  iterations <<- iterations + 1
  if ((iterations %% 1000) == 0) {
    cat("\n")
```

```
    cat(iterations," ");
    cat("=======================\n")
    for (i in 1:9) {
      if (i > keyNum) {
        cat(" - ...")
      } else {
        show.key(keys[selection[i],]);
      }
    }
    cat(" ",bestSoFar,"\n");
  }

## If the number of the key in the selection is nine, then check
## if there is only one key left in the list of OK Keys.
## If not, print an error message.
## If the last key in the selection has a penalty that is low enough,
## add the key to the selection and print the total selection
## with its total penalty.

  if (keyNum == 9) {
    if (length(okKeys) != 1) {
      cat("ERROR, ERROR, ERROR!\n");
    }
    selection[keyNum] <<- okKeys;
    curPen <- curPen + keys[okKeys,]$penalty;
    if (curPen < bestSoFar) {
      bestSoFar <<- curPen;
      cat("\n");
      for (i in 1:9) {
        show.key(keys[selection[i],]);
      }
      cat(" ",bestSoFar,"\n");
    }
  } else {

## The current penalty of the selection is increased every time
## a key is added to the selection. By keeping track of the
## "former current penalty", we make sure that the penalty of the key
## added to the selection will not have to be subtracted
## from the penalty later on.

    curPen.org <- curPen;

## Key number n has to start with the key with the lowest
## alphabetical ranking, with letters that are not part of the keys
## prior to key number n. Because not only key number n, but also
## key numbers one to n-1 are selected this way due to recursivity,
```

```
## we only have to look at key number n.

    if (keyNum>1) {
      curLowest <- keys[selection[keyNum-1],1]
      hasToBeginWith <- lettersX[which(lettersX==curLowest)+1]
      if (keys[selection[keyNum-1],2]==hasToBeginWith) {
        hasToBeginWith <- lettersX[which(lettersX==hasToBeginWith)+1]
      }
      if (keys[selection[keyNum-1],3]==hasToBeginWith) {
        hasToBeginWith <- lettersX[which(lettersX==hasToBeginWith)+1]
      }
    } else {

## The first key always has to start with "a".

      curLowest <- NA;
      hasToBeginWith <- "a";
    }

## A subset consisting of al the possible keys for the current key
## number in the selection.

    subsetForCurKey <- okKeys[keys[okKeys,]$t1==hasToBeginWith]

    for (i in subsetForCurKey) {

      curPen <- (curPen.org + keys[i,]$penalty)

      selection[keyNum] <<- i;

## The first, second and third letters of the list of OKkeys and of
## the list of total keys are put in separate lists, ...

      okKeys1 <- keys[okKeys,1]
      okKeys2 <- keys[okKeys,2]
      okKeys3 <- keys[okKeys,3]
      keys1 <- keys[i,1]
      keys2 <- keys[i,2]
      keys3 <- keys[i,3]

## ... and are used to ensure that the new list of OKKeys does not
## contain letters that have already been used in the combination.

      new.OKKeys <- okKeys[(okKeys1 != keys1) &
                           (okKeys2 != keys1) &
                           (okKeys3 != keys1) &
                           (okKeys1 != keys2) &
```

```
                                (okKeys2 != keys2) &
                                (okKeys3 != keys2) &
                                (okKeys1 != keys3) &
                                (okKeys2 != keys3) &
                                (okKeys3 != keys3)]

## The list of new OKKeys is once again checked for keys with
##  penalties that are too high.

      tmp <- keys[new.OKKeys,]
      new.OKKeys <- new.OKKeys[((curPen + tmp$penalty) <= bestSoFar)]

