Improved Text Entry for Mobile Devices: Alternate Keypad Designs
and Novel Predictive Disambiguation Methods

A DISSERTATION
SUBMITTED TO THE COLLEGE OF COMPUTER SCIENCE
OF NORTHEASTERN UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Jun Gong
October 2007
Acknowledgements

I would like to take this chance to truly thank all the people who have supported me throughout my years as a graduate student. Without their help, I do not think I could have achieved one of my life-long goals and dreams of earning a doctorate, and would not be where I am today.

First of all, I have to express my sincerest appreciation to my advisors Peter Tarasewich and Harriet Fell. I shall never forget these four years when I worked with such distinguished researchers and mentors. Thank you for always putting your trust and belief in me, and for your guidance and insights into my research.

I also have to thank my other thesis committee members: Professors Scott MacKenzie, Carole Hafner, and Javed Aslam. I would not be where I am now without the time and effort you spent helping me complete my dissertation work.

I am most indebted to my beloved parents, Pengchao Gong and Baolan Xia. I have been away from you for such a long time, and could not always be there when I should have. But in return, you have still given me the warmest care and greatest encouragement. I am forever grateful that you have given me this opportunity, and will continue to do my best to make you proud.

Among all my good friends, I first want to thank Lijun Xu. Your companionship and understanding greatly encouraged me to come this far. The completion of my Ph.D. would surely not be possible without your love.

I also thank my other colleagues and good friends, Yang Du, Wenxu Tong, Tian, Xia, Keshi Dai, Pangfeng Zhou, Yue Tang, Da Lin, Ye Wang, Jing Shan, Yunbing Zhang, Mingyan Shao, Bing
Zhang, and Biting Wu. You all made my years of study in Boston more enjoyable, and provided many joyful moments and happy memories.

Lastly, I thank all the researchers in the text entry field and all those who helped me with my usability studies. For without you, my research would not exist.
Table of Contents

Abstract .................................................................................................................................................. 1

1 Introduction ........................................................................................................................................ 2

1.1 Common Text Entry Methods Using Mobile Phone Keypads .................................................... 3

1.2 Research Areas .................................................................................................................................. 4

1.2.1 Keypad Designs and Their Alternatives ................................................................................. 4

1.2.2 Predictive Disambiguation ....................................................................................................... 5

1.2.3 Mobile Text Entry Error Analysis ............................................................................................ 6

1.3 Overview of the Dissertation ......................................................................................................... 6

1.3.1 Thesis Statement ..................................................................................................................... 6

1.3.2 Organization ............................................................................................................................ 7

2 Related Work .................................................................................................................................... 9

2.1 Measuring the Effectiveness of Text Entry Methods .................................................................... 9

2.1.1 Disambiguation Accuracy (DA) ............................................................................................. 9

2.1.2 Key Strokes Per Character (KSPC) ...................................................................................... 9

2.1.3 Limitations of Existing Metrics and Methods ....................................................................... 10

2.2 Predictive Disambiguation Text Entry Methods ......................................................................... 10

2.3 Keypad Design ................................................................................................................................ 13

2.4 Genetic Algorithm-based Heuristic ............................................................................................... 15

2.5 Text Entry with Small Keypad of Few Keys ............................................................................... 17

2.6 Word Prediction and Completion .................................................................................................. 18

2.7 N-Gram Language Model ............................................................................................................ 19

2.8 Semantic Relatedness Model ....................................................................................................... 20

2.9 Part-of-Speech Model .................................................................................................................. 21
2.9.1 Rule-based Tagging..................................................................................................... 21
2.9.2 Stochastic Tagging ...................................................................................................... 21
2.9.3 Applications of Part-of-Speech Tagging..................................................................... 24

2.10 Corpora....................................................................................................................... 24

3 Alphabetically Constrained Keypad Design...................................................26

3.1 Introduction..................................................................................................................... 26

3.2 Keypad Design Problem............................................................................................... 26

3.2.1 Keypad Design Problem for Predictive Disambiguation Text Entry Methods ....27
3.2.2 Keypad Design Problem for Multi-Tap Text Entry Method ................................. 27

3.3 Finding Solutions......................................................................................................... 28

3.3.1 GA-based Heuristic for Finding Unconstrained Keypad Solutions for Predictive
Disambiguation Text Entry Methods .............................................................................. 28
3.3.2 Finding Constrained Solutions for Predictive Disambiguation Text Entry Methods29
3.3.3 Finding Constrained Solutions for Multi-Tap Text Entry Method ............................30

3.4 Corpora....................................................................................................................... 31

3.5 Keypad Design Results............................................................................................... 33

3.5.1 Alphabetically Constrained Keypad Design Results for Predictive Disambiguation
Text Entry Method ....................................................................................................... 33
3.5.2 Unconstrained Keypad Design Results for Predictive Disambiguation Text Entry
Method ....................................................................................................................... 33
3.5.3 Alphabetically Constrained Keypad Design Results for Multi-Tap Text Entry
Method ....................................................................................................................... 33

3.6 Discussion of Keypad Design Results......................................................................... 38

3.7 Experiments to Empirically Test Keypad Designs....................................................... 45

3.7.1 First Experiment ...................................................................................................... 45
4 Semantic Relatedness and Part-of-Speech Language Models for Predictive Disambiguation Methods

4.1 Introduction

4.2 Predictive Disambiguation Problem for Small Keypads

4.3 Semantic Context-Aware Predictive Disambiguation Method

4.3.1 Statistical Semantic Relatedness of Word Pairs

4.4 Syntactic Context-Aware (Part-of-Speech) Predictive Disambiguation Method

4.4.1 Word Part-of-Speech (POS) Validity

4.5 Context-Aware Disambiguation Method Combining Semantic Relatedness, POS, and Frequency Models

4.6 Finding the Best Coefficients for Linearly Combining Semantic Relatedness, POS, and N-Gram Validities

4.6.1 The Equivalent Geometry Problem

4.6.2 The Solution to the Coefficient Optimization Problem

4.7 Reducing the Size of the Semantic Relatedness Models

4.8 Simulation

4.8.1 Corpus

4.8.2 Simulation Setup
Appendix A. The BNC Basic Tagset

Appendix B. The BNC Ambiguity Tagset

Appendix C. Groupings of Part-of-Speech Tags from the BNC Basic and Ambiguity Tagsets

Used by the improved Predictive Disambiguation POS Model

Bibliography
List of Figures

Figure 1.1: North American International Standard 12-key mobile phone keypad [2] ...................... 3
Figure 2.1: WordWise. The “1” key is used to input the grayed letter on other buttons [27]......... 12
Figure 2.2: Various Keypad Designs [10, 88, 8] ........................................................................ 14
Figure 2.3: 5-button watch top text entry system [12] .................................................................. 14
Figure 2.4: Pseudocode for Genetic Algorithm-based Heuristic .................................................... 16
Figure 2.5: Definitions for the Viterbi Algorithm for POS tagging [35] ........................................... 22
Figure 2.6: Viterbi Algorithm for POS tagging [35] ....................................................................... 23
Figure 3.1: Keypad Representation Used by the GA-based Heuristic ........................................... 28
Figure 3.2: Crossover Operation for Reproducing a Child Unconstrained Keypad Design .......... 29
Figure 3.3: Mutation Operation for Randomly Mutating a Child in the Unconstrained Keypad Design ................................................................................................................................................... 29
Figure 3.4: 26 English characters (A - Z) have 25 dividing positions (D1 – D25) ......................... 30
Figure 3.5: Picking 7 Dividing Positions Results in an 8-Key Constrained Keypad Design .......... 30
Figure 3.6: Maximum DA vs. number of keys for predictive disambiguation text entry method for each word list (constrained keypad designs) ........................................................................................................... 39
Figure 3.7: Total percentage of words of various lengths for each of the three word lists .......... 44
Figure 3.8: Constrained keypad design testing interface ................................................................... 46
Figure 3.9: Constrained uppercase letter keypad ............................................................................ 53
Figure 3.10: Unconstrained uppercase letter keypad ...................................................................... 53
Figure 3.11: Numeric keypad design ............................................................................................. 53
Figure 3.12: Punctuation keypad design, after one press of the “Done” key ................................. 53
Figure 3.13: Learning effects on text entry speeds and error rates ................................................. 55
Figure 4.1: Devices with very few keys or buttons .......................................................................... 61
Figure 4.2: Pseudocode for Context Aware Predictive Disambiguation Method ............................ 67
Figure 4.3: Flow Diagram of the Context-Based Disambiguation Method

Figure 4.4: Surface for Optimization of the Linear Combination Coefficients

Figure 4.5: Maximum disambiguation accuracies (DA) for various model combinations and different keypad sizes

Figure 4.6: Maximum Keystrokes Per Character (KSPC) for various model combinations and different keypad sizes

Figure 4.7: Maximum average DA values for frequency information and semantic relatedness models with various thresholds used on 3-key keypad design

Figure 4.8: Maximum average KSPC values for frequency information and semantic relatedness models with various thresholds used on 3-key keypad design

Figure 4.9: Size of the semantic relatedness model decreases as the cutoff threshold increases

Figure 4.10: Testing program using a 3-key constrained keypad design in editing mode

Figure 4.11: Testing program using a 3-key constrained keypad design in predictive disambiguation mode

Figure 4.12: Learning effects on text entry speeds and error rates for frequency-based and context-aware predictive disambiguation methods
List of Tables

Table 3.1: Maximum DA values for 1 to 12 key constrained keypad designs for the predictive
disambiguation text entry method by word list.................................................................34

Table 3.2: Selected optimal constrained keypad designs (4, 8, 9 and 12 keys) for the predictive
disambiguation text entry method for maximum DA values by word list.............................34

Table 3.3: Minimum KSPC values for 1 to 12 key constrained keypad designs for the predictive
disambiguation text entry method by word list.................................................................35

Table 3.4: Selected optimal constrained keypad designs (4, 8, 9 and 12 keys) for the predictive
disambiguation text entry method for minimum KSPC values by word list.............................35

Table 3.5: Maximum DA values (using the GA-based heuristic) for 1 to 12 key unconstrained
keypad designs for the predictive disambiguation text entry method by word list....................36

Table 3.6: Selected optimal unconstrained keypad designs (4, 8, 9 and 12 keys) for the predictive
disambiguation text entry method for maximum DA values by word list.............................36

Table 3.7: Minimum KSPC values for 1 to 12 key constrained keypad designs for the Multi-Tap
text entry method by word list.............................................................................................37

Table 3.8: Selected optimal constrained keypad designs (4, 8, 9 and 12 keys) for the Multi-Tap text
entry method for minimum KSPC values by word list..........................................................37

Table 3.9: Comparison of constrained keypad designs with other keypad designs (spoken wordlist)
..................................................................................................................................................40

Table 3.10: Comparison to optimal SMS keypad design...........................................................41

Table 3.11: Performance comparison of keypad designs optimized on one word list, but used with
another ........................................................................................................................................43

Table 3.12: Number of words of different lengths in different word lists.....................................44

Table 3.13: Sample set of sentences ............................................................................................47
Table 3.14: Comparison of entry speeds and error rates for constrained and unconstrained keypad designs .......................................................................................................................... 49

Table 3.15: Comparison of entry speeds and errors for constrained and standard cell phone keypad designs .......................................................................................................................... 49

Table 3.16: Comparison of entry speeds and errors across two testing sessions with the constrained keypad design ................................................................................................................. 49

Table 3.17: Example testing phrases ........................................................................................................... 54

Table 3.18: Comparison of speeds and error rates between keypad designs ............................................. 55

Table 4.1: Maximum predictive disambiguation performances for different predictive disambiguation methods and keypad sizes .......................................................................................................................... 75

Table 4.2: Predictive Disambiguation Performances When the Semantic Relatedness Model with Various Cutoff Thresholds Were Used for the Optimal Alphabetically Constrained Three-Key Keypad .......................................................................................................................... 78

Table 4.3: Example testing phrases ........................................................................................................... 86

Table 4.4: Comparison of text entry speeds and error rates between normal and improved predictive disambiguation methods ............................................................................................. 87

Table 5.1: Key sequences that have the same correction efficiency but a different cost per correction ........................................................................................................................................... 92

Table 5.2: CPC analysis on data from three previous experiments .......................................................... 95
Abstract

Despite the ever-increasing popularity of mobile devices, text entry on such devices is becoming more of a challenge. Problems primarily lie with shrinking device sizes, which can greatly limit available display space, as well as require unique input modalities and interaction techniques.

In attempting to resolve this issue, researchers have found that dictionary-based predictive disambiguation text entry methods are fairly efficient for text entry on devices such as mobile phones that use keypads instead of full keyboards. This type of text entry method “guesses” the word that a user desires by matching their sequence of keystrokes against feasible entries saved in a dictionary. However, word ambiguity, limited dictionary sizes, and large learning curves still prevent this method from being more widely adopted in many situations, and on more mobile devices.

Innovative solutions to these problems, focusing on both physical keypad designs and predictive disambiguation methods, are introduced in this dissertation work. The first part of this dissertation describes a set of keypad designs which are optimized under the constraint of keeping characters in alphabetical order across keys. Designs were found that have performance close to that of unconstrained designs, while maintaining better novice usability. The second part proposes a novel predictive disambiguation method which utilizes not only word frequency information, as do most existing dictionary-based predictive disambiguation methods, but also semantic and syntactical text information to help disambiguate the user’s desired words. Simulations and an empirical user study have shown improvements in text entry speed of up to 9.6% and reductions in the number of user errors of up to 21.2%. Furthermore, this dissertation presents a new error metric that is capable of revealing more information about user performance during experiments involving text entry methods.

In summary, this dissertation work focused on creating and validating improved methods for text entry on mobile devices.
Chapter 1

Introduction

Mobile devices, such as cell phones and personal digital assistants (PDAs), have become more popular during the last decade. While advancing technologies have enabled their use with many types of multimedia content, a lot of information is still being entered and processed on them in the form of text. Therefore, text entry remains a necessary part of human-computer interaction (HCI) with mobile devices.

Typical text-based mobile applications include "note taking", "appointment tracking", "address/phone number directories", and “short text messaging”. Although these applications usually do not require extensive amounts of text entry, a certain level of text input is inevitable. Furthermore, mobile text entry methods may be used under circumstances (e.g. where socially inappropriate or while moving [23]) that are often quite different from those where desktop computers are used. In addition, the more casual, unstructured, and hurried text entered into mobile devices can be very different from that used in more formal writing or in speech.

Different mobile devices use different interaction techniques for text entry purpose [78]. Some devices, such as Laptop, Blackberry, or Treo have regular or miniature full keyboards. Others, like PDA and Tablet PC, use touch screen and stylus as their primary interaction methods, therefore on-screen virtual keyboards are often used. Unlike any of these above devices, mobile phones, with much more limited surface area, usually use a 12-key keypad for text entry, as shown in Figure 1.1. Such a keypad design is ambiguous, because with multiple characters on a single key, a key press can be interpreted multiple ways. Therefore, popular mobile text entry methods, including MultiTap and dictionary-based predictive disambiguation method which will be introduced with more details in the next section, usually require more than one keystroke for each character input [55]. This dissertation describes some of the author’s
research in the mobile text entry field and several improvements to the dictionary-based predictive
disambiguation text entry method used with mobile phone keypads.

Figure 1.1: North American International Standard 12-key mobile phone keypad [62]

1.1 Common Text Entry Methods Using Mobile Phone Keypads

MultiTap is currently the most popular keypad-based mobile text entry method [55]. With it, users press a
text multiple times to specify the exact desired character on that key. For example, in Figure 1.1, users
would press the key "DEF" twice to input the character E, and the key "PQRS" four times for the
character S.

Dictionary-based predictive disambiguation initially requires one keystroke for each character of a word
entered. Stored linguistic knowledge (i.e., a dictionary of words) is then used to determine possible word
candidates for the given key sequence. For instance, the sequence “TUV”-“GHI”-“DEF” might produce
"the" out of all possible letter combinations. This process of predictive disambiguation must sometimes
automatically rank more than one word produced by the same set of keystrokes. For example, the
sequence “DEF”-“MNO”-“GHI” matches the words “dog” and “fog”. This ranking can be done through a
combination of word or N-Gram (sequence of n characters) frequency lists, user preferences, or past
history of user word choices. Special keystrokes must be available to cycle through a list of candidate
words when predictive disambiguation displays an undesired word as its top choice. If a desired word is
not in the dictionary, the user must somehow enter it into the system. Compared to MultiTap, dictionary-
based predictive disambiguation text entry requires fewer keystrokes, thus yielding higher theoretical text
entry speeds [73]. Examples of commercial dictionary-based predictive disambiguation text entry systems are T9 by Tegic [77], iTap by Motorola [51], and eZiText by Zi Corp. [85].

Models for text entry using mobile phone keypads [73, 38] have predicted expert input speeds as high as 27 words per minute (WPM) using MultiTap, and 46 WPM using dictionary-based predictive disambiguation methods. However, actual user studies have shown lower performances. An experiment by James and Reischel using a mobile phone [38] found that both experts and novices reached about 8 wpm with multi-tap. In comparison, the T9 method was used by novices at 9.1 WPM and by experts at 20.4 WPM.

There are also other types of mobile device text entry methods which do not use small keypads. Examples of such methods will be discussed in Chapter 2.

1.2 Research Areas

This section briefly describes the three research areas pertaining to text entry with mobile devices which are addressed by this dissertation work.

1.2.1 Keypad Designs and Their Alternatives

Previous research by Davis [13] has found that users of predictive disambiguation text entry methods on mobile phones will have to disambiguate about 5% of all the words in a common English dictionary when using the international standard keypad layout. The design of a keypad, in terms of both the number of keys, and the mapping of characters (and potentially numbers, punctuation marks, and editing functions) to keys, can significantly impact the performance of keypad-based text entry methods for mobile devices. One problem that has been attracting a lot of attention from HCI researchers is discovering and evaluating keypad designs that reduce the chances of encountering ambiguous keystroke sequences when using dictionary-based predictive disambiguation text entry methods. While the design in Figure 1.1 is widely used due to standardization issues, it has been shown that remapping characters to keys (e.g., such that “AFK” might appear on a single key) can theoretically improve text entry performance for a given
dictionary and method [e.g., 2, 48, 47, 50]. Research has been done using as few as three or four keys for text entry [e.g., 17, 54]. A significant concern with this problem, though, is creating designs that deviate as little as possible from current standard keypads to reduce learning curves and potential problems with novice usability. Furthermore, a sub-problem of telephone keypad design is creating designs with fewer than twelve keys that can work well with predictive disambiguation text entry methods. The aim here is creating viable text entry alternatives for ultra-small devices such as watches and bracelets, and methods that might potentially be used by people with limited hand and arm movement, in an effort to address concerns of universal usability of mobile devices and information systems.

1.2.2 Predictive Disambiguation

The second research area addressed in this dissertation work is improving the accuracy of dictionary-based predictive disambiguation methods themselves, irregardless of the keypad configuration used. Part of any predictive disambiguation method determines the order in which multiple candidate words corresponding to a given keystroke sequence are presented to users. Improved ordering can significantly reduce the number of keystrokes required for selecting the desired words; consequentially improving text entry performance.

Previous research has looked at various ways of improving the predictive disambiguation methods, including the use of word and character level digram information [32, 53], grammatical and linguistic information [65], and commonsense knowledge bases [76].

This dissertation research extends this work by investigating:

- How to use previously entered text and context information to help achieve better keystroke sequence disambiguation accuracy.
- The utilization of textual context information that resides far from the current word to aid the disambiguation process.
○ Potential performance improvements through utilization of a better predictive disambiguation method, especially for keypad designs with very few keys.

○ Improved language models that work well within the relative data storage and processing limitations of mobile devices.