## If there are still keys that can be placed into the combination,
## call the main function recursively.

      if (length(new.OKKeys) > 0) {
        get.best.combo(new.OKKeys,curPen,keyNum+1)
      }
    }
  }
  selection[keyNum] <- NA;
}

get.best.combo()
```

The final best combination is:
auy - btz - cjr - dkp - eqX - fmn - gsv - hlw - iox 56837

Appendix C

Sentences used in the experiment

C.1 Full set of sentences for the model and the experiment

The following sentences were used with the model and in the experiment. The sentences that are marked with an asterisk were used only in the experiment. These sentences had to be typed on the alphabetically arranged keyboard to determine the speed with which letters were typed on this keyboard.

om deze zin in te toetsen moet je misschien veel zoeken*
de olifant liep in het bos te denken over zijn volgende plan*
eigenlijk zou ik in een boom moeten klimmen dacht hij*
na een tijd kwam hij de muis tegen. dag zei de woelmuis*

ik wil je iets vragen, weet jij soms een kleine boom?
jazeker zei de muis, kom maar mee! ik breng je er heen
dichtbij de rand van het bos was een open plek
daar bleef de woelmuis plotseling staan en wees
ik zie niets zei de olifant. hij keek nog eens goed
hij ging liggen op de plaats die de muis aanwees
nu zag hij hem. hij had nog nooit zo een kleine boom gezien
a b c d e f g h i j k l m n o p q r s t u v w x y z
de olifant probeerde zijn voet ergens op te zetten
hij probeerde zijn slurf ergens omheen te slaan
maar de boom was zo klein dat dat niet lukte
die boom is inderdaad heel klein, muis!
de olifant was lange tijd in de weer
a b c d e f g h i j k l m n o p q r s t u v w x y z
hij had zijn vier voeten bij elkaar gezet
nu moet ik alleen nog mijn evenwicht bewaren dacht hij
het duurde heel even maar toen begon de olifant te wankelen
de olifant sloeg achterover. het was een harde klap
a b c d e f g h i j k l m n o p q r s t u v w x y z
toen hij zijn ogen weer open deed stond de muis voor hem
klein he zei de muis. dat gaf de olifant direct toe
hij liep verder en kwam een wilg tegen

hij zette zijn voet op de onderste tak van de wilg
maar de wind stak op en de tak zwiepte heen en weer
de olifant werd omver gesmeten
hola riep de olifant, ik was nog niet eens zover!
hij stond op en sloeg zijn slurf om de stam
a b c d e f g h i j k l m n o p q r s t u v w x y z
het was een woeste strijd, de olifant en de wilg hielden vol
uiteindelijk gooide de wilg de olifant in het water
nu moest hij op zoek naar een ander plan
a b c d e f g h i j k l m n o p q r s t u v w x y z

C.2 Set of sentences used for the "e"

The following set of sentences was used to test whether the location of the letter "e" was learned after only five occurrences with the original set of sentences because of the short distance between consecutive occurrences.

om na te gaan of dit de oplossing is zijn de klinkers uit elkaar gehaald
daarna typen we de letter weer vaker in
nu kunnen we nagaan of dit een verschil heeft gemaakt

Appendix D

Post-test questionnaire and participants' answers

D.1 Questionnaire

The following post-test questionnaire was given to the participants.

1. Wat is je leeftijd?

..... jaar

2. Wat is je geslacht?

Man / Vrouw

3. Studeer je aan de Rijksuniversiteit Groningen?
Ja / Nee

Geef bij de volgende stellingen een cijfer op de schaal van 1 tot 5. 1 = helemaal niet mee eens, 2 = niet mee eens, 3 = neutraal, 4 = mee eens, 5 = helemaal mee eens

4. Ik vond het experiment leuk om te doen

1 2 3 4 5

5. Ik vond het experiment moeilijk om te doen

1 2 3 4 5

6. Ik leerde snel waar de letters op het toetsenbord stonden

1 2 3 4 5

7. Ik was voor aanvang van het experiment erg moe.

1 2 3 4 5

8. Ik was na afloop van het experiment erg moe.

1 2 3 4 5

De volgende vragen gaan over de locaties van de verschillende letters en hoe snel je deze locaties leerde.

9. Vond je de indeling van de letters logisch?

Ja / Nee

10. Was de onbekende indeling van de letters vervelend voor de taak?

Ja / Nee

11. Van welke letters wist je al snel waar ze stonden?

12. Van welke letters duurde dat langer?

De volgende vragen gaan over de methodes die je gebruikt hebt bij het zoeken naar de letters op het toetsenbord.

13. Hoe heb je gezocht naar letters in het begin? (mogelijke antwoorden kunnen zijn: van linksboven naar rechtsonder, rond kijken, zigzag, zonder duidelijk systeem). Omschrijf je manier van zoeken zo goed mogelijk, ook als het een combinatie van bovenstaande antwoorden is of juist iets heel anders.

14. Veranderde deze manier van zoeken in de loop van het experiment? Zo ja, hoe dan?

15. Zijn je nog andere dingen opgevallen in de manier van zoeken die je op dit toetsenbord gebruikte?

De volgende vraag gaat over de manier waarop je uiteindelijk de letters op het toetsenbord vond.

16. Hoe heb je aan het eind van het experiment de letters op het toetsenbord gevonden? (mogelijke antwoorden kunnen zijn: ik zocht ze over het hele toetsenbord, ik wist de locaties van de letters, ik wist de regio op het toetsenbord waar de letters ongeveer te vinden waren, mijn hand bewoog automatisch naar de juiste plek). Omschrijf de manier waarop je de letters aan het eind van het experiment vond zo goed mogelijk, ook als het een combinatie van bovenstaande antwoorden is of juist iets heel anders.

De volgende vragen gaan over de manier waarop de letters nu op het toetsenbord zijn geplaatst.

17. Wat vond je van de manier van werken op dit nieuwe toetsenbord? Was het snel duidelijk of juist verwarrend?

18. Werkte je op het eind sneller of juist langzamer op het nieuw ingedeelde toetsenbord dan op een alfabetisch ingedeeld toetsenbord? Verklaar zo mogelijk je antwoord.

19. Moest je meer of juist minder met je hand bewegen op het nieuw ingerichte toetsenbord dan op een alfabetisch ingedeeld toetsenbord? Verklaar zo mogelijk je antwoord.

Ruimte voor overige opmerkingen:

D.2 Answers Participant 3 (Male, 22)

4. 4
5. 2
6. 4
7. 3
8. 4
9. Ja
10. Nee
11. D, e, n, r, i, o, q
12. U, s, p, f, v
13. Rond kijken, systeem zoeken in de volgorde
14. Systeem min of meer ontdekt, eerste letters zijn meest gebruikte, dus gebruikte ik dat om sneller te zoeken.
15. Zoeken naar backspace en enter soms lastig
16. Van veel letters wist ik de locatie, van anderen min of meer (zoals t, a, w), de weinig gebruikte letters vergden meer zoeken.
17. Aanvankelijk verwarrend, maar wel snel duidelijk
18. Wel langzamer, op alfabetisch toetsenbord gaat het al automatisch dus dat kan bijna zonder na te denken