1.2.3 Mobile Text Entry Error Analysis

Analysis of errors made by users with different text entry methods is another significant research concern for mobile HCI researchers. Current analysis techniques performed at the character level cannot handle some particular cases well, such as when errors are not noticed and corrected immediately. Furthermore, these techniques do not incorporate possible presses of function keys (e.g. "Next" key presses for predictive disambiguation methods, or arrow key presses for full keyboards) [74]. New metrics, which provide a more precise measure of how the user is actually entering text, have been developed and tested as part of this dissertation research to handle such cases.

1.3 Overview of the Dissertation

1.3.1 Thesis Statement

As already stated, ever-decreasing device sizes pose challenges to existing text entry methods. This dissertation documents our research work in the three areas described in Section 1.2, and presents the results of that research. This work has resulted in significant advances in the three areas as follows:

The optimal keypad designs (i.e., the mapping of characters to a given number of keys) that we found under the constraint that characters must stay in alphabetical order across keys perform favorably when compared to corresponding unconstrained versions of the designs in terms of reducing the average number of ambiguous keystroke sequences for a given word dictionary. More importantly, our constrained keypad designs have achieved significantly faster text entry speeds in our usability experiments with novice users than the unconstrained designs because of their greater similarity to standard mobile phone keypad designs (i.e., learnability increased).
This dissertation work has contributed to the second research area by creating improved predictive disambiguation methods using novel semantic relatedness and part-of-speech language models that can achieve better ordering of candidate word lists based on text already input and on contextual information. Simulations using various sizes of keypad designs have shown that the likelihood of users finding their desired words at the top of predictive disambiguation result list has significantly improved, and that the average number of extra keystrokes needed to cycle through candidate word lists has been reduced. Significant increases in text entry performance have also been seen in our empirical usability experiment using this new method with a reduced three-key keypad. In addition to allowing text entry on ultra-small devices, keypad-based text entry methods that use very few keys are potentially beneficial to users with limited arm and/or hand movement. Thus, the results of this dissertation research may increase the universal usability of particular types of information systems and devices, especially for motor- (and possibly visually-) impaired users.

In the third research area, a novel metric for text entry error analysis was created and analyzed. The metric uses the number of corrections as a new type of unit for analysis, and is capable of addressing some of the outstanding issues that exist with current metrics. Advantages of the error metric include providing more detail on 1) when text entry errors are noticed by users, 2) the amount of user effort required to correct errors, and 3) the ways users handle these corrections.

1.3.2 Organization

The remainder of this dissertation is organized as follows:

Chapter 2 reviews existing literature pertaining to previous work on the problems that are related to this dissertation and that form the foundations for some of the work presented here. Chapter 3 presents the results of our research into alphabetically-constrained keypad designs of different sizes for use with dictionary-based predictive disambiguation text entry methods. These keypad designs are validated by both simulations and empirical user studies. Chapter 4 describes our improved dictionary-based predictive disambiguation methods. These methods utilize not only word frequency information, but also semantic
relatedness and part-of-speech information, in order to improve the accuracy of predicting users’ desired words based on existing textual context. Empirical effectiveness of these methods is also validated by empirical user studies. Chapter 5 presents our novel text entry error metric. This error metric is applied to several data sets from our user studies to investigate its effectiveness. The metric provides further insights into the nature of error correction mechanisms of different text entry designs. Chapter 6 summarizes this dissertation research, reiterates its contributions to the field, and discusses possible extensions of this work and other future research in the area of text entry on mobile devices.
Chapter 2

Related Work

In this chapter, background research related to the dissertation work will be summarized. This includes popular metrics for text entry error analysis, previous research on the keypad design problem, and current techniques for predictive disambiguation text entry. Limitations of existing research will be pointed out where relevant to our dissertation work.

2.1 Measuring the Effectiveness of Text Entry Methods

Common metrics for evaluating the theoretical effectiveness of a keypad design used with predictive disambiguation text entry methods are Disambiguation Accuracy and Key Strokes per Character. They tell us theoretically how much effort users will have to devote to use a certain text entry method. These metrics both assume that the text entry is done correctly (i.e., the user makes no errors) and that the dictionary contains all words needed by the user. These two metrics are corpus-based, and can be computed from a corpus using word frequency information.

2.1.1 Disambiguation Accuracy (DA)

DA specifies the probability that after any keystroke sequence is entered, the desired word will be displayed. This assumes the disambiguation process works in such a way that if an ambiguous keystroke sequence is entered, the matching word with highest frequency of occurrence (based on the corpus) will be displayed. Given this, larger DA values are equated with better keypad designs.

2.1.2 Key Strokes Per Character (KSPC)

KSPC is the average number of keystrokes needed to enter a single character [74]. For dictionary-based predictive disambiguation methods, since additional keystrokes are often needed to choose from multiple word choices when an ambiguous keystroke sequence is encountered, KSPC values are usually greater
than 1. (Some word prediction or completion techniques, which will be discussed later, can produce KSPC values of less than 1.) Clearly, better predictive disambiguation methods achieve lower KSPC values.

2.1.3 Limitations of Existing Metrics and Methods

While DA and KSPC tell us theoretically how “good” one keypad design is compared to another for a given dictionary, metrics such as task completion time and error rate must be used when empirically testing text entry methods with users. In this regard, keystroke analysis has been particularly useful for judging the effectiveness of text entry methods. Soukoreff and MacKenzie [74, 75] classify input keystrokes into four categories: Correct (C), Incorrect and Not Fixed (INF), Incorrect but Fixed (IF), and Fixes (F). Error rates are calculated from the numbers of keystrokes falling in each of these categories, and can show us how close users come to the theoretical limits of a particular method.

Keystroke level error metrics are relatively simple. They can be easily computed and the input transcriptions can be automatically processed by computer programs [e.g., 75]. However, metrics at this level also have limitations in that they often cannot reveal additional useful information as to how a user is actually entering text, and what types of errors are occurring.

2.2 Predictive Disambiguation\(^1\) Text Entry Methods

Research on predictive disambiguation text entry methods can be traced back to work by Shannon in 1951 [71], which talked about the predictability of English text. In 1976, Rabiner [64] discussed the ambiguity problems of using the telephone keypad to enter text. Later research looked at improving text entry performance by creating better keypad designs and disambiguation algorithms. Levine et. al. [47] briefly described a technique with a reduced computer keyboard containing more than one letter on a single key. The technique determined which letter on a key was most likely desired by using both an English dictionary and letter trigram (sequences of three letters) statistics. The authors also tried improving

\(^1\) Noted here is that the term “Predictive Disambiguation” being used in this thesis refers to the broad sense of the word "predictive" (See MacKenzie and Tanaka-Ishii [20]).
keyboard layout to minimize word ambiguity by using a Genetic Algorithm-based method to place characters on keys. Afterwards, these same researchers [48] compared two text entry methods – standard QWERTY keyboard text entry versus predictive disambiguation text entry with a reduced keyboard. Results showed that although predictive disambiguation text entry required greater mental load, keyboard designs that put multiple characters on a single key still offered advantages. A benefit of reducing the total number of keys was minimizing the physical movement required for motor-impaired users to enter text.

Hasselgren et al. [32] proposed a predictive disambiguation text entry system for Swedish. It used the same mechanism as other dictionary-based systems, along with word bigram and unigram data for the Swedish language to help disambiguate lists of matching words. One problem noted, however, was the large memory requirements necessary to implement the method.

In contrast to this, Eatoni Ergonomics [53] developed LetterWise, a method based on English character digram probabilities. The system predicts a most likely character from the characters on the key being pressed. Prediction is based on the preceding two or three characters. Since the disambiguation process works after every keystroke and does not use a dictionary, LetterWise requires significantly less memory and works well even with words that are not in the dictionary. However, prediction accuracy is worse and the number of keystrokes is higher than those found in the system proposed by Hasselgren [32], an apparent trade-off for the smaller memory requirements and reduced processing time.

WordWise is another dictionary based disambiguation method from Eatoni [42]. Here, the most frequent characters on each key were labeled, as shown in Figure 2.1. These characters can be input directly if a "mode" key is pressed along with a normal key. This reduces the number of ambiguous keystrokes, which helps the disambiguation of the whole word. However, the “chording” type of keypad may require more learning time than the normal keypad design.
Figure 2.1: WordWise. The “1” key is used to input the grayed letter on other buttons [60]

As an alternative, Gong and Tarasewich proposed a MultiTap text entry system that used dynamic keypad highlighting [26]. The system predicted the most probable next characters on each key based on the last several characters entered, and dynamically highlighted the predicted characters on the cell phone keypad. Highlighted characters become the default characters of those keys that contain them. A similar system used with full soft keyboards on PDA and Tablet PC has been developed by WordLogic [82].

In 2005, researchers in Singapore [36] reported on several contributions made to the field of mobile text entry. One of these was the collection of the first substantial corpus of text messages sent via Short Messaging Service (SMS), which was used in their studies. The authors also proposed an Operation Level Model for measuring the efficiencies of mobile text entry methods, and tried to improve existing methods by remapping the English characters to different keys and by employing predictive word completion based on the last word that was entered.

Rau and Skiena [65] constructed a sentence based disambiguation system, of which the space character was also ambiguous (grouped with some other characters on a single key). Therefore, the disambiguation process needed to decide word boundaries first, and then determine the words from resulting subsequences. For example, assume the space character is grouped together with the character “z” on key 8. Then for the keystroke sequence “235867,” the disambiguation method will have to decide whether key 8 is intended as a space character, in which case the sequence is composed of two words “235” and “67,” or the character “z,” in which case the sequence is only one word. Key sequences are matched using
word-pair frequency (bigram) information and grammatical/linguistic constraints. Surprisingly good disambiguation accuracy was reported.

POBox [60] is a multi-platform text entry system. Its prediction algorithm works on partly transcribed words (such as "mdtrn" for the word "Mediterranean"), and uses the context of the document to determine the correct word. In the case of multiple possible interpretations, a list of candidate words is presented to the user for selection.

Shieber and Baker [72] proposed an abbreviated predictive disambiguation text entry method. The difference from other predictive methods is that instead of typing full words, all vowels and double consecutive letters are left out. Therefore, users are actually typing abbreviations instead of real words. For example, to input the word "association", users would only type "asctn". This method reduced the expected total number of keystrokes per character, although users needed to ensure that the proper characters (i.e., vowels and double consonants) were left out of a word entry. Such a method would most likely work well only after sufficient training time, and could cause user frustration.

2.3 Keypad Design

Past research has looked at improving text entry performance by creating keypad designs that reduce the number of keystrokes needed to enter a word. The term "design" in this section can refer to either the physical shape or layout of keys, or the mapping of characters to keys.

There are a number of studies that have looked at physical layout optimization. Hirotaka [34] analyzed the physical properties of human hands and proposed a new mobile phone design that could be operated single-handedly with greater ease and effectiveness. Himberg et al. [33] presented an adaptive touchscreen keyboard, which automatically adjusted the sizes, shapes, and positions of the keys, to make them better suited to a particular user's usage pattern over time.

As mentioned earlier, research on optimizing the mapping of characters to keys dates back to work that originally looked at ways to help motor-impaired users enter text. Researchers [48, 47, 21] described the
optimization of character placement and the development of disambiguation algorithms on various-sized keypads using word dictionaries and statistics. Light and Anderson [50] tried to accomplish similar keyboard design optimization by using simulated annealing. They considered the benefits of assigning common English letters to the dominant hand and stronger fingers, and assigned common letter pairs to alternate hands. In other early research on this problem, Arnott et al. [2] did a series of detailed experiments on reduced keypads and disambiguation methods. Performances of four keypad layouts (TOC, Frequency, Levine, and Alphabetical, shown in Figure 2.2) were compared.

Lesher et al. tried a different solution for optimizing the character arrangement of ambiguous keyboards [45]. If two characters are to be placed on a same key, their corresponding “conflicting values” from a pre-computed table will become part of the total performance score. The authors also proposed an n-opt optimization algorithm, which was shown to be quite effective, deriving good (but still sub-optimal) layouts in a matter of several seconds.

Researchers have also looked at “nontraditional” keypads, such as a watch-top text entry system that functions similarly to T9, as shown in Figure 2.3 [17]. Other research on small text entry devices with
very few keys includes a study by Mackenzie [54], which compared six text entry methods, all of which used only three keys.

For the rest of this thesis, the term "keypad design" refers only to the mapping of characters to keys, and not to physical keypad properties.

2.4 Genetic Algorithm-based Heuristic

A Genetic Algorithm (GA) simulates biological evolution behaviors such as reproduction, crossover, and mutation to quickly search for solutions to complex problems. Initial work in this field can be traced back to Holland [35]. GA starts with a set of (usually random) possible solutions to a given problem, with each solution represented by a string of bits or characters. A fixed number of solutions that are the most “fit,” i.e. the best current solutions to a problem, are carried over to the next generation and chosen to reproduce. In other words, the best current solutions are saved, and are used to generate new solutions. These strings (current solutions) can be altered to produce children (new solutions) using genetic operators. One of these genetic operators is crossover, in which a randomly chosen sequence of bits is exchanged between two strings to produce one or two new strings. Another operation is mutation, which randomly alters bits within a single string to produce a new string. The new population is evaluated, a new reproducing population is chosen, and the process of crossovers and mutations repeats for a predetermined number of iterations or until an acceptable evaluation level is reached.

The Genetic Algorithm approach to problem solving has some advantages over other optimization and search procedures [24]. GA searching is done from a population of points, rather than a single point (as with branch-and-bound and other techniques), which increases exploratory capability. Payoff (objective function) information is used directly for evaluation, rather than derivatives used by gradient search techniques. Genetic algorithms evaluate specified candidate solutions completely, versus building profiles one attribute at a time. Optimization problems, such as the product design problem, that are characterized as discontinuous, high-dimensional, and multi-modal are especially suited for GA as opposed to gradient or random search techniques [3].
Besides early work done by Levine et al. [74, 75] that uses genetic algorithm-based heuristic to find solutions to the keypad design problem, researchers have also used other popular optimization techniques to solve the keypad design problem, such as simulated annealing [50] and the n-opt algorithm [45].

The three most important components to any genetic algorithm-based heuristics are the problem representation, the crossover operator, and the mutation operator. The pseudocode for a genetic algorithm-based heuristic is shown in Figure 2.4.

```plaintext
Initialize current population containing $n$ random solutions $P = \{p_1, \ldots, p_n\}$.

Initialize the threshold for mutation operation $R$.

Loop for $T$ times ($T$ is an arbitrary number giving how many iterations the genetic algorithm-based heuristic will run for)

For $p_i \in \{p_1, \ldots, p_n\}$

For $p_j \in \{p_1, \ldots, p_n\}$ and $p_j \neq p_i$

$p_{new} = CrossOver(p_i, p_j)$

Generate a random number $RandomNum$

If $RandomNum > R$, $p_{new} = Mutate(p_{new})$

Add $p_{new}$ into $P$

End For

End For

Examine $P$, keep only the best $N$ individuals

Set the new population as the current population

End Loop

Return the best individual from $P$ as the optimized result
```

Figure 2.4: Pseudocode for Genetic Algorithm-based Heuristic
### 2.5  Text Entry with Small Keypad of Few Keys

Text entry methods that use small keypads of very few keys have gained popularity among users of ultrasmall mobile devices, and is important to those who are physically challenged and have special requirements for the devices, interfaces, and/or interaction methods. For example, as mentioned in Section 2.3, Dunlop [17] developed a 5-key text entry system on a watch top with good text entry performance, and keypad design optimization can be traced back twenty years to research aimed at helping motor impaired users with text entry [48, 47].

Sandnes et. al. [67] provided a good overview and comparison of the (then) available text entry methods used on small devices with few keys (usually 3 or 4). This work found a trade-off between the usability of a method and the average KSPC needed. In other words, methods that used fewer keystrokes for each character input demanded more attention from the user, therefore making them harder to use. The authors also created a method for graphically modeling text entry methods used with portable and miniature devices using finite state automata [68].

In 2004 and 2005, Evreinova et al. [18, 19] published work on a 4-key selection text entry method that can be used by physically challenged users. Available characters were placed into groups, and users used top-down selection to specify the characters that they desired (i.e., users first selected a larger character group, then a sub-group, repeating the process until they found the desired character). In order to assist visually impaired users, she also proposed a method which used color blinking as augmentative feedback about the character just input.

A recent publication by Baljko [4] talked about a method of optimized character arrangement for text entry using two keys to help motor impaired users reduce the number of keystrokes needed. This selection text entry method is similar to the one proposed by Evreinova [18].
2.6 Word Prediction and Completion

In cases where an ambiguous key sequence (corresponding to multiple matching words) is entered, predicative disambiguation algorithms determine the most likely desired word. Most algorithms choose the word with the highest frequency of occurrence among all matching words based on a given corpus. Digram based disambiguation, at the character or word level, has also proven to be quite effective. Shannon first discussed the problem of English language entropy and analyzed N-Gram character-level prediction in his 1951 paper [71].

Unlike predictive word disambiguation, which works on a sequence of keystrokes that is assumed to represent an entire word, word prediction and completion methods predict additional characters or complete words based on partial input or previous keystrokes. However, both methods are similar in the techniques they use. For example, Masui [60] looked at improving mobile pen text entry performance using word prediction. A dictionary of word tokens and an approximate string-matching algorithm were used to achieve word completion based on what users had partially input.

Word prediction and completion is often used in Alternative and Augmentative Communication (AAC), to reduce the keystrokes required for motor impaired users to input text. For example, Boggess [5] experimented with two simple prediction algorithms. Results showed that for a single user’s language, a prediction window of 50 words would give about 50 percent success rate in predicting the next desired word. However, where the first letter of the next word available, a prediction window of 20 words succeeded about 82 percent of the time.

In addition to statistical models, previous research has also tried to build word completion systems based on other types of context, such as that described by Stocky et al. [76], which finds best possible completions by semantic information derived from a “commonsense knowledge base.”
2.7 N-Gram Language Model

The N-Gram model and its variants have achieved substantial success because of their simplicity and effectiveness. The general form of an N-Gram model can be defined as:

\[
p(w) = p(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} p(w_i \mid w_{\text{max}(1, i-N+1)} \ldots w_{i-1}) = \prod_{i=1}^{n} p(w_i \mid h_i)
\]  

(1)

where \( w_i \) is the \( i \)th word, and \( h_i \) is its history. For the N-Gram model, it is assumed that the probability of \( w_i \) only depends on its latest \( N \) preceding characters, therefore \( h_i = w_{\text{max}(1, i-N+1)} w_{i-N+1} \ldots w_{i-1} \).

The N-Gram Language Model has many variants, such as the Class-Based N-Gram Model [16], Grammatical Link N-Gram Model [34], and the Sequence N-Gram Model [59].

Lesher et al. [46] performed research with the N-Gram model that showed the effects of training text size as well as order on a digram model. They empirically showed that a unigram model can be saturated easily – meaning that the performance of a unigram model may stop improving quickly – whereas bigram and trigram models continue to improve with more training data. This result is consistent with what Shannon reported [71].

Carlberger et al. [10] described the Profet word prediction system based on the N-Gram model. They found that although keystrokes were saved, a speedup in communication rate was not always achieved. However, they found that word prediction greatly reduced physical fatigue and improved the grammaticality of users’ sentences.

Klarlund et al. [41] described research in which N-Gram models were used for predictive text entry disambiguation on reduced QWERTY keyboards of different sizes. The authors showed that error rates can be reduced to 0.5%, which is about \( \frac{1}{4} \) of that of unigram models. Their work is similar to some of that done in Chapter 4 of this dissertation.

Seymore et al. [69] pointed out that with increases in the order of the N-Gram language model, it would begin to require more working memory than practical system capacities. Their research investigated the
degradations of a trigram back-off model's perplexity as the sizes of the bigram and trigram models were reduced. More importantly, the paper investigated alternative ways of excluding digram entries using criteria other than the frequency of specific digram entries.

2.8 Semantic Relatedness Model

Context is another factor to use with mobile text entry systems to improve word predictions. In this case, context has two meanings: the physical context (such as location, time, or weather) and textual context (text surrounding ambiguous words). Making use of physical context at this time is rather difficult. For example, research described by Dominowska et al. [15] used contextual information to aid the text communication between computers and disabled users by automatically adjusting available vocabularies and interface layouts, but was done using preprogrammed context data related to location. While using physical context in situations such as this is potentially beneficial, there are still many hurdles to overcome in terms of sensing, interpreting, and applying such context data.

On the other hand, previous research suggests that better prediction results are achieved using textual context, so part of this dissertation research uses textual context to further improve text entry methods. Recent work by Budanitsky [8] surveyed five WordNet-based measures of lexical semantic relatedness. An information-content-based measure proposed by Jiang and Conrath [39] was found superior to those proposed by Hirst and St-Onge [30], Leacock and Chodorow [12], Lin and Resnik [14]. In Manning and Schutze [59], "semantic relatedness" is defined as entities in the world that are likely to co-occur. Therefore, compared with a model of the real meanings of each word, a simpler model might be just based on the words' co-occurrences in some large corpora.

In research proposed by Li and Hirst [49], an N-Gram model was combined with a semantic knowledge model to achieve prediction of future words that are semantically much more appropriate than the N-Gram model alone yields. Algorithms for finding the semantic relatedness between English word pairs were proposed, which mainly used the text co-occurrence information. Significant improvements with the integrated model were reported.
2.9 Part-of-Speech Models

Various Part-of-Speech (POS) models are also widely used by researchers in the Natural Language Processing (NLP) field. There are many approaches to automatic POS tagging problem, which can usually be categorized into two types: Rule-based Tagging and Stochastic Tagging.

2.9.1 Rule-based Tagging

Rule-based tagging systems use contextual information, also known as context rules, to assign tags to unknown words. For example, a context rule may be:

- If an unknown word is preceded by a determiner and followed by a noun, it is an adjective.

In addition to contextual information, many rule-based tagging systems use morphological information, or even capitalization and punctuation to aid in the tagging process. For example, tagging rules using morphological or capitalization information may be:

- If an unknown word ends with “tion” then it is a noun.
- If an unknown word contains all capital letters then it is a noun.

One typical example of a rule-based tagging system is the Brill TBL Tagger created by Eric Brill [6]. It has a transformation-based learning algorithm and is designed to automatically learn a set of context rules along with the order that they should be applied from a tagged training corpus. It works by first assigning the most likely part-of-speech tag for each word, then applying all of the context rules in order. State-of-art rule based taggers can usually achieve about 97% accuracy.

2.9.2 Stochastic Tagging

Tagging systems that incorporate frequency, statistics, or probability are categorized as stochastic tagging systems. A simple stochastic tagging system may be built based on the probability of the occurrence of a given sequence of tags. This is a form of the N-Gram/Hidden Markov Model-based approach (where the best tag for a given word is determined by the N previous tags). The most common algorithm for
implementing an N-Gram approach is the Viterbi Algorithm [79]. The Viterbi Algorithm is a dynamic programming algorithm, which states that the best path through a particular intermediate place (I) is the best path to it from a starting place (S), followed by the best path from I to the end (E). Therefore, compared with other algorithms, it saves the cost of storing all other paths that are not the minimum-length path from S to E, which greatly improves the efficiency of the tagging process. A mathematical description of the Viterbi Algorithm for Bi-Gram POS tagging [59] is given below in Figure 2.5 and Figure 2.6:

\[
\begin{align*}
  w_i & : \text{An English word.} \\
  t_0 & : \text{A special tag for sentence delimiter.} \\
  t_i & \in \{t_1 \ldots t_n\} : \text{one of the n possible Part-of-Speech tags.} \\
  S = t_0 w_1 w_2 \ldots w_i : \text{A sentence of } l \text{ words that needs to be part-of-speech tagged.} \\
  X = X_1 \ldots X_i : \text{a list of } l \text{ most appropriate tags that are chosen for each word.} \\
  P(t_j | t_k) : \text{The probability of the POS Bi-Gram } t_k t_j \text{ in the training corpus.} \\
  P(w_i | t_j) : \text{The probability of a word of POS } t_j \text{ being the word } w_i \text{ in the training corpus.} \\
  \delta_i (t_j) : \text{A function that gives the probability of the most likely tagging for the sentence } \\
  S = t_0 w_1 w_2 \ldots w_{i-1} w_i \text{ such that } w_i \text{ is supposedly POS tagged as } t_j. \\
  \psi_{i+1} (t_j) : \text{A function that gives the most likely tag at } w_i \text{ given that } w_{i+1} \text{ is supposedly POS} \\
  \text{tagged as } t_j.
\end{align*}
\]

Figure 2.5: Definitions for the Viterbi Algorithm for POS tagging [59]
POSTagger\( (S = w_1w_2...w_l) \)

\[ \delta_i(t_0) = 1.0 \]

\[ \delta_i(t_i) = 0.0 \text{ for } 1 \leq i \leq n, \]

For \( i = 1 \) to \( l \), step 1, do

For \( t_j \in \{t_1..t_n\} \), do

\[ \delta_{i+1}(t_j) = \max_{1 \leq k \leq n} \{\delta_i(t_k) \times P(t_j | t_k) \times P(w_{i+1} | t_j)\} \]

\[ \psi_{i+1}(t_j) = \arg \max_{1 \leq k \leq n} \{\delta_i(t_k) \times P(t_j | t_k) \times P(w_{i+1} | t_j)\} \]

End For

End For

// Retrace the best path back from \( w_l \) to the sentence delimiter

// to find out the most likely POS tags for each word

\[ X_{n+1} = \arg \max_{1 \leq j \leq n} \delta_{n+1}(j) \]

For \( j = n \) to 1, step −1, do

\[ X_j = \psi_{j+1}(X_{j+1}) \]

End For

End Function

Figure 2.6: Viterbi Algorithm for POS tagging [59]
2.9.3 Applications of Part-of-Speech Tagging

It is widely believed that many natural language processing applications will benefit from syntactically disambiguated text [59]. Therefore, Part-of-Speech tagging has been used in various NLP research applications, such as partial parsing [1], information extraction [9], and question answering [43].

2.10 Corpora

Large text corpora are necessary to train a language model and derive a semantic relatedness model. Such corpora should be representative of the text that is used for specific tasks [55]. Popular general-purpose corpora that have been widely used for text entry research include:

- The British National Corpus (BNC) [7]
- The Brown Corpus [22]
- Reuters Corpus [66]
- Phrase Sets for Evaluating Text Entry Techniques (MacKenzie and Soukoreff [57])

Text messaging appears to be the most prevalent text entry task on mobile phones. The best corpora for this dissertation work are therefore a collection of actual short messages sent over mobile phones. But finding such a corpora for academic research proved to be difficult at best. A relatively small but publicly available SMS corpus that was found is:

- National University of Singapore (NUS) SMS Corpus [36, 40]

For the purposes of this dissertation work, the written and spoken corpora of the BNC corpus and the NUS SMS Corpus were used to test the effects that different corpora have on a language model when used in different domains. Although the NUS SMS corpus may not be a perfect representation of English short messages, it was the best choice available when our initial research [e.g., 25] was performed. The BNC Corpus was used to implement and test the semantic relatedness predictive disambiguation model described in Chapter 4. Since the new version of the BNC corpus is POS tagged, it is also appropriate for
testing the effectiveness of POS rules in helping the disambiguation process. The MacKenzie and Soukoreff phrase set [57] was used in user studies to test the empirical performance of the improved predictive disambiguation text entry methods because of its popularity among text entry researchers (allowing comparison of results across studies). This set contains about 500 phrases, and is artificially designed to have a high correlation with common English letter frequency counts.
Chapter 3

Alphabetically Constrained Keypad Design

3.1 Introduction

For any given keypad-based text entry method, the keypad design (and character mapping) plays a large part in determining the overall efficiency and usability of the method in the hands of the user. This chapter presents our research work on creating keypad designs that not only perform well in the hands of experts but also have good novice usability and small learning curves.

The next part of this chapter (Section 3.2) discusses problems and limitations with current keypad designs used for predictive disambiguation and MultiTap text entry methods. Section 3.3 presents methods we used for designing keypads that overcome some of these problems and limitations. Section 3.4 discusses the different corpora used in this research and their effects on keypad designs. This is followed by results and analysis of keypad designs that were found using our corpora and methods (Sections 3.5 and 3.6). The results of two usability studies conducted to empirically validate the keypad designs are given in Section 3.7. A summary of this research and possible future work are discussed in Section 3.8.

3.2 Finding Keypad Designs

For all text entry design and experimentation in this dissertation work, only letters from English alphabet were used (no numbers or punctuation). However, the work should be extensible to a fuller character set and to other languages as well, although the complexity of the problem space could potentially increase. Disambiguation Accuracy (DA) and Key Strokes Per Character (KSPC) were both investigated as measurements of the performance of different text entry methods.
3.2.1 Keypad Designs for Predictive Disambiguation Text Entry Methods

The predictive disambiguation keypad design problem places $M$ letters on $N$ keys such that some performance objective (DA or KSPC) is maximized (or minimized). The keypad design problem for the predictive disambiguation text entry method is difficult because if more than one letter can be placed on a given key, a given key sequence might correspond to many possible words. Moving letters between keys can reduce the possibility of this happening. The *unconstrained* version of the keypad problem allows each letter to be placed on any key; the *constrained* version requires letters to remain in alphabetical order across all keys. Previous studies have shown that users of predictive disambiguation keypad methods with unconstrained letter placement can achieve high performance, but only after much practice [38]. We therefore hypothesized that constrained designs should increase usability and lessen learning time for novice users. This seemed reasonable given the work of Smith and Zhai [83], who found that a virtual keyboard created with alphabetical ordering tendency (i.e., keeping letters close to alphabetical order), while not optimal for advanced users, did increase performance and preference ratings for novice users. In addition, alphabetization should allow novices to build expertise more quickly.

3.2.2 Keypad Designs for the Multi-Tap Text Entry Method

Similar to that for predictive disambiguation methods, the keypad design problem for the Multi-Tap text entry method has also been widely studied. The most common solutions include the optimization of character placements on physical keys, so that characters with higher frequencies of occurrence can be entered with fewer same-key keystrokes [45, 26].

The alphabetical constraint can also be applied to the keypad design problem for multi-tap text entry. It not only provides a benefit to novice users, such as smaller learning curves, but also requires significantly less time to compute optimal constrained keypad designs of various sizes, when compared to time required for computing the unconstrained designs.
3.3 Finding Solutions

3.3.1 GA-based Heuristic for Finding Unconstrained Keypad Solutions for Predictive Disambiguation Text Entry Methods

For predictive disambiguation text entry methods, the problem of arranging characters on keys belongs to the NP-complete class of mathematical optimization problems [45]. The only way to guarantee finding an optimal solution is by searching every possible arrangement for the best one. Enumerating the total number of different unconstrained keypad arrangements quickly becomes computationally unrealistic as the number of letters and keys increases. Instead, a GA-based heuristic was used to find solutions to this problem. Any GA-based heuristic requires three important components for it to work: the representation of each individual in the population, the crossover operation, and the mutation operation. These are discussed in detail below.

The GA-based heuristic uses a keypad representation of an array of 26 integers. The \( i \)th integer indicates the key that the \( i \)th English character belongs to, as shown in Figure 3.1.

<table>
<thead>
<tr>
<th>ABCD</th>
<th>EFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIJKL</td>
<td>MN</td>
</tr>
<tr>
<td>O</td>
<td>PQRS</td>
</tr>
<tr>
<td>TUV</td>
<td>WXYZ</td>
</tr>
</tbody>
</table>

![Keypad Representation Used by the GA-based Heuristic](image)

Figure 3.1: Keypad Representation Used by the GA-based Heuristic

For a crossover operation, two “parent” keypad designs “reproduce” to form a new “child” keypad design. Each character of the new design is assigned from either the first design or the second design. An example of this is shown in Figure 3.2. The shaded bits are those picked from each parent.
The mutation operation for deriving potential solutions for the unconstrained keypad design problem involves randomly choosing (using a given probability) one character from the array and assigning it to another randomly chosen key. As shown in Figure 3.3, the shaded character is moved from key 1 to key 5.

These operations were used with the standard Genetic Algorithm heuristic introduced in Section 2.4 to derive good unconstrained keypad designs of various sizes (i.e., various numbers of keys).

3.3.2 Finding Constrained Solutions for Predictive Disambiguation Text Entry Methods

Fortunately, it is easier to completely enumerate all of the constrained keypad designs for a given number of characters and keys. The constrained problem can be viewed as placing N-1 dividing positions between
a sequence of M characters (as shown in Figures 3.4 and 3.5), which can be solved to completion in a more reasonable time.

![Diagram of 26 English characters with dividing positions](image)

Figure 3.4: 26 English characters (A - Z) have 25 dividing positions (D1 – D25)

For example, for M=26 and N=8 (the standard mobile phone keypad design), the unconstrained problem has $\approx 1.6 \times 10^{20}$ possible solutions, while the constrained design has 480,700. While large problems still took several days to solve, we used complete enumeration to calculate all possible constrained keypad designs for varying numbers of keys. The program simply generated all of the qualifying keypad designs, checked the performances measurements of the designs, and tracked the best ones.

![Diagram of 7 dividing positions and 8 keys](image)

Figure 3.5: Picking 7 Dividing Positions Results in an 8-Key Constrained Keypad Design

### 3.3.3 Finding Constrained Solutions for Multi-Tap Text Entry Method

Finding solutions for optimal alphabetically-constrained keypad designs for the MultiTap text entry method is not as time-consuming as it is for the predictive disambiguation method. MultiTap has the unique property of "subpart independence," such that the number of keystrokes required for entering one character only depends on the character sequence of that same key, and is totally unrelated to the arrangements of other characters. Therefore, with the alphabetical constraint, searching for the best
keypad design under the goal of minimizing KSPC can be done with dynamic programming algorithms in almost real time. Mathematically, the dynamic programming algorithm can be stated as:

\[
\text{OptimKSPC}(i, j) = \min_{1 \leq k \leq i} \{ \text{OptimKSPC}(k - 1, j - 1) + \text{KSPC}(k, i) \}
\]

\[
\text{OptimChar}(i, j) = \arg\min_{1 \leq k \leq i} \{ \text{OptimKSPC}(k - 1, j - 1) + \text{KSPC}(k, i) \}
\]

Where \( \text{OptimKSPC}(i, j) \) is the KSPC value of the optimal \( i \)-key alphabetically-constrained keypad design for characters 1 to \( j \) for the Multi-tap text entry method. \( \text{OptimChar}(i, j) \) is the first character on the last key of this optimal design, which can be used to recursively calculate an entire design. \( \text{KSPC}(k, i) \), is the KSPC value for characters \( k \) to \( i \), if they are arranged on a single key. Note that two base cases for the above equation have to be defined here as: \( \text{OptimKSPC}(0, j) = 0 \) for all \( j > 0 \) and \( \text{OptimKSPC}(i, 1) = \text{KSPC}(1, i) \) for all \( i > 0 \).

### 3.4 Corpora

One limitation of most text entry research for mobile devices is that frequency information for words and phrases comes from text corpora derived from publications such as newspapers and books. But text entry in the mobile environment can be quite different from that in the desktop environment in terms of words used and their frequency of usage. Text entry can also vary based on the context of use, and the task being performed. To see what effect different task environments might have on text entry interface design, we used three distinct English word lists in this research. One list was derived from written language, one from spoken language, and one from short text messages.

The written and spoken word lists used in our study are derived from the British National Corpus (BNC) [44], which contains approximately 65,000 distinct words and their frequencies of occurrence. These words are derived from a total sample of 100 million words of present-day English, of which approximately 90 million come from books and ten million come from conversations and monologues. The written word list, with a total of 4420 words, contains all words that appear with a frequency of at least 20 times per million written words. The spoken word list, with a total of 4307 words, contains all
words that appear with a frequency of at least 10 per million spoken words. While the cutoff frequencies used by these two word lists are different, they result in two lists of very close sizes. These two lists are shorter than the complete word lists derived from the BNC, however the written word list used in our study accounts for approximately 84% of all written words, and the spoken list accounts for about 96% of all spoken words. The written word list is similar to those used in many previous mobile device text entry studies. The spoken word list should better reflect the mobile environment in terms of word usage and frequencies because speech is usually less formal than writing.

The SMS word list, which contains 7325 words, was derived from a corpus consisting of about 10,000 SMS messages collected as part of a research study in Singapore [36, 40]. The study looked at ways to improve SMS messaging on mobile phones by remapping characters to keys (but without alphabetization constraints). Two-thirds of the SMS corpus was collected from several regular users of mobile phones. The rest was primarily collected over the Web from a larger sample of other solicited mobile phone users. The corpus was processed to create a word list and a set of associated frequencies.

There are several differences that stand out when comparing the spoken, written, and SMS word lists. Abbreviations, such as “hrs” for “hours”, “vry” for “very”, and “u” for “you” occur frequently in the SMS corpus. Abbreviations containing periods (e.g., “U.S.A.”), and large numbers (e.g., “12,345”), occur only in written language. On the other hand, single letters, such as “U,” “S,” and “A” tend to occur only in speech [44]. Significant differences in word frequencies also occur between the three lists. For example, the word “you” occurs 25,957 times in one million spoken words, about 9700 times in one million SMS words, and only 4,755 times in that many written words. The word “yeah” occurs 7,890 times in speech and only 17 times in written text per one million words.

Two modifications were made to all three lists before they were used in our study. First, all punctuation was removed and any upper-case characters were converted into lower-case. So occurrences such as “let’s” and “U.S.A” became “lets” and “usa”. Second, any phrase found on the list was removed, and its frequency of occurrence was added to the frequencies of all its member words. For example, if the phrase
“in addition to” appeared ten times, ten was added to the frequencies of “in”, “addition”, and “to”. This modification, which ensured that dictionaries consisted only of words and not phrases, was necessary because the predictive disambiguation text entry method studied works only with word-based input.

3.5 Keypad Design Results

3.5.1 Constrained Keypad Design Results for the Predictive Disambiguation Text Entry Method

Using complete enumeration, optimal keypad designs for predictive disambiguation methods were found for keypad sizes ranging from one to twelve keys. Disambiguation accuracy was maximized, using each of the three word lists, to find the optimal keypad designs (DA values are shown in Table 3.1). Selected optimal designs (for four, eight, nine, and twelve keys) for all three word lists are shown in Table 3.2.

Complete enumeration was then used to find optimal keypad designs for predictive disambiguation text entry methods using the KSPC metric. Table 3.3 shows the minimum KSPC values found for keypad designs having one to twelve keys across the three different word lists. Table 3.4 shows selected optimal designs corresponding to these KSPC values.

3.5.2 Unconstrained Keypad Design Results for the Predictive Disambiguation Text Entry Method

Since the search space for unconstrained keypad designs for the predictive disambiguation text entry method is so large, it is impractical to use complete enumeration to find optimal layouts. Therefore in order to compare the optimal constrained keypad designs with the unconstrained designs that are derived using the same performance metric, as described before, a GA-based heuristic was used to find good (near-optimal) solutions to the unconstrained keypad design problem using the DA metric. Maximum disambiguation accuracy values for each word list are shown in Table 3.5. Selected optimal keypad designs for the predictive disambiguation method are shown in Table 3.6.

3.5.3 Alphabetically Constrained Keypad Design Results for the Multi-Tap Text Entry Method

Using a dynamic programming algorithm, the optimal keypad designs for the Multi-Tap text entry method were found for keypad sizes ranging from one to twelve keys. The KSPC metric was minimized,
using each of the three word lists, to find the optimal keypad designs (KSPC values are shown in Table 3.7). Selected optimal designs (for four, eight, nine, and twelve keys) for all three word lists are shown in Table 3.8.