19. Ik denk minder, vooral bij woorden zoals "de" en "een"

D.3 Answers participant 4 (Female, 22)

4. 4
5. 2
6. 4
7. 3 / 4
8. 4
9. Ja
10. Nee

11. D, e, e
12. S, t, p, k
13. Zonder duidelijk systeem, zoeken naar "vorm" van de letter
14. Ging sneller, je ging alleen nog de "onbekende" letters zoeken
15. Vaak was het zo dat veelgebruikte letters op een voorname (vaak 1e) plaats stonden. Bijv x, y, z op de derde.
16. Ik wist de locatie van de letters. Sommige letters die je niet veel gebruikt waren lastiger.
17. (snel) duidelijk, aan het eind vooral. In het begin wel verwarrend omdat je met een ander systeem was begonnen.
18. Op het eind ging het vinden van letters snel, maar door vermoeidheid maakte ik denk ik sneller fouten, daardoor vertraging.
19.Minder, letters stonden heel logisch ingedeeld. Gemakkelijk woorden mee te formuleren.

D.4 Answers Participant 5 (Female, 21)

4. 4
5. 1
6. 3
7. 2
8. 3
9. Nee
10. Nee
11. E, n, d, r, o
12. H, p, b
13. Zonder systeem
14. Nee
15. Meest voorkomende letters staan op de eerste plaats
16. Mijn hand bewoog naar de juiste plek
17. In het begin verwarrend, later wen je er aan
18. Ik denk even snel. Als je vaak hetzelfde woord intypt, leer je automatisch de letters op het toetsenbord.
19.Woorden als "de", "en", "er" gaan sneller. Een woord als "olifant" gaat vervelender.

D.5 Answers participant 6 (Female, 20)

4. 3
5. 2
6. 2
7. 3
8. 3
9. Nee
10. Nee
11. D, e, n, i, j, l

12. S, U, p
13. Van linksboven naar rechtsonder
14. Nee
15. Ja, ik heb de backspace toets gedurende het hele experiment vaak aangezien voor de spatiebalk, met het gevolg dat ik mijn eigen woorden vaak wiste. De "s" vind ik een onlogische plek hebben, daar heb ik vaak naar gezocht, net als de "u", omdat deze letter naast de "a", een "au" vormde.
16. Ik wist de locatie van en groot aantal letters d.m.v. een aantal woorden bijv. bij "olifant" sneller te vinden
17. Verwarrend. Naar het einde van het experiment toe ging dat beter.
18. Wel sneller, maar ik heb niet het idee gehad dat ik het alfabet m.b.v. het nieuwe systeem steeds sneller in kon tikken.
19.Meer, omdat ik langer aan het zoeken was naar bepaalde letters, "screende" ik het scherm met de muis.

D.6 Answers Participant 7 (Female, 19)

4. 5
5. 2
6. 4
7. 2
8. 2
9. Ja
10. Nee
11. E, i, j, n, m
12. P, c, f
13. Ik heb gezocht van linksboven naar rechtsonder
14. Later heb ik zigzag gezocht, omdat ik de meeste letters wel kon vinden
15. Naarmate ik meer letters wist te vinden zocht ik minder geordend naar de letters. Dus zonder duidelijk systeem.
16. Ik wist de letters ongeveer te vinden, af en toe wist ik niet meer waar een letter zat. Maar over het algemeen wist ik de letters te vinden.
17. Het werd snel duidelijk tijdens het oefenen.
18. Aan het eind werkte ik sneller op het nieuw ingedeelde toetsenbord. Door de oefeningen was ik er snel aan gewend.
19.Minder, door de volgorde van de letters kon je makkelijker woorden vormen zonder je hand veel te gebruiken.

D.7 Answers participant 8 (Male, 22)

4. 5
5. 2
6. 4

7. 2
8. 3
9. Ja
10. Nee
11. E, n, i, d, k, w, m
12. S, u, h, v
13. Zonder duidelijk systeem, gewoon beginnen met de eerste zin en al doende handigheid krijgen. Wel heb ik de locatie van de "e" en de "n" bekeken.