<table>
<thead>
<tr>
<th>Keys</th>
<th>Written</th>
<th>Spoken</th>
<th>SMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17.29%</td>
<td>14.60%</td>
<td>7.79%</td>
</tr>
<tr>
<td>2</td>
<td>50.72%</td>
<td>45.77%</td>
<td>31.42%</td>
</tr>
<tr>
<td>3</td>
<td>74.38%</td>
<td>69.45%</td>
<td>54.36%</td>
</tr>
<tr>
<td>4</td>
<td>85.35%</td>
<td>79.89%</td>
<td>68.71%</td>
</tr>
<tr>
<td>5</td>
<td>91.10%</td>
<td>87.25%</td>
<td>78.13%</td>
</tr>
<tr>
<td>6</td>
<td>93.85%</td>
<td>91.97%</td>
<td>84.84%</td>
</tr>
<tr>
<td>7</td>
<td>95.96%</td>
<td>95.04%</td>
<td>89.55%</td>
</tr>
<tr>
<td>8</td>
<td>97.11%</td>
<td>96.69%</td>
<td>92.49%</td>
</tr>
<tr>
<td>9</td>
<td>97.70%</td>
<td>97.72%</td>
<td>94.02%</td>
</tr>
<tr>
<td>10</td>
<td>98.13%</td>
<td>98.41%</td>
<td>95.13%</td>
</tr>
<tr>
<td>11</td>
<td>98.51%</td>
<td>99.01%</td>
<td>96.11%</td>
</tr>
<tr>
<td>12</td>
<td>98.76%</td>
<td>99.32%</td>
<td>96.85%</td>
</tr>
</tbody>
</table>

Table 3.1: Maximum DA values for 1 to 12 key constrained keypad designs for the predictive disambiguation text entry method by word list

<table>
<thead>
<tr>
<th>Keys</th>
<th>Written</th>
<th>Spoken</th>
<th>SMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>ABCDEF</td>
<td>ABCDEF</td>
<td>ABCDEFG</td>
</tr>
<tr>
<td></td>
<td>GHIJKLMN</td>
<td>GHIJKL</td>
<td>HJKLM</td>
</tr>
<tr>
<td></td>
<td>OPQRS</td>
<td>MNOPQ</td>
<td>NOPQRS</td>
</tr>
<tr>
<td></td>
<td>TUVWXYZ</td>
<td>RSTUWXY</td>
<td>TUVWXYZ</td>
</tr>
<tr>
<td>8</td>
<td>ABCD</td>
<td>ABCD</td>
<td>ABCD</td>
</tr>
<tr>
<td></td>
<td>IJKL</td>
<td>HIJKL</td>
<td>HIJ</td>
</tr>
<tr>
<td></td>
<td>OPQR</td>
<td>S</td>
<td>PQR</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>UVWX</td>
<td>TUVX</td>
</tr>
<tr>
<td>9</td>
<td>ABCD</td>
<td>EF</td>
<td>ABCD</td>
</tr>
<tr>
<td></td>
<td>GH</td>
<td>IJKL</td>
<td>GH</td>
</tr>
<tr>
<td></td>
<td>MN</td>
<td>OPQR</td>
<td>MN</td>
</tr>
<tr>
<td></td>
<td>S</td>
<td>T</td>
<td>UVWX</td>
</tr>
<tr>
<td>12</td>
<td>AB</td>
<td>CD</td>
<td>AB</td>
</tr>
<tr>
<td></td>
<td>GH</td>
<td>JK</td>
<td>GH</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>O</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>S</td>
<td>TUV</td>
<td>S</td>
</tr>
</tbody>
</table>

Table 3.2: Selected optimal constrained keypad designs (4, 8, 9 and 12 keys) for the predictive disambiguation text entry method for maximum DA values by word list
### Table 3.3: Minimum KSPC values for 1 to 12 key constrained keypad designs for the predictive disambiguation text entry method by word list

<table>
<thead>
<tr>
<th>Keys</th>
<th>Written</th>
<th>Spoken</th>
<th>SMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>67.85</td>
<td>87.94</td>
<td>211.28</td>
</tr>
<tr>
<td>2</td>
<td>4.62</td>
<td>6.64</td>
<td>20.24</td>
</tr>
<tr>
<td>3</td>
<td>1.99</td>
<td>2.66</td>
<td>6.92</td>
</tr>
<tr>
<td>4</td>
<td>1.45</td>
<td>1.71</td>
<td>3.69</td>
</tr>
<tr>
<td>5</td>
<td>1.25</td>
<td>1.41</td>
<td>2.54</td>
</tr>
<tr>
<td>6</td>
<td>1.14</td>
<td>1.26</td>
<td>1.92</td>
</tr>
<tr>
<td>7</td>
<td>1.10</td>
<td>1.17</td>
<td>1.63</td>
</tr>
<tr>
<td>8</td>
<td>1.07</td>
<td>1.12</td>
<td>1.43</td>
</tr>
<tr>
<td>9</td>
<td>1.06</td>
<td>1.08</td>
<td>1.31</td>
</tr>
<tr>
<td>10</td>
<td>1.05</td>
<td>1.05</td>
<td>1.25</td>
</tr>
<tr>
<td>11</td>
<td>1.05</td>
<td>1.03</td>
<td>1.20</td>
</tr>
<tr>
<td>12</td>
<td>1.04</td>
<td>1.02</td>
<td>1.16</td>
</tr>
</tbody>
</table>

### Table 3.4: Selected optimal constrained keypad designs (4, 8, 9 and 12 keys) for the predictive disambiguation text entry method for minimum KSPC values by word list

<table>
<thead>
<tr>
<th>Keys</th>
<th>Written</th>
<th>Spoken</th>
<th>SMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>ABCD</td>
<td>ABC</td>
<td>ABCD</td>
</tr>
<tr>
<td></td>
<td>EF</td>
<td>DEF</td>
<td>EFG</td>
</tr>
<tr>
<td></td>
<td>HI</td>
<td>IJKL</td>
<td>HIJ</td>
</tr>
<tr>
<td></td>
<td>NOPQ</td>
<td>RS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>TUVWXYZ</td>
<td>TU VWXYZ</td>
<td>TU VWXYZ</td>
</tr>
<tr>
<td>8</td>
<td>AB</td>
<td>CDE</td>
<td>AB</td>
</tr>
<tr>
<td></td>
<td>FG</td>
<td>HIJKL</td>
<td>FG</td>
</tr>
<tr>
<td></td>
<td>MN OPQR</td>
<td>S</td>
<td>LM</td>
</tr>
<tr>
<td></td>
<td>T UVWXYZ</td>
<td>TU VWXYZ</td>
<td>TU VWXYZ</td>
</tr>
<tr>
<td>9</td>
<td>AB</td>
<td>CDE</td>
<td>AB</td>
</tr>
<tr>
<td></td>
<td>FG</td>
<td>HIJKL</td>
<td>FG</td>
</tr>
<tr>
<td></td>
<td>MN OPQR</td>
<td>S</td>
<td>LM</td>
</tr>
<tr>
<td></td>
<td>T UVWXYZ</td>
<td>TU VWXYZ</td>
<td>TU VWXYZ</td>
</tr>
<tr>
<td>12</td>
<td>A BCD</td>
<td>EFG</td>
<td>AB CD</td>
</tr>
<tr>
<td></td>
<td>H I JKL</td>
<td>I JKL</td>
<td>GH I JKL</td>
</tr>
<tr>
<td></td>
<td>M NO PQR</td>
<td>M NO PQR</td>
<td>M N O PQR</td>
</tr>
<tr>
<td></td>
<td>S T UVWXYZ</td>
<td>S T UVWXYZ</td>
<td>S T UVWXYZ</td>
</tr>
</tbody>
</table>

Table 3.4: Selected optimal constrained keypad designs (4, 8, 9 and 12 keys) for the predictive disambiguation text entry method for minimum KSPC values by word list
Table 3.5: Maximum DA values (using the GA-based heuristic) for 1 to 12 key unconstrained keypad designs for the predictive disambiguation text entry method by word list.

<table>
<thead>
<tr>
<th>Keys</th>
<th>Written</th>
<th>Spoken</th>
<th>SMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17.29%</td>
<td>14.60%</td>
<td>7.79%</td>
</tr>
<tr>
<td>2</td>
<td>52.39%</td>
<td>48.67%</td>
<td>33.61%</td>
</tr>
<tr>
<td>3</td>
<td>69.69%</td>
<td>72.31%</td>
<td>56.36%</td>
</tr>
<tr>
<td>4</td>
<td>87.39%</td>
<td>85.27%</td>
<td>70.45%</td>
</tr>
<tr>
<td>5</td>
<td>90.17%</td>
<td>87.02%</td>
<td>70.45%</td>
</tr>
<tr>
<td>6</td>
<td>96.10%</td>
<td>96.40%</td>
<td>80.82%</td>
</tr>
<tr>
<td>7</td>
<td>97.20%</td>
<td>97.72%</td>
<td>88.28%</td>
</tr>
<tr>
<td>8</td>
<td>98.05%</td>
<td>98.50%</td>
<td>92.43%</td>
</tr>
<tr>
<td>9</td>
<td>98.55%</td>
<td>98.81%</td>
<td>94.31%</td>
</tr>
<tr>
<td>10</td>
<td>98.78%</td>
<td>99.27%</td>
<td>96.42%</td>
</tr>
<tr>
<td>11</td>
<td>98.84%</td>
<td>99.42%</td>
<td>97.34%</td>
</tr>
<tr>
<td>12</td>
<td>99.04%</td>
<td>99.51%</td>
<td>97.43%</td>
</tr>
</tbody>
</table>

Table 3.6: Selected optimal unconstrained keypad designs (4, 8, 9 and 12 keys) for the predictive disambiguation text entry method for maximum DA values by word list.

<table>
<thead>
<tr>
<th>Keys</th>
<th>Written</th>
<th>Spoken</th>
<th>SMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>ABCDFT</td>
<td>AFTUVXY</td>
<td>ANPT</td>
</tr>
<tr>
<td></td>
<td>EILWY</td>
<td>BCDEL</td>
<td>BFIKS</td>
</tr>
<tr>
<td></td>
<td>GKMNOQVZ</td>
<td>GHKMOS</td>
<td>CEJLMUXZ</td>
</tr>
<tr>
<td></td>
<td>HJPRSUX</td>
<td>JNPQRZW</td>
<td>DGHQGRVW</td>
</tr>
<tr>
<td>8</td>
<td>AQT</td>
<td>AFLX</td>
<td>MOQ</td>
</tr>
<tr>
<td></td>
<td>BNZ</td>
<td>BIT</td>
<td>HS</td>
</tr>
<tr>
<td></td>
<td>CIKY</td>
<td>GPR</td>
<td>GPUY</td>
</tr>
<tr>
<td></td>
<td>DUVW</td>
<td>LM</td>
<td>NWZ</td>
</tr>
<tr>
<td>9</td>
<td>ACJK</td>
<td>BDF</td>
<td>AFP</td>
</tr>
<tr>
<td></td>
<td>EGM</td>
<td>HOX</td>
<td>CN</td>
</tr>
<tr>
<td></td>
<td>ITV</td>
<td>LQY</td>
<td>NW</td>
</tr>
<tr>
<td>12</td>
<td>PRZ</td>
<td>SU</td>
<td>JSVX</td>
</tr>
<tr>
<td></td>
<td>AFP</td>
<td>BG</td>
<td>CE</td>
</tr>
<tr>
<td></td>
<td>DHQ</td>
<td>IJKYZ</td>
<td>LV</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>N</td>
<td>OS</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>T</td>
<td>UWX</td>
</tr>
</tbody>
</table>
### Table 3.7: Minimum KSPC values for 1 to 12 key constrained keypad designs for the Multi-Tap text entry method by word list

<table>
<thead>
<tr>
<th>Keys</th>
<th>Written</th>
<th>Spoken</th>
<th>SMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11.79</td>
<td>12.16</td>
<td>12.07</td>
</tr>
<tr>
<td>2</td>
<td>5.58</td>
<td>5.85</td>
<td>5.94</td>
</tr>
<tr>
<td>3</td>
<td>4.01</td>
<td>4.14</td>
<td>4.22</td>
</tr>
<tr>
<td>4</td>
<td>2.84</td>
<td>3.02</td>
<td>3.05</td>
</tr>
<tr>
<td>5</td>
<td>2.33</td>
<td>2.42</td>
<td>2.54</td>
</tr>
<tr>
<td>6</td>
<td>1.98</td>
<td>2.06</td>
<td>2.19</td>
</tr>
<tr>
<td>7</td>
<td>1.74</td>
<td>1.82</td>
<td>1.88</td>
</tr>
<tr>
<td>8</td>
<td>1.62</td>
<td>1.63</td>
<td>1.69</td>
</tr>
<tr>
<td>9</td>
<td>1.49</td>
<td>1.53</td>
<td>1.58</td>
</tr>
<tr>
<td>10</td>
<td>1.39</td>
<td>1.42</td>
<td>1.47</td>
</tr>
<tr>
<td>11</td>
<td>1.32</td>
<td>1.34</td>
<td>1.38</td>
</tr>
<tr>
<td>12</td>
<td>1.25</td>
<td>1.28</td>
<td>1.32</td>
</tr>
</tbody>
</table>

### Table 3.8: Selected optimal constrained keypad designs (4, 8, 9 and 12 keys) for the Multi-Tap text entry method for minimum KSPC values by word list

<table>
<thead>
<tr>
<th>Keys</th>
<th>Written</th>
<th>Spoken</th>
<th>SMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>ABCD</td>
<td>ABCD</td>
<td>ABCD</td>
</tr>
<tr>
<td></td>
<td>EFGHJK</td>
<td>EFGHJK</td>
<td>EFGHJK</td>
</tr>
<tr>
<td></td>
<td>LMNOPQ</td>
<td>LMNOPQ</td>
<td>LMNOPQ</td>
</tr>
<tr>
<td></td>
<td>RSTUVWXYZ</td>
<td>RSTUVWXYZ</td>
<td>RSTUVWXYZ</td>
</tr>
<tr>
<td>8</td>
<td>AB</td>
<td>CD</td>
<td>EFG</td>
</tr>
<tr>
<td></td>
<td>HIJK</td>
<td>LM</td>
<td>HIJK</td>
</tr>
<tr>
<td></td>
<td>NOPQ</td>
<td>RS</td>
<td>NOPQ</td>
</tr>
<tr>
<td></td>
<td>TUVWXYZ</td>
<td>TUVWXYZ</td>
<td>TUVWXYZ</td>
</tr>
<tr>
<td>9</td>
<td>AB</td>
<td>CD</td>
<td>EFG</td>
</tr>
<tr>
<td></td>
<td>HIJK</td>
<td>LM</td>
<td>HIJK</td>
</tr>
<tr>
<td></td>
<td>NOPQ</td>
<td>RS</td>
<td>NOPQ</td>
</tr>
<tr>
<td></td>
<td>TUVWXYZ</td>
<td>TUVWXYZ</td>
<td>TUVWXYZ</td>
</tr>
<tr>
<td>12</td>
<td>AB</td>
<td>CD</td>
<td>EFG</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>IJK</td>
<td>LM</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>OPQ</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>S</td>
<td>TUVWXYZ</td>
<td>TUVWXYZ</td>
</tr>
</tbody>
</table>

Table 3.8: Selected optimal constrained keypad designs (4, 8, 9 and 12 keys) for the Multi-Tap text entry method for minimum KSPC values by word list
3.6 Discussion of Keypad Design Results

The DA values from Table 3.1 for predictive disambiguation text entry method for each of the three word lists are plotted against the number of keypad keys in Figure 3.6. Predictive disambiguation performance is poor for small designs, but increases rapidly as the number of keys increases and then begins to level off. For keypad sizes up to nine keys, the performance level for the written word list is always greater than or equal to the spoken list. For ten keys and above, the performance level for the spoken word list becomes higher than that of the written word list. Performance with the SMS word list is consistently lower than that obtained with the other two lists, and the rate of increase in performance is also lower. Very good performance (DA over 95%) was achieved with designs of seven keys for the written and spoken word lists, but not until ten keys with the SMS list. Good results (DA over 90%) were also obtained for designs with five or six keys with the written and spoken word lists, suggesting that using fewer keys (i.e., smaller keypads) may be feasible on devices were space is at a premium. But this result is tempered by results with the SMS word list, which requires seven keys to approach the same performance level.

Table 3.2 shows four selected optimal constrained keypad designs. It can be seen from the table that designs of the same number of keys are different for each of the three word lists, except at twelve keys, where the written and spoken list designs are the same.

A similar trend in performance is also found with the KSPC values for constrained keypad designs for both the predictive disambiguation method (Table 3.3) and Multi-Tap method (Table 3.7). For up to ten keys, optimum predicted performance with the written word list is greater than with the spoken word list, which in turn is greater than with the SMS word list. A comparison of Tables 3.2 and 3.4 shows that for all three word lists, the two metrics provide different optimal keypad designs for a given number of keys. This suggests that optimal keypad designs might be task or condition specific, and that the choice of keypad design may depend on the type of performance that is desired. That is, whether it is more desirable to minimize the number of keystrokes, or maximize the number of correct word predictions.
Similarly, a comparison of Table 3.4 and Table 3.8 demonstrates that the optimal keypad designs for predictive disambiguation method and the Multi-Tap method are different, which suggests that optimal keypad designs might also depend on text entry methods, even with the same number of keys, optimization metric, and word list.

Interestingly, by comparing Tables 3.3 and 3.7, we can see that small keypad of only one key has much higher KSPC values for the predictive disambiguation method than for the Multi-Tap method. However, with the increasing number of available keys (two or more keys), the KSPC performance of the predictive disambiguation method quickly exceeds that of the Multi-Tap method. This is not unexpected considering the fact that KSPC performance of predictive disambiguation method depends on the average length of the matching candidate words, while the KSPC of Multi-Tap method depends on the average number of characters that are placed on a single key. Lengths of the matching candidate lists should grow extremely quickly with only one key available.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Disambiguation Accuracy (DA)</td>
<td>95.41%</td>
<td>97.11%</td>
<td>96.63%</td>
<td>93.52%</td>
</tr>
<tr>
<td>Comparison Design</td>
<td>8-Key Constrained</td>
<td>9-Key Constrained</td>
<td>9-Key Constrained</td>
<td>9-Key Constrained</td>
</tr>
<tr>
<td>Comparison DA</td>
<td>96.69%</td>
<td>97.72%</td>
<td>97.72%</td>
<td>97.72%</td>
</tr>
<tr>
<td>Improvement</td>
<td>1.34%</td>
<td>0.61%</td>
<td>1.09%</td>
<td>4.20%</td>
</tr>
</tbody>
</table>

Table 3.9: Comparison of constrained keypad designs with other keypad designs (spoken wordlist)
Finally, comparing Table 3.5 with Table 3.3 shows that for predictive disambiguation methods, maximum DA values for unconstrained designs consistently meet or exceed the DA values of equivalent constrained designs across the three word lists. But the optimal unconstrained keypad designs (Table 3.4) are radically different than their constrained counterparts (Table 3.6).

Table 3.9 shows a performance comparison (using the spoken word list) between our optimal constrained keypad designs and three unconstrained designs proposed in the literature [47, 21, 2] for predictive disambiguation methods. A comparison is also made to the standard North American telephone keypad, which is a constrained design [31]. Size-equivalent (i.e., same number of keys) constrained designs from our research have performance advantages of between 0.61% and 4.20% over all four other designs. In addition, the constrained designs should offer greater ease of learning and usability for novices. Also, the constrained eight-key design is similar enough to the North American design such that user effort to change over to the new design would be minimal compared to any of the unconstrained designs.