14. Eigenlijk niet. Na verloop van tijd weet je welke letter onder welk nummer staat en ga je op die manier te werk.
15. De letters onder de cijfers 4, 5 en 6 hebben de meeste aandacht in je blikveld en vind je het eenvoudigst.
16. Ik wist de locatie van bijna alle letters. Als ik de locatie vergeten was, wist ik de regio.
17. Het was veel logischer. Je hoefde bijna nooit hetzelfde cijfers tweemaal achter elkaar te gebruiken. Uiteraard was gewenning nodig
18. Ik denk dat ik sneller was op het nieuwe toetsenbord, omdat met name lidwoorden en voorvoegsels veel makkelijker te typen zijn.
19.Geen idee. Naar mijn gevoel iets minder voor de meest voorkomende lettercombinaties. De locatie van de "s" en de "v" waren echter onprettig.

Appendix E

Tables

E.1 Minimum Search Times per letter

This appendix contains six tables , Table E.1 to Table E.6, with the shortest search times per letter, the moment in the experiment when this minimum search time was displayed and the mean and standard deviation of the search times for that particular letter. Search time for a letter on a button is defined as the time between the last press on another button than the current button and the first time the current button is pressed. Each table gives the data for one participant. Minimum search time (ST) is in milliseconds, elapsed time (ET) in minutes.

Table E.1: Search times for participant 3

letter	Min ST	ET	mean ST	sd ST
a	250	33.2	1612.8	805.9
b	450	36.5	1251.8	1546.5
c	390	20.4	1578.1	1093.0
d	341	29.2	1098.9	655.5
e	**200**	**24.8**	**887.8**	**548.0**
f	390	36.3	1085.4	785.2
g	611	26.4	1543.4	808.7
h	330	40.3	1895.3	1331.2
i	381	30.5	854.2	326.0
j	240	21.9	625.8	532.8
k	400	40.4	1307.7	759.8
l	370	40.4	1944.2	996.6
m	341	31.1	801.4	417.4
n	210	33.1	678.4	401.1
o	420	34.5	1210.8	578.7
p	651	35.5	1973.7	900.0
q	350	23.7	682.4	514.9
r	310	34.3	822.8	390.5
s	381	40.5	1838.9	850.9
t	371	18.3	2235.3	1162.1
u	961	22.3	2020.7	809.6
v	461	36.8	2282.2	971.0
w	360	36.8	650.3	251.8
x	350	26.9	835.3	445.8
y	581	36.9	1117.6	302.0
z	431	31.6	3048.6	1344.7

Table E.2: Search times for participant 4

letter	Min ST	ET	mean ST	sd ST
a	411	7.0	1800.8	1071.1
b	781	27.7	1241.2	378.7
c	551	30.4	2806.8	2362.4
d	461	39.3	1773.6	108.4
e	**321**	**40.7**	**1146.9**	**650.0**
f	471	30.5	1436.2	803.8
g	501	38.1	1933.7	1276.2
h	430	48.3	2439.5	1329.7
i	420	49.8	1401.4	688.4
j	460	43.9	1373.5	1087.4
k	521	30.1	1694.2	927.4
l	541	13.2	1767.6	857.9
m	500	30.9	874.2	306.6
n	350	46.6	775.1	435.2
o	370	50.0	1358.3	842.5
p	631	50.0	1974.6	1123.3
q	451	50.0	1928.8	1308.9
r	501	37.6	1280.4	814.6
s	530	46.0	2464.1	1189.2
t	541	19.3	2127.9	951.0
u	511	44.0	2039.7	1279.4
v	801	46.9	2763.5	1488.9
w	330	47.5	906.8	341.8
x	621	45.4	1398.0	857.8
y	701	45.4	997.3	324.2
z	621	40.3	1917.2	883.3

Table E.3: Search times for participant 5

letter	Min ST	ET	mean ST	sd ST
a	571	39.3	1883.1	1005.5
b	551	17.1	1039.8	489.0
c	511	41.8	1953.8	1628.1
d	361	32.2	1261.0	1006.1
e	**200**	**33.3**	**972.7**	**582.1**
f	501	41.9	1351.0	914.6
g	391	35.5	1453.1	1127.8
h	551	41.9	3013.6	2426.6
i	420	36.0	1174.4	881.0
j	451	20.3	1180.2	1032.1
k	641	3.3	1457.2	703.7
l	480	42.0	2186.8	1155.9
m	421	30.9	1051.5	733.0
n	270	33.8	820.4	629.0
o	391	37.9	1489.2	833.6
p	350	41.2	2339.1	2143.1