An optimal keypad design was also proposed in the study that collected the SMS corpus used in our research [36]. The optimal constrained keypad design for predictive disambiguation methods using the SMS word list from our study is compared to the eight-key unconstrained design from [36] in Table 3.10. The difference in DA values is 1.76% in favor of the SMS keypad design, but the difficulty in learning the unconstrained keypad design may negate the performance advantages that can be gained.

<table>
<thead>
<tr>
<th>Keypad</th>
<th>SMS</th>
<th>8-Key Constrained</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 EGP</td>
<td>1 ABCD</td>
</tr>
<tr>
<td></td>
<td>2 CDR</td>
<td>2 EFG</td>
</tr>
<tr>
<td>4 HIVZ</td>
<td>3 KMO</td>
<td>4 HIJ</td>
</tr>
<tr>
<td>5 LNQX</td>
<td>6 KMO</td>
<td>5 KLM</td>
</tr>
<tr>
<td>7 BSY</td>
<td>8 AFW</td>
<td>7 RS</td>
</tr>
<tr>
<td>8 JTU</td>
<td>9 UVWXYZ</td>
<td>8 T</td>
</tr>
<tr>
<td>9 AFW</td>
<td>10 UVWXYZ</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 3.10: Comparison to optimal SMS keypad design
In addition to providing support for constrained keypad designs, our research has shown the effects of different word lists on keypad design and performance. If a design is optimized on an inappropriate corpus, performance will suffer. Table 3.11 shows DA values with the SMS word list for keypad designs originally created (optimized) using the written or spoken word lists for predictive disambiguation methods. The values are consistently much lower than the DA values achieved when the keypad is designed using the SMS word list.

The effect of the word lists is also shown by the differences in optimal keyboard design for predictive disambiguation methods (not just DA value) for different size keypads. Except for the one, two, and twelve key designs, at least one of the constrained optimal designs is different from the other two for a given keypad size. For half of the keypad sizes, all three constrained optimal designs are different.

The size of a word list can factor into how well a keypad design can be optimized in terms of DA and KSPC. The more words on a list, the harder it is to find key combinations that match only one word from the list. The SMS word list is much larger than the spoken and written word lists, so it seems reasonable that final DA and KSPC values for given keypad sizes might be lower than for the other word lists.

Another factor that can affect keypad performance is the distribution of word sizes. Table 3.12 shows the percentage of words of a given word length in each word list. Figure 3.7 summarizes this same data graphically. The SMS list distribution is skewed leftward more than the written and spoken word list distributions. This is not surprising since it is expected that users would tend to favor shorter words for text messaging due to text entry difficulty and limitations on message size. But placing increased emphasis on shorter words, especially with larger word lists, makes it more difficult to find well-performing solutions. Distribution of word sizes may also be the reason that performance with the spoken word list becomes better than written list performance for designs of ten to twelve keys.
<table>
<thead>
<tr>
<th>Constrained keypad designs optimized on written word list</th>
<th>ABCDEF</th>
<th>ABCD</th>
<th>EFGH</th>
<th>ABCD</th>
<th>EF</th>
<th>AB</th>
<th>CD</th>
<th>EF</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHIJKLMN</td>
<td>IJKL</td>
<td>MN</td>
<td>GH</td>
<td>IJKL</td>
<td>GH</td>
<td>IJKL</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>OPQRS</td>
<td>OPQR</td>
<td>S</td>
<td>MN</td>
<td>OPQR</td>
<td>S</td>
<td>O</td>
<td>PQR</td>
<td></td>
</tr>
<tr>
<td>TUVWXYZ</td>
<td>T</td>
<td>UVWXYZ</td>
<td>T</td>
<td>UVWXYZ</td>
<td>S</td>
<td>TUV</td>
<td>WXYZ</td>
<td></td>
</tr>
<tr>
<td>DA value if SMS word list used</td>
<td>67.46%</td>
<td>91.18%</td>
<td>93.01%</td>
<td>96.54%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constrained keypad designs optimized on spoken word list</th>
<th>ABCDEF</th>
<th>ABCD</th>
<th>EFG</th>
<th>ABC</th>
<th>DEF</th>
<th>AB</th>
<th>CD</th>
<th>EF</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHIJKL</td>
<td>HIJKL</td>
<td>MN</td>
<td>GH</td>
<td>IJKL</td>
<td>GH</td>
<td>IJKL</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>MNOPQ</td>
<td>O</td>
<td>PQRS</td>
<td>MN</td>
<td>O</td>
<td>PQRS</td>
<td>O</td>
<td>PQR</td>
<td></td>
</tr>
<tr>
<td>RSTUWVXYZ</td>
<td>TUV</td>
<td>WXYZ</td>
<td>TUV</td>
<td>WXYZ</td>
<td>S</td>
<td>TUV</td>
<td>WXYZ</td>
<td></td>
</tr>
<tr>
<td>DA value if SMS word list used</td>
<td>67.50%</td>
<td>91.07%</td>
<td>92.77%</td>
<td>96.54%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| DA value of keypad design optimized on SMS word list    | 68.71% | 92.49% | 94.02% | 96.85% |

<table>
<thead>
<tr>
<th>Increase in DA over:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>written list optimization</td>
<td>1.85%</td>
<td>1.44%</td>
<td>1.09%</td>
<td>0.32%</td>
</tr>
<tr>
<td>spoken list optimization</td>
<td>1.79%</td>
<td>1.56%</td>
<td>1.35%</td>
<td>0.32%</td>
</tr>
</tbody>
</table>

Table 3.11: Performance comparison of keypad designs optimized on one word list, but used with another
<table>
<thead>
<tr>
<th>Word size (characters)</th>
<th>Percentage of Total Word List</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Written</td>
</tr>
<tr>
<td>2</td>
<td>1.95%</td>
</tr>
<tr>
<td>3</td>
<td>4.41%</td>
</tr>
<tr>
<td>4</td>
<td>13.46%</td>
</tr>
<tr>
<td>5</td>
<td>15.54%</td>
</tr>
<tr>
<td>6</td>
<td>16.02%</td>
</tr>
<tr>
<td>7</td>
<td>15.18%</td>
</tr>
<tr>
<td>8</td>
<td>11.88%</td>
</tr>
<tr>
<td>9</td>
<td>8.46%</td>
</tr>
<tr>
<td>10</td>
<td>5.95%</td>
</tr>
<tr>
<td>11</td>
<td>3.19%</td>
</tr>
<tr>
<td>12</td>
<td>1.76%</td>
</tr>
<tr>
<td>Other</td>
<td>2.19%</td>
</tr>
</tbody>
</table>

Table 3.12: Number of words of different lengths in different word lists

Figure 3.7: Total percentage of words of various lengths for each of the three word lists
3.7 Experiments to Empirically Test Keypad Designs

To investigate the hypothesis that the constrained keypad designs are easier for novices to learn and use with the predictive disambiguation method than their equivalent unconstrained designs, we performed two usability experiments.

3.7.1 First Experiment

The first experiment compared the text entry performances of the optimal 8-key constrained keypad design, the near optimal 8-key unconstrained keypad design, and the 8-key standard telephone keypad design. The learning curve when using the optimal 8-key constrained keypad design was also examined.

3.7.1.1 Participants

Eight computer science graduate and undergraduate students voluntarily participated in the experiment. Six of the subjects were male, and two were female. The median age of the subjects was 29 years, and all but one carried a mobile phone. The other seven had carried mobile phones for 3.7 years on average, and three stated that they used the mobile phone daily. Subjects primarily used their mobile phones for voice calls, checking the time, and for reminders or alarms. Among the seven subjects who carried mobile phones, five of them reported occasional use of text messages, and the other two said they never received or sent text messages. Since none of the subjects used the phones for text messages on a regular basis, they were all categorized as novices for the purpose of this study.

3.7.1.2 Apparatus

The optimal eight-key constrained and unconstrained keypad designs for predictive disambiguation methods found using the spoken word list were used to create two text entry interfaces using virtual keys. These interfaces were implemented on a Pocket PC using the .NET Compact Framework environment. The interface with the constrained keypad design is shown in Figure 3.8.
Figure 3.8: Constrained keypad design testing interface

The interface was divided into two parts. The top section was a text box, used to display the text that is entered. The bottom section was a twelve-button keypad. A stylus was used to “press” each key. Eight keys were assigned letters. The four remaining keys were assigned functions to create an interface that allows multiple word entry and basic editing as follows:

- “↵” key: This “backspace” key is used to delete the last character displayed.
- “Done” key: This is pressed after a sentence has been entered. The text box is cleared for the next sentence.
- “Next” key: The “next” key is used to cycle through all the word candidates resulting from an ambiguous key press sequence.
- “→” key: This key adds a space and allows the next word to be entered.

3.7.1.3 Procedure

Two testing sessions were conducted. For the first session, each subject first filled out a questionnaire asking for background information. The subject was then given either the constrained or unconstrained interface design (randomly chosen). An explanation on how to enter a sentence was given, and the subject
entered two sample sentences for practice. After this training, each subject performed four task sessions. During each task session, the subject entered one of four sets of six phrases. Each subject entered all four sets over the course of the testing session, but the order of the sets was randomly chosen. A sample set of sentences is shown in Table 3.13. Sentences were chosen from those provided in [57]. The sets were balanced to provide the same number of characters in each set, and the same number of words that require an ambiguous keystroke sequence (and therefore use of the Next key). For each task session, subjects alternated between the constrained and unconstrained keypad designs, starting with the one they trained on, therefore using each interface twice. Opinions about the interfaces were solicited at the end of the testing session. Subjects were told to work as quickly and accurately as possible, and to correct any errors that they noticed.

<table>
<thead>
<tr>
<th>an efficient way to heat a house</th>
</tr>
</thead>
<tbody>
<tr>
<td>watch out for low flying object</td>
</tr>
<tr>
<td>pay off a mortgage for a house</td>
</tr>
<tr>
<td>express delivery is very fast</td>
</tr>
<tr>
<td>life is but a dream</td>
</tr>
<tr>
<td>have a good weekend</td>
</tr>
</tbody>
</table>

Table 3.13: Sample set of phrases

For the second session, the same subjects returned one week later to perform a similar set of tasks, although a questionnaire and training session were not administered. In addition, instead of using the unconstrained design, the constrained design was tested against the North American standard telephone keypad design. Subjects used the constrained interface for inputting the same two sentence sets they entered in session one. The remaining sets of sentences were used with the telephone keypad design.
3.7.1.4 Results of The First Experiment

Non-parametric (Wilcoxon) tests were used to see if there were differences in performance for the two keypad designs. For the first testing session, Table 3.14 shows the average input speed (in words per minute (WPM)) for the constrained and unconstrained designs, along with the average number of errors per sentence. Errors are defined as each use of the backspace key to correct a character (no other types of errors were recorded). The significance of the difference between the two values is reported in the last column. This analysis shows that average entry speed using the constrained keypad design is significantly faster than that with the unconstrained design, but the error rate is not significantly different for the two interfaces.

For the second session, Table 3.15 shows the average input speed for the constrained and standard telephone keypad designs, along with the average number of errors made per sentence. This analysis shows that average entry speed using the constrained keypad design is not significantly different than that with the standard telephone design, and the error rate is not significantly different either.

Since the same sentences were input by the same subjects using the constrained keypad design in both sessions one and two, learning effects can be analyzed. Table 3.16 shows the average text entry speed for the constrained keypad designs in both sessions, along with the average number of errors per sentence. This analysis shows that average entry speed in session two is significantly higher than that in session one, and the error rate is not significantly different between sessions.

3.7.1.5 Discussion of First Experiment Results

While this experiment is small, and only looks at three designs from one keypad size, its results generally support the hypothesis that novice user testing and usability is greater for constrained keypad designs versus unconstrained designs. From Table 3.14, we see that novice users were able to use the constrained eight-key keypad design more effectively than an equivalent unconstrained design. Text input speeds are higher without any difference in error rate.
<table>
<thead>
<tr>
<th></th>
<th>Constrained</th>
<th>Unconstrained</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Text Entry Speed (WPM)</td>
<td>6.74</td>
<td>5.50</td>
<td>0.000</td>
</tr>
<tr>
<td>Avg. Error Rate (Errors/Sentence)</td>
<td>0.39</td>
<td>0.31</td>
<td>0.344</td>
</tr>
</tbody>
</table>

Table 3.14: Comparison of entry speeds and error rates for constrained and unconstrained keypad designs

<table>
<thead>
<tr>
<th></th>
<th>Constrained</th>
<th>Cell phone</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Text Entry Speed (WPM)</td>
<td>7.89</td>
<td>8.22</td>
<td>0.169</td>
</tr>
<tr>
<td>Avg. Error Rate (Errors/Sentence)</td>
<td>0.47</td>
<td>0.42</td>
<td>0.777</td>
</tr>
</tbody>
</table>

Table 3.15: Comparison of entry speeds and errors for constrained and standard cell phone keypad designs

<table>
<thead>
<tr>
<th></th>
<th>Session 1</th>
<th>Session 2</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Text Entry Speed (WPM)</td>
<td>6.74</td>
<td>7.89</td>
<td>0.000</td>
</tr>
<tr>
<td>Avg. Error Rate (Errors/Sentence)</td>
<td>0.39</td>
<td>0.47</td>
<td>0.457</td>
</tr>
</tbody>
</table>

Table 3.16: Comparison of entry speeds and errors across two testing sessions with the constrained keypad design
Table 3.15 shows that while the average input speed with the standard telephone keypad is slightly higher than with the equivalent constrained design, the difference is not significant (by convention, the level of statistical significance is often set to the value of 0.05). There is no significant performance difference in terms of error rate between the two designs as well. Table 3.16 suggests that subjects can effectively learn and use constrained keypad techniques over time, especially since they were not provided with any additional training at the beginning of the second session.

3.7.2 Second Experiment

One drawback of the first usability experiment [25] was that the design and testing of predictive keypads with predictive disambiguation methods used only lower-case letters. While the use of lower-case letters is consistent with many text entry studies, many researchers are of the opinion that testing should also include upper-case letters, numbers, and punctuation marks to more accurately reflect real-world text creation. Isokoski [37] specifically added upper-case letters and punctuation marks to the MacKenzie and Soukoreff phrase set [57] when testing a new soft keyboard design. MacKenzie and Soukoreff [56] have also pointed out that existing corpora fail to accommodate certain distinctions, such as differences between lower-case and upper-case characters. Zhai, Hunter, and Smith [84] have recently raised similar concerns about punctuation.

In order to address this concern, we extended the first experiment and tested modified versions of the alphabetically constrained and unconstrained keypad designs used in that study with a slightly modified phrase set that includes lowercase and uppercase letters, numbers, and punctuation marks [27].

An experiment was performed to compare text entry performance of the alphabetically constrained and unconstrained keypad designs with predictive disambiguation methods from the first experiment [25] with the additional capitalization, number, and punctuation function. Our hypothesis was that the constrained keypad design would outperform the unconstrained design even with these new characters.
3.7.2.1 Participants

Twenty-six graduate students (17 male, 9 female) with an average age of 27.6 years, voluntarily participated in the experiment. Subjects were all categorized as novice users for the purposes of this study, and were compensated $10 for participating.

3.7.2.2 Apparatus

In order to stay as consistent as possible with the previously tested 12-key keypad designs used in the first experiment, we introduced the extra functionality required for capitalization, numbers, and punctuation without adding additional keys. In order to do this, however, the following four simplifying assumptions were made:

1. Only one space is allowed between two words
2. All sentences begin with a capital letter, and end with a punctuation mark
3. Letters and numbers can not be mixed
4. Capital letters can only appear as the first letter of a word

The first and second assumptions seem reasonable for most situations. The third assumption would realistically cause some problems over long-term use. The fourth assumption does not allow for acronyms. However, these assumptions allowed the design of a simpler and more intuitive interface for the purposes of this testing. The assumptions can be removed without adding more keys, but the interface would become more complex, and this is left for future research.

Under these three assumptions, a dictionary-based disambiguation text entry system was implemented on a Compaq iPAQ Pocket PC using .NET Compact Framework. A stylus was used to press the virtual keys. Each interface consisted of a text box for showing the words entered and a simulated 12-button keypad. Eight of the twelve buttons were used for characters. The other four were functional keys that enabled the users to accomplish basic text editing and multiple word entry. For a new sentence, the keypad starts with all uppercase letters (based on assumption two). After choosing the first letter, the keypad changes to
lowercase letters (assumption four). Remaining text is entered using the character keys and the following function keys:

- “Done” key: When a sentence is complete, this key is pressed and the keypad will show three punctuation marks (period, question mark, and exclamation point). After one of these marks is picked, the keypad changes back to lowercase letters to give the user an opportunity to make corrections or changes. The user presses the “Done” key again to clear the text box and begin another sentence.
- “←” key: This is an editing key used to delete the last input character.
- “Next” key: This key is used to cycle through all the possible matching words if a key sequence is ambiguous.
- “→” key: This key is used to commit the current word, add a space, and start the next word (assumption one). At this point, the “→” key changes to a “shift” key which the user may use to cycle through uppercase letter, lowercase letter, and numerical character sets on the keypad. If a user chooses a capital letter when starting a new word, the keypad will then change back to lowercase letter (assumption four). If a user starts a new word with a number, the number must be committed before text is available again (assumption three).

An illustration of various testing interfaces is shown in Figures 3.9 through 3.12. Note that the bottom right-most key can be either “→” or “Shift” depending on mode. Our letter keypad designs were the same as those proposed by the research presented in the first experiment [25]. The numeric keypad follows the international convention, and the punctuation keypad is simply a set of three common punctuation marks.
3.7.2.3 Procedure

Half the subjects (randomly chosen) used the constrained keypad design for predictive disambiguation methods, while the other half used the unconstrained design. Subjects were given questionnaires before the first testing session asking for background information. Subjects also received training on the interface they used, and were allowed to enter two practice phrases. Subjects were then asked to complete two sessions, with a ten-minute break between each one. In each session, subjects were asked to input a set of twenty testing phrases. Each phrase contained lowercase letters and a random number of uppercase letters and/or numbers. Punctuation was always present at the end of each phrase. After the last data entry sessions, subjects were asked for their opinions on the keypad design that they used. Specifically, how did
they perceive the tested keypad design when compared with the standard telephone keypad design? Subjects were also asked how they felt about the text entry speed they just achieved, compared with their initial expectations.

We created testing sentences by modifying the MacKenzie and Soukoreff phrase set [57]. Uppercase letters, numbers, and punctuation marks were added where they seemed to be appropriate, similar to Isokoski [37]. Sentences were carefully chosen so that each set contained the same number of ambiguous words for each keypad design. Sample testing phrases are shown in Table 3.17. Subjects were told to work as quickly and accurately as possible, and to correct any errors that they noticed. Time was recorded for each sentence entered together with error rate, which consists of two types of errors (corrected and uncorrected). We counted each block of consecutive presses of the “backspace” key as a single corrected error, and each wrong word in the final input phrase as one uncorrected error.

| This electric car has 12 big fuel cells! |
| Delivery in 15 minutes is very fast! |
| Play it again Sam! |
| Did you have a good time? |
| The king sends you to the tower. |

Table 3.17: Example testing sentences

3.7.2.4 Results of the Second Experiment

The average text entry speeds and error rates for both keypad designs can be found in Table 3.18. A paired T-test was conducted to find the significance level between the pairs of text entry speeds and error rates. Subjects performed significantly faster with the constrained keypad design. They also made slightly more errors, but the difference is not significant. The total number of uncorrected errors was small compared to the number of corrected errors, so the errors are combined for the purposes of analysis.
Table 3.18: Comparison of speeds and error rates between keypad designs

<table>
<thead>
<tr>
<th></th>
<th>Constrained</th>
<th>Unconstrained</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Text Entry Speed (WPM)</td>
<td>7.20</td>
<td>6.60</td>
<td>&lt; 0.05</td>
</tr>
<tr>
<td>Avg. Error Rate (Errors/Sentence)</td>
<td>0.74</td>
<td>0.60</td>
<td>0.078</td>
</tr>
</tbody>
</table>

Figure 3.13 shows learning effects for text entry speeds and error rates for users of both keypad designs by plotting average performance during each of four “periods” over time. Every ten phrases are grouped into a period, starting with the first ten phrases from session one being grouped into period one. (Since the difficulty for entering each single testing phrase varies, a plot of average entry speeds over individual phrases does not clearly show the learning effect. Therefore phrases are grouped and “period” is used as the unit instead.)