q	431	12.8	734.2	302.6
r	360	20.5	1155.9	516.5
s	431	20.5	1840.8	1152.8
t	401	17.6	2322.6	1309.1
u	521	20.6	2430.9	2148.0
v	621	24.0	2017.3	1118.2
w	381	27.0	813.0	380.8
x	450	40.0	864.6	419.3
y	431	27.1	1177.6	729.9
z	951	28.1	2854.5	2353.1

Table E.4: Search times for participant 6

letter	Min ST	ET	mean ST	sd ST
a	641	37.5	1793.3	953.7
b	601	12.1	1466.1	1139.0
c	731	30.1	1790.5	933.5
d	450	37.6	1625.2	954.4
e	**330**	**36.3**	**1082.7**	**620.5**
f	480	45.2	1276.0	881.4
g	521	40.8	1631.8	966.6
h	540	45.2	1818.1	1133.2
i	411	7.2	1094.5	559.3
j	391	45.3	1233.2	982.1
k	490	45.3	1549.8	1412.9
l	541	41.9	1721.5	961.7
m	411	10.8	850.8	488.5
n	381	34.5	948.5	616.5
o	581	38.7	1559.2	571.9
p	701	44.8	2084.9	1069.9
q	441	41.1	1100.7	780.9
r	510	39.3	13268	763.5
s	451	23.5	1774.3	963.8
t	390	38.9	1705.6	905.2
u	451	26.7	1928.9	1871.3
v	751	42.9	2347.4	1401.0
w	380	31.3	791.6	609.5
x	510	13.6	1260.2	772.7
y	731	30.9	1862.7	1523.5
z	691	32.9	1899.6	1208.2

Table E.5: Search times for participant 7

letter	Min ST	ET	mean ST	sd ST
a	581	11.6	2145.9	1270.8
b	902	13.6	2987.1	2121.7
c	851	32.8	2507.9	1443.9
d	500	225.5	1947.0	1295.3
e	411	59.1	1216.8	674.1
f	600	71.2	2546.2	2029.5
g	681	34.7	2567.8	1121.0
h	651	34.4	2899.2	1823.0
i	411	27.1	1224.4	666.8
j	410	65.0	1067.5	765.4
k	631	34.4	2814.5	2018.0
l	480	66.2	2152.8	1141.2
m	471	26.3	1292.2	1230.6
n	**391**	**53.2**	**935.0**	**672.2**
o	491	66.2	1903.6	916.9
p	671	65.2	2328.5	1489.8
q	421	71.6	1127.3	673.9
r	621	26.4	2038.1	1149.2
s	491	65.3	1751.8	1086.5
t	671	15.8	2681.4	1173.1
u	611	34.9	2161.2	1307.4
v	1171	34.7	3251.1	1192.0
w	490	59.2	1298.7	1364.9
x	621	71.9	1154.3	406.2
y	561	65.6	807.4	338.5
z	751	57.7	2715.8	931.7

Table E.6: Search times for participant 8

letter	Min ST	ET	mean ST	sd ST
a	591	19.0	1749.0	1138.6
b	531	46.3	2086.2	1028.6
c	401	28.2	1544.4	1304.0
d	301	44.0	1491.2	1174.6
e	280	33.2	1074.0	642.6
f	481	41.4	1660.7	1091.2
g	360	40.5	1512.1	983.1
h	481	41.5	3165.9	2318.6
i	400	9.8	1541.0	782.5
j	331	45.5	1309.2	1668.5
k	401	39.0	1958.8	1420.6
l	631	20.1	2086.0	1119.6
m	541	46.7	1506.1	1268.2
n	**200**	**4.7**	**1038.9**	**750.4**
o	550	42.9	1885.1	852.3
p	300	40.3	1546.4	1031.7
q	410	46.8	1864.6	944.6
r	481	35.9	1457.2	962.9
s	260	40.7	2063.7	1324.1
t	360	19.0	2102.6	1503.9
u	721	23.8	2545.6	1623.0
v	811	28.8	2557.0	1377.5
w	411	47.0	1372.0	540.3
x	1101	42.0	2175.5	1051.3
y	501	13.5	560.8	102.7
z	1272	22.8	2840.8	1498.4

E.2 Learning Times per Letter

Table E.7 contains information about the number of occurrences (n) of the ten most common letters in the text prior to the moment the location of the letter was learned and the elapsed time (ET, in seconds) during the experiment. A letter is considered to be learned if the search time to find that letter is within a range of 300 milliseconds from the minimum search time for that letter. The button prior to the current button was checked and the minimum movement time between the two buttons was added to the minimum search time for that letter. The maximum difference, 300 milliseconds, is the time it takes to move the finger from the upper left corner to the lower right corner of the keyboard. P3 to P8 are the codes for the six participants. For letters that show a floor effect in search times, the learning times were hard to determine because the search times were already very low at the beginning of the task.