Overall, subjects who used the constrained keypad design perceived it as being a closer match to the standard telephone keypad than those who used the unconstrained design. Subjects who used the
constrained design also felt that they achieved performance closer to their initial expectations than those using the unconstrained design.

3.7.2.5 Discussion of Second Experiment Results

The text entry speed achieved with predictive disambiguation methods by using the alphabetically constrained keypad design is significantly higher than that of the unconstrained design. In addition, while the overall error rate was higher for the constrained keypad design, the rate was not significantly different than that found for the unconstrained design. Overall, these results support our initial hypothesis. This was in light of the fact that the constrained design had a lower Disambiguation Accuracy than the unconstrained design, which means that theoretically the unconstrained keypad design should achieve higher entry speeds.

The higher error rate that resulted with the constrained keypad design may be related to the increased entry speed achieved on constrained keypad design. There is probably a tradeoff between entry speed and error rate that needs to be considered in any overall measure of performance, especially for novice users. This is something that most text entry performance models do not take into account.

From Figure 3.8, users seem to learn both keypad designs at same rates, although the average text entry speeds for the constrained design are always higher than those for the unconstrained design. A longer study with multiple testing sessions over days or weeks would give a better indication on how performance changes over long-term use, and would shed light on whether the advantages of the constrained design lessen or disappear as novice users become experts.

Overall, the constrained design was perceived as being a closer match to the current telephone keypad standard, and users felt better about their performance with it. The general user population might more readily accept a new constrained keypad design if it were introduced on a device such as a cell phone.
3.8 Conclusions and Future Work

This part of the dissertation work has contributed to knowledge about predictive disambiguation keypad text entry techniques in two ways. First, optimal keypads of various sizes were designed under the constraint of keeping letters in alphabetical order across keys, a problem which had not been studied previously. While analysis showed that performance tradeoffs exist between the designs, reasonable performance can be achieved by relatively small keypads, and performance is close to that found for size-equivalent constrained designs. Optimal keypad designs also differed for the two performance metrics used, suggesting that the best keypad design might depend on the type of task involved. User testing supported the hypothesis that novice user ease of learning and usability is greater for constrained designs than for unconstrained designs with both a simpler lower-case letter only interface and a more complex interface that allows the capitalization, numbers, and some punctuation (albeit under some simplifying assumptions).

Second, the study showed that optimal predictive disambiguation keypad designs are affected by different word lists. The study used three word lists from different corpora, including one created from SMS messages, and found that performance for the same size keypad can vary greatly based on the word list used. It was hypothesized that not only list size, but the distribution of word length in a list, could affect performance of keypad designs because keystroke ambiguities become more difficult to resolve. The analysis of different word lists and corpora for keypad design has not been addressed in previous studies.

The study also showed by creating optimal keypad designs for Multi-Tap text entry method that, different text entry methods use different keypad designs to achieve best performances. Keypad designs in this case were optimized using three word lists under KSPC metric using a dynamic programming algorithm.

Several secondary contributions of this study include 1) the development of a GA-based heuristic to find effective solutions (using reasonable computational effort) to the NP-complete problem of creating unconstrained keypad designs. The GA based heuristic also finds good solutions to the constrained keypad design problem. And 2) a dynamic programming algorithm to find the solutions to the problem of
creating optimal constrained keypad designs for Multi-Tap method under KSPC metric using various word lists.

While GA techniques have been applied to unconstrained keypad design in the past, they have not been applied to constrained keypad design. Also, dynamic programming technique has never been applied to the keypad design problem for Multi-Tap problem before.

Many “optimized” text entry methods are counterintuitive and radically different from what users are already familiar with. They can require great amounts of learning time to use effectively, and can frustrate users in the process. This study has investigated keypad design problems for predictive disambiguation text entry methods and Multi-Tap text entry method that can increase ease of learning, usability, and performance for novice users. The methods conform to the basic design principle of standardization, which is done through alphabetization of letters across keys. Techniques that build on familiarity and intuition should work well not only for novices, but also for mobile users who may be attending to other tasks while entering text.

In terms of future research, in order to better understand what specific aspects of word lists cause performance results to differ, it may work best to collect various corpora from different mobile environments and applications (such as text messaging, calendar functions, and information retrieval), which are lacking in current research studies. Work described in this section optimized mobile phone keypad design by rearranging characters under alphabetical constraints. However, there exist other character layouts that are familiar to many users, such as the QWERTY keyboard layout. Optimization of mobile phone keypad designs under different layout constraints may be interesting and practical future work. Another area for future research is the development and validation of user models to aid the understanding and prediction of user performance with dictionary based predictive disambiguation mobile text entry methods.
Chapter 4

Semantic Relatedness and Part-of-Speech Language Models for Predictive Disambiguation Methods

4.1 Introduction

In addition to the research on keypad character mapping that was discussed in Chapter 3, part of this dissertation work focused on modifying the predictive disambiguation method itself using various language models to increase text entry effectiveness on mobile devices.

This chapter is organized as follows. The problem of predictive disambiguation for text entry methods used with small keypads is described in Section 4.2. Next, Section 4.3 details a model that utilizes semantic relatedness between word pairs as a novel measurement for ordering candidate word lists produced by ambiguous keystrokes. Section 4.4 presents a similar syntactic model, which differs in that it uses part-of-speech information for ordering candidate word lists. Section 4.5 describes how these two new models were combined with a standard word frequency model to produce a “context-aware” predictive disambiguation method. This method uses three coefficients to linearly combine all of the models to produce candidate word lists. Section 4.6 describes the process developed to find coefficients that achieve the best predictive disambiguation performance. Since the size of the semantic relatedness model is a practical concern, a cut-off threshold method was used, as detailed in Section 4.7. Section 4.8 describes a computer simulation that was conducted to find the theoretical improvements in predictive disambiguation performance of the proposed context-aware predictive disambiguation method (from Section 4.5) and the results of the simulations are reported in Section 4.9. To empirically validate the context-aware predictive disambiguation method, a usability study was carried out and is presented in Section 4.10. Section 4.11 summarizes the results of the research presented in this chapter and provides avenues for possible future work.
4.2 Predictive Disambiguation Problems when Using Small Keypads

As introduced in Chapter 2, normal dictionary-based predictive disambiguation text entry methods only use word frequency information to help decide the order of a list of matching candidate words. Therefore, statistically, fewer keystrokes will be required for inputting more frequent words, which improves overall text entry performance.

Given the results of many existing research studies on the problem of word prediction and/or disambiguation [e.g., 32, 76, 60, 49], it is reasonable to hypothesize that even better predictive disambiguation accuracy for dictionary-based predictive disambiguation text entry methods might be achieved by utilizing previously entered text as contextual information. The highest payoffs in this problem are realized if performance improvements are found for keypad designs with very few keys by utilizing a contextually aware predictive disambiguation method [29].

Finding well-performing, usable text entry methods for keypads with very few keys remains a significant challenge for HCI researchers. Besides the 5-key watch-top interface from Dunlop [17] and the edge-write technique from Wobbrock [80], there are several other research studies related to this topic [18, 19, 4, 20]. A paper from Sandnes [67] gives a good overview of existing text entry methods used on very small devices with only a few keys (usually 3 to 4). Text entry on such devices is particularly difficult using techniques such as the dictionary-based predictive disambiguation method because of the increased ambiguity with fewer available physical keys. As stated in Section 2.3, this problem is becoming even more critical with the decreasing size of mobile devices. There are also potential applications of the predictive disambiguation text entry method on certain devices with few keys (or buttons) for users with special needs (e.g., those with motor or visual impairments). Examples of existing small devices that have very few keys or limited available interaction space are shown in Figure 4.1.

While the potential benefits of context-aware predictive disambiguation methods are great, practical concerns include the data storage and processing limitations of current mobile devices that any predictive
disambiguation language model needs to work within. To address such concerns, efficient ways of storing data required by the methods were studied.

Based on the results of previous research [46], better disambiguation performance can be achieved if a higher ordered N-Gram or semantic information model can be utilized, in addition to the word frequency information used by normal predictive disambiguation text entry methods to order candidate word lists. This led us to develop a predictive disambiguation method that uses semantic relatedness (a measurement of semantic distance between pairs of words) as a new measurement for ordering a list of matching candidate words.

4.3.1 Statistical Semantic Relatedness of Word Pairs

N-Gram models have been used by various text entry or natural language processing systems to capture the relationships between different words [16, 11, 59]. However, such models may not be suitable for applications to mobile devices because of storage and computational power limitations [69]. For example,
in sentence “The dog was really sick and barked all night”, the word “dog” clearly would serve as a good contextual word for disambiguating the word “barked”. But in order to capture this relationship, a word level N-Gram model length of at least six would be necessary, which exceeds the hardware limitations of most existing mobile devices.

To solve this problem, a co-occurrence-based semantic relatedness model is proposed here to replace the more limited word N-Gram model. The model is defined, by modifying a similar semantic relatedness model proposed by Li and Hirst [49], to have the following form:

\[
SEM(w_1 \mid w_2) = \frac{C(\text{Stem}(w_1), \text{Stem}(w_2))}{C(\text{Stem}(w_2))}
\]

Where \( w_1 \) and \( w_2 \) are any two words in the dictionary. \( \text{Stem}(w_1) \) and \( \text{Stem}(w_2) \) are the word stems of \( w_1 \) and \( w_2 \). A word stem is the combination of the basic form of a word plus any derivational morphemes (e.g., those prefixes and suffixes that change the meaning and/or part-of-speech of a word, such as “un” or “less”), but excluding inflectional elements (e.g., suffixes that only create a change in the function of a word, such as “s” and “ly”). Word stems used here are derived by a stemming algorithm [63], therefore may not be exact. \( C(\text{Stem}(w_2)) \) is the number of times that the stem of \( w_2 \) occurs in the training corpus. \( C(\text{Stem}(w_1), \text{Stem}(w_2)) \) is the number of times the stems of both words \( w_1 \) and \( w_2 \) occur in the same defined contexts in the training corpus. For this dissertation work, two words in the same sentence are defined as being in the same context. The choice of the sentence as a unit of context is based on the results from two experiments using the sentence and paragraph as two forms of contexts. Comparing the two experiments, using paragraph as a unit of context required much more computation for training the semantic relatedness model, and surprisingly decreased the predictive disambiguation performance. Therefore, sentence is clearly the better choice for this dissertation work. Finally, \( SEM(w_1 \mid w_2) \) denotes the co-occurrence based semantic relatedness of word \( w_1 \) after seeing word \( w_2 \) in the context.
Note that the semantic relatedness model defined by Li and Hirst [49] is symmetric, which means that $SEM(w_1 \mid w_2)$ should always equal $SEM(w_2 \mid w_1)$, while this dissertation work is by definition conditional-probability-based, and therefore asymmetric.

Consider the following example: Suppose word $w_1$ and $w_2$ appear for 100 and 20 times in a training corpus respectively. They also appear together in the same sentences 19 times. Now, suppose word $w_2$ has already been seen in the context. What is the probability of an occurrence of the word $w_1$ later in the same context? The answer is $19/20 = 95\%$. However, if the other way around, the probability of seeing an occurrence of the word $w_2$ in the same context after seeing $w_1$ is only $19/100 = 19\%$, a much lower chance. Therefore, the relatedness between a pair of words depends on which word occurs first, and we modified our semantic relatedness model accordingly to reflect this observation.

Also note that the semantic relatedness model is built on word stems instead of words themselves, since what is really interesting is the relationship between meanings of words. This reduces the size of the model and to some extent helps the problem of sparse training data.

4.4 Syntactic Context-Aware (Part-of-Speech) Predictive Disambiguation Method

As discussed in Section 2.7, part-of-speech (POS) tagging is a well-studied natural language processing problem, and has many applications to various research fields. In terms of using POS information in the dictionary-based predictive disambiguation problem, it is expected that in the case where several words share the same keystroke sequence, the user-desired word should have a part-of-speech that is most compatible with those of the preceding words. Therefore, based on preceding word input, it is possible to calculate a part-of-speech validity for each matching word of the same keystroke sequence, which can be used as another type of measurement for ordering a list of candidate words.
4.4.1 Word Part-of-Speech Validity

Word POS validity in this dissertation research is calculated based on POS groupings instead of individual POS tags. POS groupings were used because it is necessary to reduce the number of POS tags used in the algorithm, and thereby minimize the computation needed by the predictive disambiguation method running as a real-time algorithm. The BNC corpus that will be used in the simulation described later uses two tagsets: the basic and the ambiguity tagsets, which are introduced in Appendices A and B. Closely related POS tags from these two tagsets are grouped into POS groupings that were used in our algorithm. For example, the Part-of-Speech “VBB”, “VBD”, “VBG”, “VBI”, “VBN”, and “VBZ” are different forms of the verb BE, therefore they are grouped into POS grouping number 12, which was then used by the POS model and the algorithm to calculate the word POS validity. Nineteen POS groupings are listed in Appendix C.

Using the same definitions as those in Section 2.9.2, the POS validity of a word based on the preceding input words can be derived by modifying the Bi-Gram Viterbi algorithm [59] as follows:

\[
POS(w_j \mid S = t_0w_1w_2\ldots w_{j-1})
\]

\[
= \max_{1 \leq j \leq n} \{ \delta_j(t_j) \}
\]

\[
= \max_{1 \leq j \leq n} \{ \max_{1 \leq k \leq n} \{ \delta_{j-1}(t_k) \times P(t_j \mid t_k) \times P(w_j \mid t_j) \} \}
\]

Where the definitions of \( w_j, S = t_0w_1w_2\ldots w_{j-1}, n, \delta_j(t_j), P(t_j \mid t_k), \) and \( P(w_j \mid t_j) \) are all the same as those defined in Section 2.9.2, except that the definition of \( t_i \in \{ t_1\ldots t_n \} \) now refers to one of the 19 POS groupings instead of individual POS tags. Furthermore, \( POS(w_j \mid S = t_0w_1w_2\ldots w_{j-1}) \) denotes the POS validity of a word \( w_j \) with the context of the preceding input sentence \( S = t_0w_1w_2\ldots w_{j-1} \).

The POS validity of a word \( w_i \) is explained as follows. Assume \( S = t_0w_1w_2\ldots w_{i-1} \) is entered, and \( w_i \) is one of the candidate words derived from the current keystroke sequence. According to the Viterbi algorithm, we calculate \( n \) paths from the sentence delimiter to each \( t_j \) of the \( n \) possible POS categories.
of word $w_j$. The scores of these $n$ paths are named $\delta_i(t_1)$ to $\delta_i(t_n)$, which represent the probabilities of the most likely tagging for the sentence $S = t_0 w_1 w_2 \ldots w_{j-1} w_j$ such that $w_j$ has the POS tags $t_1$ to $t_n$. However, for the dictionary-based predictive disambiguation problem, the primary concern is not which POS tag is the most likely for word $w_j$. Instead, we want to know, in the case of word $w_j$ being the user-desired word, the POS validity for $w_j$ in the $ith$ place in the current sentence. Therefore, where $w_j$ is the user desired word, the greatest probability among $\delta_i(t_1)$ to $\delta_i(t_n)$ should be chosen to represent the word POS validity for $w_j$. This is exactly the definition of the word POS validity above.

4.5 Context-Aware Disambiguation Method Combining Semantic Relatedness, POS, and Frequency Models

The work of predictive keystroke sequence disambiguation can be viewed as the decoding of specially encoded English text. Assume that there is an encoder that for every English character in the input text, outputs a number corresponding to the key that contains the character. Encoding English text is easy, while the decoding of coded keystroke sequences is not. In the following sections, a new predictive disambiguation algorithm is proposed. The objective is to find out which decoding is the most probable one based on the semantic and syntactic information of the sentence context already available. This method is referred to as the context-aware predictive disambiguation method in following sections.

Definitions:

$w_i$: A particular word in the dictionary.

$ks_i$: The keystroke encoding of $w_i$.

$match(ks_i) = \{w_1^w, w_2^w, \ldots, w_n^w\}$: The set of words that share the same keystroke sequence with $w_i$.

Note that $w_i \in match(ks_i)$.

$w_j^w \in match(ks_i)$: A word that shares the same keystroke encoding with $w_i$. 

65
\( Freq(w_j^w) \): The normalized frequency of \( w_j^w \) among all the words in the set of \( match(ks_i) \). Note that
\[
\sum_{w_j^w \in match(ks_i)} Freq(w_j^w) = 1.
\]
If \( w_i \) is not ambiguous, \( match(ks_i) \) should only have one element \( w_i^w = w_i \), therefore \( Freq(w_i^w) = 1 \).

Description of the Context-Aware Predictive Disambiguation Method

The inputs to the disambiguation method are the keystroke encoding \( ks_i \) of a word \( w_i \) and a sequence \( H_i \) of \( ks_i \)’s context words. The content of \( H_i \) depends on the arbitrarily defined context for a word. For example, \( H_i \) contains all the input words preceding \( w_i \) and in the same sentence in this dissertation work. Since the context is defined to be a sentence here, it is assumed that \( H_i \) only contains the words in the same sentence of \( w_i \). Note that only words already input and preceding the current word are used in \( H_i \).
Thus \( H_i \) is viewed as the history for word \( w_i \).

The output of the disambiguation method is an ordered list of English words \( w_1, w_2, \ldots, w_m \) that the disambiguation method thinks is the most probable list of decoding given \( ks_i \) and \( H_i \).

First, we define the validity of word \( w_i \) given its history \( H_i \) of context words as follows:

1. The estimated semantic validity \( SEMV(w_i \mid H_i) \) of a word \( w_i \) given \( H_i \) can be defined as:
\[
SEMV(w_i \mid H_i) = \sum_{w \in H_i} SEM(w_i, w')
\]
(4)

2. The normalized estimated semantic validity \( NSEMV(w_i \mid H_i) \) of a word \( w_i \) given \( H_i \) is defined as:
\[
NSEMV(w_i \mid H_i) = \frac{SEMV(w_i \mid H_i)}{\sum_{w_j^w \in match(ks_i)} SEMV(w_j^w \mid H_i)}
\]
(5)

3. The estimated POS validity \( POSV(w_i \mid H_i) \) of a word \( w_i \) given \( H_i \) is defined as:
\[
POSV(w_i \mid H_i) = POS(w_i \mid H_i)
\]
(6)
4. The normalized estimated POS validity \( NPOSV(w_i \mid H_i) \) of a word \( w_i \) given \( H_i \) is defined as:

\[
NPOSV(w_i \mid H_i) = \frac{POSV(w_i \mid H_i)}{\sum_{w_j \in match(ks_i)} POSV(w_j \mid H_i)}
\] 

Therefore, the estimated validity \( EV(w_i \mid H_i) \) of a word \( w_i \) given \( H_i \), can be defined as the linear combination of its normalized semantic validity, its normalized POS validity, and its normalized frequency:

\[
EV(w_i \mid H_i) = \alpha \times \text{Freq}(w_i) + \beta \times \text{NSEMV}(w_i \mid H_i) + \gamma \times NPOSV(w_i \mid H_i)
\] 

Where \( \alpha, \beta, \gamma \) (with values between 0 and 1) are coefficients specifying how much we would like to believe in the frequency, semantic, or POS validities.