Table E.7: Learning times per letter

letter	P3		P4		P5		P6		P7		P8	
	n	ET	n	ET	n	ET	n	ET	n	ET	n	ET
a	10	121	10	154	60	2096	55	1991	13	366	26	890
d	6	265	5	313	8	433	28	1781	12	672	12	1155
e	16	155	7	74	14	158	9	80	25	416	25	316
g	10	1540	10	1296	10	1153	10	2931	10	2006	7	571
i	14	660	26	2978	8	218	12	415	9	301	18	576
l	11	1233	11	773	11	1573	17	1732	11	1557	18	1193
n	9	212	26	1308	14	368	9	273	12	430	12	348
o	13	518	11	487	18	740	22	940	26	1203	27	1048
r	9	921	25	2181	4	162	4	163	13	1158	14	1057
t	16	1074	16	1130	16	1078	16	1165	9	918	16	1105

E.3 Absolute speed-up per letter on the new keyboard

Table E.8 shows the relative speed-up per letter on the new keyboard. The absolute speed-up per letter on the new keyboard is calculated by subtracting the average search time in the last sentences typed on the new keyboard from the search times for these letters in the first sentences. For the letters that only occur in the alphabetical sequences, the last search time is subtracted from the first. The mean search time and standard deviation for each letter are listed as well.

Table E.8: Absolute speed-up per letter

letter	P3	P4	P5	P6	P7	P8	$\bar{X}$	sd
a	-32.4	641.7	1178.9	518.8	856.5	216	564.8	408.1
b	1609.2	141.6	202.9	1212.4	2622.2	560.3	1058.1	958.4
c	144.5	-1066.0	1301.8	2174.8	-358.4	1212.3	568.1	1203.5
d	1167.2	11473.5	1024.3	1533.2	1099.0	1861.1	1359.7	320.0
e	559.1	387.2	194.7	593.1	408.1	388.5	421.8	143.0
f	2070.7	560.0	577.7	370.3	914.3	2674.0	1071.1	1111.3
g	1061.3	2292.9	1134.3	715.2	196.8	1135.2	1089.3	691.4
h	198.9	81.1	1866.0	-362.4	50.2	-1699.1	22.4	1144.2
i	230.2	518.3	522.2	609.3	403.6	175.9	410.0	173.9
j	1128.6	1595.0	1379.8	2023.3	1348.4	2665.3	1690.1	565.6
k	169.8	314.5	513.2	-110.0	2563.7	1467.6	819.8	1009.7
l	1266.9	905.7	1087.8	1383.7	-200.8	513.9	826.2	588.6
m	270.3	91.9	646.9	121.5	922.0	405.8	409.7	323.2
n	58.5	115.7	-169.6	82.4	81.4	-83.3	14.2	113.8
o	466.8	1164.0	826.5	878.4	597.1	676.2	768.2	245.3
p	945.3	2146.3	1061.1	841.3	2189.7	828.0	1335.3	650.6
q	120.0	2664.0	-220.0	-359.0	421.0	1443.0	678.2	1165.2
r	293.8	-211.8	423.4	67.9	1571.5	1338.6	580.6	714.7
s	1098.2	1675.6	1178.1	372.3	1781.9	1046.8	1192.1	506.7
t	1321.6	632.8	1356.1	774.9	-516.1	238.5	634.6	706.2
u	170.5	-467.3	3355.2	4025.7	-513.8	56.7	1104.5	2032.7
v	1366.5	2314.0	1792.2	1529.2	1081.5	626.5	1451.7	581.5
w	256.2	367.8	536.8	-219.8	-814.9	-320.4	-35.4	519.2
x	731.0	721.0	31.0	-71.0	1051.0	3055.0	919.7	1133.4
y	-1.0	60.0	641.0	2714.0	811.0	-10.0	702.5	1046.6
z	2289.0	340.5	-1803.0	811.5	627.7	-606.0	276.6	1383.3

E.4 Relative speed-up per letter on the new keyboard

Table E.9 lists the speed-up for each letter relative to the search time for that letter in the beginning of the task (in percentages).

Table E.9: Relative speed-up per letter

letter	P3	P4	P5	P6	P7	P8
a	-2.1	34.1	47.3	27.3	36.3	13.0
b	65.5	16.9	14.2	52.4	60.4	25.6
c	5.6	-45.9	35.0	55.8	-12.8	49.0
d	57.0	54.6	47.6	59.2	42.0	63.0
e	47.4	30.8	17.8	44.3	28.6	26.4
f	80.4	29.3	41.8	-67.2	24.2	62.4
g	45.5	67.0	41.5	34.2	6.1	51.4
h	9.6	3.3	40.8	-28.7	2.0	-68.2
i	22.9	28.5	32.7	38.7	26.1	10.9
j	73.4	64.9	57.8	73.3	58.4	81.9