Given the definitions above, the context-aware predictive disambiguation method becomes straightforward. It first finds the list of matching words of \( ks_i \), and then the estimated validity values of each matching word. It then sorts the list of words based on their estimated validity values and returns the sorted word list. This is formally stated as pseudocode in Figure 4.2:

```
Disambiguate (ks_i, H_i)

m_i = match(ks_i)

For each \( w_k \in m_i \), find \( EV(w_k \mid H_i) \)

Sort \( m_i \) based on \( EV(w_k \mid H_i) \), and return sorted \( m_i \)

End Disambiguate
```

Figure 4.2: Pseudocode for the Context Aware Predictive Disambiguation Method
A flow diagram of the disambiguation method is shown in Figure 4.3.

![Flow Diagram of the Context-Based Disambiguation Method](image)

**Figure 4.3: Flow Diagram of the Context-Based Disambiguation Method**

### 4.6 Finding the Best Coefficients for Linearly Combining Semantic Relatedness, POS, and N-Gram Validities

As an important step of the context-aware predictive disambiguation method described in the previous section, the frequency, semantic relatedness, and POS validities of each matching candidate word need to be linearly combined into an overall estimated validity, so that the whole candidate list can be appropriately sorted. This section describes the method by which we may find the optimal values for these linear combination coefficients $\alpha$, $\beta$, and $\gamma$.

#### 4.6.1 The Equivalent Geometry Problem

Using the same definitions as in Section 4.5, assume the current ambiguous keystroke sequence is $k_s$,

$match(k_s) = \{w_1, w_2, ..., w_l\}$

is the list of $l$ matching words, among which $w_k$ is the user desired word,
and \( \{Freq(w_i),...,Freq(w_j)\} \), \( \{SEMV(w_i),...,SEMV(w_j)\} \), and \( \{POSV(w_i),...,POSV(w_j)\} \) are the frequency, semantic relatedness, and POS validities for each of the matching words.

Let us first assume that we only need to optimize the linear combination coefficients for this one single keystroke sequence. Then we may pick any \( \alpha, \beta, \gamma \), such that for any \( 1 \leq i \leq l \) and \( i \neq k \) :

\[
\alpha \times Freq(w_i) + \beta \times SEMV(w_i) + \gamma \times POSV(w_i) > \alpha \times Freq(w_j) + \beta \times SEMV(w_j) + \gamma \times POSV(w_j)
\]  

(9)

Since the left-hand side of Equation (9) is \( EV(w_k \mid H) \) and the right-hand side of Equation (9) is \( EV(w_i \mid H) \) for all other \( w_i \) such that \( i \neq k \), we can guarantee that the overall estimated validity for this user desired word \( w_k \) is greater than those of the other matching words. Therefore \( w_k \) will be placed at the top of the returned candidate list.

Next, transform Equation (9) to Equation (10) below:

For any \( 1 \leq i \leq l \) and \( i \neq k \) :

\[
\alpha \times Freq(w_i) + \beta \times SEMV(w_i) + \gamma \times POSV(w_i) > \alpha \times Freq(w_j) + \beta \times SEMV(w_j) + \gamma \times POSV(w_j)
\]

\[\iff\]

\[
\alpha \times (Freq(w_i) - Freq(w_j)) + \beta \times (SEMV(w_i) - SEMV(w_j)) + \gamma \times (POSV(w_i) - POSV(w_j)) > 0
\]

(10)

If we think \((Freq(w_i) - Freq(w_j), SEMV(w_i) - SEMV(w_j), POSV(w_i) - POSV(w_j))\) as \( l-1 \) data points in a 3-dimensional space corresponding to the \( l-1 \) matching but undesired words, Equation (10) is simply the basic geometry equation saying that all the \( l-1 \) data points are in the positive plane of a surface determined by the origin and the normal vector \((\alpha, \beta, \gamma)\). On the other hand, if there is any data point that lands in the negative plane of the surface, the word that corresponds to this point will have an estimated overall validity greater than that of the user desired word, which will result in one extra keystroke before the user can cycle to the word that is really desired.

Going back to the original problem, for a testing corpus composed of large number of ambiguous keystroke sequences, each sequence is disambiguated to a list of matching candidate words, which
correspond to a list of data points in the 3D space. And now, think of putting all these data points in the space, the problem has become to find a surface \((\alpha, \beta, \gamma)\) that contains the origin, such that the greatest number of data points sit in the positive plane of this surface.

Figure 4.4: Surface for Optimization of the Linear Combination Coefficients

Figure 4.4 shows a hypothetic optimization process, where \(w_2\) is the user desired word, and \(w_1, w_3,\) and \(w_4\) are three conflicting words. The three data points in the figure correspond to each of the conflicting words, and are all “above” the optimized surface \(A\). The normal vector \((\alpha, \beta, \gamma)\) of the surface \(A\) gives the three coefficients needed.

4.6.2 Solving the Coefficient Optimization Problem

With the transformed equivalent problem, finding the best coefficients for linearly combining various estimated validities is much easier. As shown in Figure 4.4, the surface containing the origin can also be determined by specifying two angles \(\theta\) and \(\varphi\), such that:

\[
\begin{align*}
\alpha &= \cos(\theta)\sin(\varphi) \\
\beta &= \cos(\theta)\cos(\varphi) \\
\gamma &= \sin(\theta)
\end{align*}
\]  
(11)
Therefore, the optimization algorithm simply collects all of the data points and progressively varies the value of $\theta$ and $\varphi$ from 0 degree to 90 degree with a step of 1 degree. The best angles are those that determine a surface with the most number of data points in the positive plane. The parameters $\alpha$, $\beta$, and $\gamma$ are then calculated based on the best angles.

4.7 Reducing the Size of the Semantic Relatedness Models

Seymore and Rosenfeld mentioned in their publications [69, 70] that word sequences that occur the fewest number of times in a training text can be regarded as noise and lead to unreliable predictions. Therefore, it is necessary to apply a cutoff on a trained text model, where only the most frequently occurring information is included. This argument also applies to the semantic relatedness language model that is proposed in this dissertation work.

Besides the noise problem, there is also another practical issue that requires the reduction of the size of the resulting semantic relatedness model. Mobile devices usually come with relatively limited hardware resources, such as processing power and memory. Therefore a large semantic relatedness model could easily become problematic and of no practical use (for example, if it occupies too much memory or needs too much time to load).

The objective here is to see whether the semantic relatedness language model can accept a certain level of information loss, which could significantly reduce its size while still achieving reasonable disambiguation performance. This was done through iterative applications of different cutoff thresholds on the semantic model, and a measurement of any changes in disambiguation performances. The cutoff thresholds tested ranged from $0.1^0 = 1$ to $0.1^3 = 0.001$, with an incremental step of 0.2 in the exponent, plus the special case of $0.1^\infty = 0$. This range was determined by pretesting various threshold values. A semantic relatedness entry less than the threshold value was removed from the model.
4.8 Simulation

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

4.8.1 Corpus

The BNC Baby corpus was used in this simulation [7]. BNC-baby is a four million word sampling of the entire British National Corpus, and is the first version of the BNC to be distributed entirely in XML. The BNC Baby corpus consists of four one-million word samples, each compiled as an example of a particular genre: fiction, newspapers, academic writing, and spoken conversation. The BNC Baby corpus was chosen for simulation and testing purposes because of its XML formatting and because it is entirely POS tagged, which offered great convenience for building our semantic relatedness and POS language models.

4.8.2 Simulation Setup

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

The computer simulation examined the disambiguation performance when the semantic model alone, the POS model alone, or both the semantic and POS models together, were used in the context-aware predictive disambiguation method. For reporting purposes these are called the semantic context-aware method, the syntactic context-aware method, and the context-aware predictive disambiguation method, respectively. Simulations were also run for different sized keypad designs to examine theoretical performance differences based on the number of keys. The different keypad designs tested in the
simulations were the optimal alphabetically constrained keypad designs found by the previous research described in Chapter 3 [25].

Cutoff threshold effects on the semantic relatedness model were investigated by running a series of simulations using the model with the three-key optimal alphabetically constrained keypad design. Cutoff thresholds applied to the semantic relatedness model ranged from $0.1^0 = 1$ to $0.1^3 = 0.001$ with a constant step of $0.2$ in the exponent coefficient. The coefficient was constantly increased in the exponent because thresholds that are evenly distributed between 1 and 0.001 are not suitable for this experiment. Thresholds close to 1 should be altered by greater amounts, and thresholds close to 0.001 by smaller amounts. This was achieved by constantly increasing the exponential coefficient.

4.8.3 Simulation Results

Since for each run of the ten-fold cross validation simulation, nine tenths of the BNC Baby corpus was used for building the vocabulary and training various models, the resulting training and testing corpora contain about 3.6 million and 0.4 million English words respectively. To build a vocabulary of reasonable size, words that occurred fewer than 10 times were removed. Therefore, vocabularies of approximately 16170 word tokens were used in the ten-fold cross validation simulations.

As stated, the semantic relatedness model was built on word stems instead of words. The Porter stemming algorithm [63] was applied to the vocabulary to get a list of word stems. This resulted in ten lists of 9909 word stems on average. These stem lists were then used to build the semantic relatedness models.

Results of the computer simulations using the semantic context-aware, the syntactic context-aware, and the context-aware predictive disambiguation methods are presented below to show theoretical maximum text entry performance. Results are shown for both the theoretical DA and KSPC measurements.

The maximum predictive disambiguation performances (in DA and KSPC metrics) for different methods and various sizes of keypad designs are presented in Table 4.1 and plotted in Figures 4.5 and 4.6. Note that since the simulation used a 10-fold cross validation design, each data point presented in the table is
the average of the results from ten separate runs using the same set of parameters (number of keys and performance metric), but each has different training and testing sets.

The performance and size of the semantic relatedness model when various cutoff thresholds were applied are presented in Table 4.2 and Figures 4.7 and 4.8. As planned, the threshold was varied from 1 to 0.001 with a constant step of 0.2 in the exponent, plus threshold 0, which was derived from an exponent of $+\infty$. Because of the exclusion of many entries from the semantic relatedness model, a lot of candidate words will have the semantic validity of 0. Thus the semantic relatedness model was combined with the word frequency model to resolve such situations without additionally using the POS model.

After running two tailed paired t-tests, it was not surprising that for all keypad sizes, the performances of the context-aware, the semantic context-aware, and the syntactic context-aware predictive disambiguation methods are consistently and significantly better than the corresponding performances of the normal predictive disambiguation method using the same keypad sizes. In fact, the performances of the three context-aware methods always exceeded the performances of the normal method in each individual run of the ten-fold cross validation simulations. Therefore, compared to the performances of the normal method using the same keypad sizes, all of the data points of the three context-aware methods presented in Table 4.1 have statistical significances of less than 0.001. Since all the statistical significances are the same, they are simply reported here instead of in Table 4.1 to avoid unnecessary redundancy.

The size of the semantic relatedness model decreases as the applied cutoff threshold increases. Sizes of the semantic relatedness model are plotted versus various cutoff thresholds in Figure 4.9. The x-axis represents the exponent (ranging from 0 to 3 with a constant step of 0.2, plus $+\infty$) by which we derive these cutoff thresholds (ranging from 1 to 0.001 plus 0).
<table>
<thead>
<tr>
<th>Number of Character Keys</th>
<th>3-Key</th>
<th>4-Key</th>
<th>5-Key</th>
<th>6-Key</th>
<th>7-Key</th>
<th>8-Key</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keypad Design</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ABCDEFG</td>
<td>ABCDEF</td>
<td>ABCD</td>
<td>ABCD</td>
<td>EFG</td>
<td>ABCD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HIJKLNO</td>
<td>GHJKLM</td>
<td>EFGHIJ</td>
<td>HIJKL</td>
<td>HIJKL</td>
<td>HIJKL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PQRSTU</td>
<td>MNOPQ</td>
<td>KLMNO</td>
<td>PQRS</td>
<td>MNO</td>
<td>PQRS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/</td>
<td>RSTUVW</td>
<td>TUVWXY</td>
<td>TUVWXY</td>
<td>TUV</td>
<td>WXYZ</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Freq Only</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. DA</td>
<td>67.58%</td>
<td>79.12%</td>
<td>87.46%</td>
<td>91.91%</td>
<td>94.54%</td>
<td>95.98%</td>
</tr>
<tr>
<td>Avg. KSPC</td>
<td>1.2124</td>
<td>1.0942</td>
<td>1.0449</td>
<td>1.0253</td>
<td>1.0163</td>
<td>1.0118</td>
</tr>
<tr>
<td><strong>Freq + Sem Models</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. DA</td>
<td>68.51%</td>
<td>80.04%</td>
<td>88.06%</td>
<td>92.26%</td>
<td>94.79%</td>
<td>96.28%</td>
</tr>
<tr>
<td>Avg. KSPC</td>
<td>1.2027</td>
<td>1.0908</td>
<td>1.0418</td>
<td>1.0237</td>
<td>1.0153</td>
<td>1.0108</td>
</tr>
<tr>
<td><strong>Freq + POS Models</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. DA</td>
<td>71.22%</td>
<td>81.42%</td>
<td>89.26%</td>
<td>92.80%</td>
<td>95.15%</td>
<td>96.27%</td>
</tr>
<tr>
<td>Avg. KSPC</td>
<td>1.1856</td>
<td>1.0827</td>
<td>1.0378</td>
<td>1.0223</td>
<td>1.0144</td>
<td>1.0110</td>
</tr>
<tr>
<td><strong>Freq + Sem + POS Models</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. DA</td>
<td>71.82%</td>
<td>81.99%</td>
<td>89.80%</td>
<td>93.25%</td>
<td>95.39%</td>
<td>96.49%</td>
</tr>
<tr>
<td>Avg. KSPC</td>
<td>1.1789</td>
<td>1.0803</td>
<td>1.0358</td>
<td>1.0211</td>
<td>1.0136</td>
<td>1.0102</td>
</tr>
</tbody>
</table>

Table 4.1: Maximum predictive disambiguation performances for different predictive disambiguation methods and keypad sizes
Figure 4.5: Maximum disambiguation accuracies (DA) for various model combinations and different keypad sizes
Figure 4.6: Maximum Keystrokes Per Character (KSPC) for various model combinations and different keypad sizes.
Table 4.2: Predictive Disambiguation Performances When the Semantic Relatedness Model with Various Cutoff Thresholds Were Used for the Optimal Alphabetically Constrained Three-Key Keypad

<table>
<thead>
<tr>
<th>T</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
<th>0.8</th>
<th>1</th>
<th>1.2</th>
<th>1.4</th>
<th>1.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. DA</td>
<td>68.00%</td>
<td>68.26%</td>
<td>68.27%</td>
<td>68.01%</td>
<td>68.16%</td>
<td>68.30%</td>
<td>68.26%</td>
<td>68.01%</td>
</tr>
<tr>
<td>Avg. KSPC</td>
<td>1.2077</td>
<td>1.2083</td>
<td>1.2087</td>
<td>1.2079</td>
<td>1.2067</td>
<td>1.2058</td>
<td>1.2049</td>
<td>+∞</td>
</tr>
<tr>
<td>Avg. Model Size (KB)</td>
<td>201</td>
<td>554</td>
<td>937</td>
<td>1404</td>
<td>2978</td>
<td>4715</td>
<td>6650</td>
<td>9041</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T</th>
<th>1.8</th>
<th>2</th>
<th>2.2</th>
<th>2.4</th>
<th>2.6</th>
<th>2.8</th>
<th>3</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. DA</td>
<td>68.40%</td>
<td>68.40%</td>
<td>68.40%</td>
<td>68.43%</td>
<td>68.48%</td>
<td>68.52%</td>
<td>68.51%</td>
<td>68.52%</td>
</tr>
<tr>
<td>Avg. KSPC</td>
<td>1.2043</td>
<td>1.2037</td>
<td>1.2035</td>
<td>1.2033</td>
<td>1.2032</td>
<td>1.2031</td>
<td>1.2027</td>
<td>1.2031</td>
</tr>
<tr>
<td>Avg. Model Size (KB)</td>
<td>9041</td>
<td>11732</td>
<td>14471</td>
<td>17338</td>
<td>19905</td>
<td>22219</td>
<td>23927</td>
<td>27882</td>
</tr>
</tbody>
</table>
Figure 4.7: Maximum average DA values for frequency information and semantic relatedness models with various thresholds used on 3-key keypad design

Figure 4.8: Maximum average KSPC values for frequency information and semantic relatedness models with various thresholds used on 3-key keypad design
4.9 Discussion of the Simulation Results

By examining Figures 4.5 and 4.6, it is clear that both the semantic relatedness and POS models consistently helped the dictionary-based predictive disambiguation method to achieve better theoretical disambiguation performances in terms of both the DA and KSPC metrics. Performance improvements were gained with all of the keypads of different sizes (from three character keys to eight character keys). The most significant improvement was gained with the three-key keypad using the context-aware predictive disambiguation method. The DA value improved 4.24% from the original 67.58% to 71.82%, and the KSPC value improved 0.0335, dropping from 1.2124 to 1.1789. It should be noted that a 4.24% gain in the DA metric actually accounts for 6.3% of the original disambiguation performance, and the improvement of 0.0335 in the KSPC measurement accounts for almost 15.8% of all the extra keystrokes needed.

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

With the increase of keypad sizes, the DA and KSPC performances with the standard predictive disambiguation method increase rapidly, which leaves less room for improvements from the semantic relatedness and POS models. However, even with eight character keys, the semantic relatedness and POS models still managed to improve the DA and KSPC values by 0.21% and 0.0016 respectively.

By examining the performance improvements of the semantic context-aware or syntactic context-aware predictive disambiguation methods, it can be seen that the POS model actually achieves more improvements than the semantic relatedness model does for all of the keypad sizes. Considering the smaller size of the POS model (2KB for Bi-Grams of 19 part-of-speech categories) when compared with that of the semantic relatedness model (28MB with a dictionary size of about 17000), the POS model appears to be a more economical choice if available computing resources are limited.

It is also worth noting that the semantic relatedness and POS models seem to be independent of each other, and contribute to different aspects of the improvements to the predictive disambiguation performance. The corresponding performances of the normal predictive disambiguation method, when enhanced with both the semantic relatedness and POS models, are always better than performances when just the semantic relatedness model or the POS model is used. In addition, the total improvements are close to the sum of the two corresponding performance improvements when one of the two models is applied. Therefore, if resource limitations are not primary concerns, it should be reasonable to use both models to achieve maximum possible benefits.

From our observation of the simulations, we found that the training set of the size of four million words should be enough for saturating a POS model of 19 part-of-speech categories, but may not be enough for the semantic relatedness model of 17000 words. It is believed that the performance of the semantic relatedness model will continue to improve as more training data is used.
As shown in Table 4.2 and Figures 4.7 and 4.8, when cutoff thresholds are between $0.1^{+\infty} = 0$ and $0.1^2 = 0.01$, the performance of the semantic relatedness model decreases fairly slowly. Its performance starts to decrease rapidly only after the thresholds are greater than $0.1^2 = 0.01$. Figures 4.7 and 4.8 interestingly show that the performance has a local peak with the threshold of $0.1^{0.4} = 0.3981$. However, as shown in Table 4.2 and Figure 4.9, the size of the semantic relatedness model decreases linearly all along when various cutoff thresholds are applied. Therefore, for different applications, it is possible to use a threshold that achieves maximum possible performance and uses a practical amount of memory. (In the usability experiment described below, a cutoff threshold of 0.006 was used to derive a semantic relatedness model that fits the available memory on the Pocket PC that was used in the testing.)

4.10 Usability Experiment

To test the performance of the context-aware dictionary-based predictive disambiguation method on small mobile devices with very few keys in an empirical setting, usability testing was conducted using a PDA implementation of the method and a three-key keypad design.