k	14.9	16.2	31.3	-12.8	63.8	58.9
l	45.4	41.2	35.4	52.4	-10.9	18.4
m	30.0	8.7	54.5	16.1	54.6	36.0
n	8.1	16.6	-21.3	9.3	8.1	-8.3
o	30.0	52.0	37.9	41.2	25.4	28.1
p	48.9	52.8	44.0	41.7	61.7	40.2
q	22.6	85.5	-51.0	-48.4	50.0	77.9
r	26.3	-16.6	26.8	6.0	52.4	59.5
s	45.5	51.7	39.8	18.8	67.2	40.4
t	40.4	25.0	41.1	29.0	-23.0	10.9
u	7.0	-24.7	50.7	70.0	-21.6	1.6
v	50.3	74.3	50.3	67.1	226.5	17.2
w	31.7	36.2	43.2	-22.9	-87.0	-21.4
x	50.0	27.6	5.1	-13.9	62.9	84.7
y	-0.1	7.1	28.7	57.1	58.7	-2.0
z	52.6	16.5	-46.1	26.8	23.6	-26.4

Bibliography

Anderson, J. R. & Lebiere, C., The atomic components of thought. Mahwah, NJ: Erlbaum. (1998)

Baayen, R. H., Piepenbrock, R., & Rijn, H. van, The CELEX Lexical Database (Release 1) [CD-ROM]. *Philadelphia, PA: Linguistic Data Consortium*, University of Pennsylvania [Distributor]. (1993)

Chasalow, S., Combinat: combinatorics utilities. R package version 0.0-4. (2002)

Doering, N.,"Kurzm. wird gesendet" - Abkürzungen und Akronyme in der SMS Kommunikation. *Muttersprache. Vierteljahresschrift für Deutsche Sprache*, 112(2). (2002)

Fitts, P. M., The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology, 47*, 381-391 (1954). Reprinted in *Journal of Experimental Psychology: General, 121(3)*, 262-269. (1992)

Klahr, D., Chase, W. & Lovelace, E., Structure and process in alphabetic retrieval. *Journal of Experimental Psychology: Learning, Memory and Cognition, vol. 9, no. 3*, 462 - 477. (1983)

Lovelace, E. A. & Snodgrass, R. D., Decision times for alphabetic order of letter pairs. *Journal of Experimental Psychology, 88*, 258-264. (1971)

MacKenzie, I. S., & Buxton, W., Extending Fitts' law to two dimensional tasks. *Proceedings of the CHI '92 Conference on Human Factors in Computing Systems*. New York: ACM. (1992)

MacKenzie, I. S, & Soukoreff, R. W., Text Entry for Mobile Computing: Models and Methods, Theory and Practice. *Human Computer Interaction*. (2002)

National University of Singapore, Short Messages Service Corpus (2004). Retrieved May 2004 from http://www.comp.nus.edu.sg/~rpnlpir/downloads/corpora/smsCorpus/.

R Development Core Team, R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/. (2004)

Sears, A., Jacko, J. A., Chu, J. & Moro, F. , The role of visual search in the design of effective soft keyboards. *Behaviour and Information Technology, 3*, 159-166. (2001)

Sears, A. & Zha, Y, Data entry for mobile devices using soft keyboards: understanding the effects of keyboard size and user tasks. *International Journal of Human Computer Interaction, 16(2)*, 163-184. (2003)

Silfverberg, M., MacKenzie, I. S. & Kohonen, P., Predicting Text Entry Speed on Mobile Phones. *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI2000,* 9-16. New York: ACM. (2000)

Taatgen, N. & Lee, F. J., Production compilation: a simple mechanism to model complex skill acquisition. *Human Factors 45(1)*: 61-76. (2003)

West, L., The Standard and Dvorak Keyboards Revisited: Direct Measures of Speed. *Technical report, Santa Fe Institute.* (1998) See also http://www.santafe.edu/research/publications/workingpapers/98-05-041.pdf

Zhai, S., Kristensson, P., & Smith, B. A., In Search of Effective Text Input Interfaces for Off the Desktop Computing. *Interacting with Computers, Vol.16.* (2004)