4.10.1 Participants

32 students, 22 males and 10 females, with an average age of 26.1 years, voluntarily participated in the experiment. Thirty-one subjects were right-handed. Twenty-six subjects were graduate students and the rest were undergraduate students from Northeastern’s College of Computer & Information Science. Only 3 of these participants used dictionary-based predictive disambiguation text entry methods on a regular basis, therefore most participants can be categorized as novice users for the purposes of this study. However, all of them carried cell phones, for an average of 4.93 years. Most of the participants (25 out of 32) reported frequent or occasional usage (once a week or once a month) of text entry on their cell phones, and “Short Text Messaging” and “Phone Book” were the two most popular applications that used text entry. Participants were compensated $10 upon completion of the entire experiment.
4.10.2 Apparatus

The experiment studied the text entry performance using dictionary-based predictive disambiguation with a very small keypad design (three character keys and two function keys). The three-key design was chosen, because of the increasing popularity of small mobile devices that will only accommodate three character keys. Examples of such devices are shown in Figure 4.1. In addition, the greater keypad ambiguity that resulted from the limited number of keys also allowed for more potential improvement in terms of predictive disambiguation performance, as shown in Figures 4.5 and 4.6. A semantic relatedness model with a cutoff threshold of 0.006, which just fit into the PDA’s 32 MB of memory, along with the POS model, were used in implementing the predictive disambiguation method.

For testing purposes, the dictionary-based predictive disambiguation text entry system was implemented on a Dell Axim X30 Pocket PC using Embedded Visual C++ (EVC++) language. EVC++ was used instead of the .NET Compact Framework because it enabled the creation of a system that could run the context-aware method in real time with a dictionary of more than ten thousand words.

The participants were instructed to hold the PDA with one hand, and press virtual “keys” using the index finger of their other (dominant) hand. The text entry interface consisted of a text box for displaying input and a virtual 5-key keypad. Three of the five keys were character keys whose letter distribution follows our optimal alphabetically constrained keypad design previously found [25]. The other two keys were functional in nature and allowed cycling through all of the candidate words and basic editing. The four available functions (implemented on the two virtual keys) were:

- “Done” key: When a text phrase is complete, this key is pressed to clear the text box and begin another phrase.
- “Del” key: This is an editing function key to delete the last word input. Because of the limited number of keys, deletion was performed on a word basis and was only allowed in editing mode (when keystroke sequences were committed). Therefore, in cases where participants made incorrect character keystrokes, they needed to first commit the current incorrect word to leave the
predictive disambiguation mode and then delete the whole word by pressing “Del”.

- “Next” key: In predictive disambiguation mode, this key cycles through all possible matching words if a keystroke sequence is ambiguous.
- “___” (space) key: This key commits the currently disambiguated word. A space is also automatically inserted after the committed word.

In predictive disambiguation mode (when a sequence of key strokes is entered but not committed), the two available function keys were “Next” and “___”. After words were committed, they changed to “Del” and “Done” keys in editing mode, as illustrated in Figures 4.10 and 4.11.

Figure 4.10: Testing program using a 3-key constrained keypad design in editing mode

Figure 4.11: Testing program using a 3-key constrained keypad design in predictive disambiguation mode
4.10.3 Procedure

The study was performed in the Human-Computer Interaction Laboratory in the College of Computer and Information Science at Northeastern University. Before testing started, participants first completed a questionnaire asking for background information on their use of mobile devices. They were then briefly introduced to the normal dictionary-based predictive disambiguation text entry method and the functionalities of our 5-Key keypad design, including inputting characters, cycling through candidate lists, committing words and phrases, and correcting errors. Two training phrases were then given to the participants to familiarize them with the usage of the testing program.

Following this training was the first session of testing, during which participants entered a set of twenty text phrases using either a normal frequency-based predictive disambiguation method or our context-aware method. Participants were then asked to enter the other set of twenty text phrases in the second session using the other method. The order of the two text phrase sets and the two methods were randomized and balanced in order to minimize the confounding effect of learning on the experiment results. Participants were not told about the order of the methods and phrase sets during the testing.

The short text phrases used in the testing were selected from the phrase set published by MacKenzie and Soukoreff [57]. Computer simulation shows that the KSPC values for inputting the entire set of testing phrases using the normal predictive disambiguation and our context-aware predictive disambiguation methods are 1.4467 and 1.3273 respectively. Thus, the participants should theoretically be able to save about 8.25% of the total keystrokes. Some examples of the text phrases used are shown in Table 4.3.

Finally, after both sets of text phrases were entered, participants were asked for their opinions about whether they noticed any differences in terms of their interactions with both text entry methods, and whether they felt that one method was more efficient than the other one.
4.10.4 Experiment Results

Table 4.4 shows the average text entry speed (in words per minute (WPM), assuming an average word length of five characters) for the frequency-based and context-based predictive disambiguation methods, along with the average number of errors the participants made for each short text phrase. Errors are defined as presses of the “Del” key to correct an input word (no other types of error corrections were recorded). Two tailed paired t-tests were tested for statistically significant differences in performances for the two predictive disambiguation methods. The statistical significances of the differences between the two methods are reported in the last column.

On average, the participants achieved text entry speeds of 7.31 WPM using the existing frequency-based predictive disambiguation method and 8.01 WPM for our context-aware method. The improvement of 0.70 WPM is about 9.58% of the original performance for the normal frequency-based method. The improvement also approximates the predicted theoretical reduction of 8.25% of the total keystrokes required. Statistical analysis showed a statistically significant difference between the text entry speeds for both methods (t = 0.003).

Experimental results also showed a reduction in the error rate for our context-based predictive disambiguation method of 0.388 errors per short phrase, compared with the existing method of 0.492 errors per phrase. Statistical significance was also observed for this metric (t = 0.016).
Table 4.4: Comparison of text entry speeds and error rates between normal and improved predictive disambiguation methods

<table>
<thead>
<tr>
<th></th>
<th>Frequency Only</th>
<th>Context-Aware</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Text Entry Speed (WPM)</td>
<td>7.31</td>
<td>8.01</td>
<td>0.003</td>
</tr>
<tr>
<td>Avg. Error Rate (Errors/Phrase)</td>
<td>0.104</td>
<td>0.082</td>
<td>0.016</td>
</tr>
</tbody>
</table>

With every ten short phrases being grouped into a period, learning effects for text entry speeds and error rates for users of both frequency-based and context-aware predictive disambiguation methods are plotted in Figure 4.12. Periods are defined so that the first 10 short phrases from the first session become period 1, the last 10 short phrases from the first session become period 2, and so on and so forth. Average performances are plotted over each of these four “periods”.

Figure 4.12: Learning effects on text entry speeds and error rates for frequency-based and context-aware predictive disambiguation methods

In the post-study questionnaire, participants gave positive answers when asked about whether they felt any differences in using both methods from an interaction standpoint, without considering the predictive
disambiguation accuracy. The average level of difference they felt was 3.06 with a standard deviation of 2.06 (with 1 being the least different and 10 being the most different).

For a second question on whether participants thought one of the two methods was more efficient than the other, about half of the participants (15 out of 32) correctly identified the improved predictive disambiguation method by pointing out that the context-based method was “smarter”. For the rest 17 participants, 6 of them did not notice much difference in terms of efficiency, and the other 11 thought that the standard method was more efficient.

4.10.5 Discussion of the Usability Experiment Results

The results of the experiment support our hypothesis that text entry speed would increase if the number of keystrokes required for inputting certain texts using dictionary-based predictive disambiguation text entry methods is reduced. From Table 4.4, it is clear that text entry users were able to use our improved predictive disambiguation method more effectively than the normal frequency based method by 9.58%. Moreover, the error rate was also statistically significantly reduced by 26.8% for the context-aware method, compared with the frequency-based method.

Initially, a major concern regarding the semantic and syntactic context-aware predictive disambiguation methods was that the theoretical savings of the number of required keystrokes might not be carried over to actual usage of the methods. This concern has been lessened by the results of the usability testing, which support our proposed method and simulation results.

The learning effects are clear from Figure 4.12. Users of both methods show continuous improvements. Although performance dropped a little during the third period, the context-aware method almost always outperformed the standard frequency-based method. Also, there was a slight increase in error rates while using the frequency-based method, while those resulting from the context-aware method remained constant during the four periods.
The context-aware predictive disambiguation method works well in the background and does significantly alter the way a user interacts with the text entry method. That is, the changes are almost transparent as far as the interface design is concerned. The only difference that users may notice is that word lists for an ambiguous keystroke sequence may be in a different order with the context-aware method. The fact that participants did not report a large difference in using one method versus another during the usability study supports the notion that the context-aware method does not seem to noticeably increase overall attention demands on users. However, as some participants commented, one potential limitation of the context-aware method is that disambiguation results can be different each time for the same keystroke sequence under different textual contexts. Therefore, users will always have to check the word list to verify their desired word. While for the standard predictive disambiguation method, users may eventually memorize the number of required “Next” key presses required for a given word and input the word directly without looking at the display.

4.11 Conclusion and Future Work

For this part of the dissertation, significant improvements were made to existing dictionary-based predictive disambiguation text entry methods. Novel semantic relatedness and part-of-speech models were developed and utilized by existing predictive disambiguation methods to achieve better disambiguation accuracy and fewer required keystrokes. Laboratory usability testing with this new context-aware method supported the method’s faster text entry performance as was first predicted using simulation. Significant reduction in the error rates was also observed for the new context-aware method, and there seems to be minimal additional attention costs by using the context-aware method, at least for novice users.

The proposed semantic relatedness and POS models can be easily trained using a large text corpus. Because of the relatively larger size of the semantic relatedness model, a cutoff threshold method was investigated to reduce the model sizes while still maintaining reasonable improvements in predictive disambiguation performance.
A method for finding the best linear combination of frequency, semantic relatedness, and POS models for the KSPC metric were also introduced in this dissertation work and used in our implementation of the context-aware predictive disambiguation method.

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

A potential limitation of the context-aware method is the fact that the same ambiguous keystroke sequence can produce word lists with different orderings due to the current context, possibly confusing a user who saw a different order in the past and/or increasing the effort required to locate a desired word. This concern is most relevant to expert users, and longitudinal studies could be conducted to find out whether initial benefits in text entry performance and usability for novice users provided by the context-aware predictive disambiguation method continue to outweigh any longer-term limitations. A possible solution is to put the most probable words, based on frequency, semantic relatedness, and part-of-speech validity models, at the top of the candidate list, and simply order the rest of those less probable words by alphabetical order. This may greatly reduce the visual effort in searching for a word that is not recognized as a most probable choice by the predictive disambiguation method. Another possibility may be through user-customized ordering of word lists for certain keystroke sequences and context conditions.

Finally, possible future work may also include the development of predictive disambiguation methods that are more tolerable to errors, a problem discussed by MacKay [52]. For dictionary-based methods, a word cannot be input unless the correct sequence of keystrokes is made first. However, because of the less formal style of mobile text entry, it can be difficult for the user to make sure that every keystroke is correct. Therefore, devising dictionary-based predictive disambiguation methods that allow for incorrect keystrokes should be research that is well worth investigating.
Chapter 5

CPC Error Metric

5.1 Introduction

Most people will make occasional errors when interacting with any system or device. Good interface designs can increase usability and effectively reduce the probability of making errors. Therefore, error analysis has always been an important part of HCI for judging the effectiveness of interface designs, especially for text entry. This chapter describes a new error metric that was developed and evaluated as the third part of this dissertation. Existing problems with the current error analysis techniques are described first in Section 5.2. To address some of these problems, a new type of error metric is described in Section 5.3. To demonstrate the usefulness of the error metric, it was used in the analysis of data from three experiments conducted as part of this dissertation work. Results of the analysis and discussion of these results are presented in Sections 5.4 and 5.5. Section 5.6 summarizes the work with this new error metric and proposes future work.

5.2 Mobile Text Entry Methods Error Analysis Problem

Past research [e.g., 57, 74, 75, and 81] has provided excellent metrics for post-experimental text entry error analysis, even providing automated analysis software [75]. However, as mentioned in Section 2.1.3, current metrics used for measuring text entry error rates have limitations in terms of the types of errors they account for, and cannot easily distinguish between different types of errors. For example, as mentioned by Soukoreff and MacKenzie [74] as well as by Wobbrock and Myers [81], analysis done on the character level cannot handle some particular cases very well, such as when errors are not noticed and corrected immediately. Furthermore, existing metrics, such as the one proposed in [74], do not incorporate possible presses of function keys, such as the “next” key for predictive text entry methods, or arrow keys on full keyboards.
5.3 Solution for the Mobile Text Entry Error Analysis Problem

To address some of the limitations of previous metrics, a new text entry error metric was developed [28]. Specifically, by moving beyond keystroke level error measurement, the metric accounts in detail for the way the user handles corrections during text entry, and provides additional details about the nature and use of various text entry methods. The feasibility and usefulness of this new metric is shown through the analysis of the three usability experiments that were conducted for this dissertation.

5.3.1 Error Rate Analysis

Suppose we consider the following three different keystroke sequences for inputting the word “quick,” as shown in Table 5.1. Bold and italic texts in parentheses are incorrect key presses and left arrows are backspace key presses:

<table>
<thead>
<tr>
<th>Sequence 1</th>
<th>(pui←←←) quick</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence 2</td>
<td>(p←) q (r←) (r←) uick</td>
</tr>
<tr>
<td>Sequence 3</td>
<td>(p←) q (v←) u (j←) ick</td>
</tr>
</tbody>
</table>

Table 5.1: Key sequences that have the same correction efficiency but a different cost per correction

For the first sequence, the user incorrectly input the letter “p” instead of “q”. This error was not noticed until two other correct letters were input, so the user corrected this by deleting all three letters already typed (one incorrect and two correct), and started the word “quick” again. For the second sequence, the user initially input “p” instead of “q,” but noticed the error immediately and made the correction. However during the inputting of the letter “u”, the same error was repeated twice, and the correct letter was only input after the third try. For the third sequence, the user made three errors while trying to input three different characters. The user noticed the incorrect keystrokes immediately each time, and made three corrections during second tries.

When these three sequences are analyzed based on the keystroke categories proposed by [74], they all have 5 correct characters (C), 3 incorrect but fixed characters (IF), and 3 fixing keystrokes (F). Therefore,
all the keystroke level metrics will give the same results for these three significantly different keystroke sequences. In order to more accurately account for these differences in text entry, we introduced “Correction” as a new class for text entry error analysis.

5.3.2 Cost Per Correction

Looking at the examples in Table 1, the primary difference between the three sequences is the number of corrections the user made. For Sequence 1, three backspace keystrokes were used to correct an error that occurred during the inputting of only one character. For Sequence 2, three backspace keystrokes were used for correcting errors that occurred while inputting two characters. For Sequence 3, three backspace keystrokes were used to correct an equal number of errors. We define keystroke classes as following:

- **WLS** – Wasted Letter Key Strokes. Same as IF in [74].
- **FS** – Fixing Key Strokes. Same as F in [74].
- **WCS** – Wasted Control Key Strokes. For predictive text entry methods, there may be control keystrokes (e.g., the “next” key) that do not appear in transcribed text. All such keystrokes belong to this classification.

Given the above keystroke classes, we can define Correction as a sequence of consecutive keystrokes that may consist of WLS, WCS, and FS. One sequence of correction keystrokes is used only for correcting errors made while inputting one character. Correction keystrokes do not appear in the transcribed text. The Number of Corrections (NoC) is the total number of Corrections that happen during a text entry task.

With above definitions, we define the metric CPC (Cost per Correction), as follows:

\[
CPC = \frac{WLS + WCS + FS}{NoC}
\]  

(12)

For the above example, Sequences 1, 2, and 3 have one, two and three Corrections respectively. The corresponding CPC values are 6, 3, and 2, respectively. Since making one error and deleting it should at least use two keystrokes, a CPC value is usually greater than or equal to 2.
With the novel definitions of the WCS and NoC classes, the new CPC metric only takes into account those wasted control and correction keystrokes that are related to corrected errors. Therefore, it is not affected by the number of correct key strokes made, and usually cannot be derived from other existing error metrics because of different keystroke class definitions.

5.4 Testing and Analysis

In order to analyze our new error metric and to demonstrate its usefulness, we conducted a CPC analysis on the experimental data collected for the three experiments performed as part of this dissertation work. The first experiment compared constrained and unconstrained keypad designs for predictive disambiguation text entry methods (Section 3.6). The second experiment extended the first experiment by incorporating numbers, uppercase letters, and punctuation marks in the keypad designs (Section 3.7). The third experiment compared the normal dictionary-based predictive disambiguation method that only uses frequency information with our context-aware predictive disambiguation method that also uses semantic and syntactic contextual information (Section 4.9).

The results are shown in Table 5.2. The acronyms used in the table were defined previously, except for CLS, which stands for the correctly transcribed letter keystrokes. Since the new metric only works on the input keystroke streams, we only include the Total Error Rate (TERR) and Correction Efficiency (CE) metrics from [74] for comparison purposes.

The overall CPC value for all data is 9.65. This number means that in terms of the predictive text entry method we used, and regardless of keypad designs, predictive disambiguation methods, or error correction mechanisms, on average about 9.65 key presses would be used or wasted for one correction.

Remember that the deletion operation designed in the third experiment was word-based. Users were capable of using one “Del” key press to delete the entire incorrect word. No single characters were allowed to be deleted alone for that design. Therefore, the methods tested in the third experiment resulted in the least number of fixing key presses, and thus had Correction Efficiencies (CE) greater than 1.
<table>
<thead>
<tr>
<th>Design</th>
<th>CLS</th>
<th>WLS</th>
<th>WCS</th>
<th>FS</th>
<th>NoC</th>
<th>TERR</th>
<th>CE</th>
<th>CPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp 1 Constrained</td>
<td>4608</td>
<td>295</td>
<td>98</td>
<td>295</td>
<td>82</td>
<td>0.060</td>
<td>1.0</td>
<td>8.39</td>
</tr>
<tr>
<td>Exp 1 Unconstrained</td>
<td>2304</td>
<td>83</td>
<td>31</td>
<td>83</td>
<td>27</td>
<td>0.035</td>
<td>1.0</td>
<td>7.30</td>
</tr>
<tr>
<td>Exp 1 Int’l Standard</td>
<td>2304</td>
<td>94</td>
<td>31</td>
<td>94</td>
<td>37</td>
<td>0.039</td>
<td>1.0</td>
<td>5.92</td>
</tr>
<tr>
<td>Exp 2 Constrained</td>
<td>12766</td>
<td>1194</td>
<td>272</td>
<td>1194</td>
<td>285</td>
<td>0.086</td>
<td>1.0</td>
<td>9.33</td>
</tr>
<tr>
<td>Exp 2 Unconstrained</td>
<td>12766</td>
<td>901</td>
<td>190</td>
<td>901</td>
<td>222</td>
<td>0.066</td>
<td>1.0</td>
<td>8.97</td>
</tr>
<tr>
<td>Exp 3 Frequency</td>
<td>15152</td>
<td>1154</td>
<td>1567</td>
<td>315</td>
<td>281</td>
<td>0.071</td>
<td>3.66</td>
<td>10.80</td>
</tr>
<tr>
<td>Exp 3 Context</td>
<td>15152</td>
<td>927</td>
<td>1172</td>
<td>248</td>
<td>220</td>
<td>0.058</td>
<td>3.74</td>
<td>10.67</td>
</tr>
<tr>
<td>Overall</td>
<td>65052</td>
<td>4648</td>
<td>3361</td>
<td>3130</td>
<td>1154</td>
<td>0.067</td>
<td>1.48</td>
<td>9.65</td>
</tr>
</tbody>
</table>

Table 5.2: CPC analysis on data from three previous experiments.

A series of two-tailed t-tests were conducted to find out whether there exists statistical significance between any pair of the seven analyzed sets of CPC data. Results revealed that the CPC data of the first experiment using the standard telephone keypad design has statistically significant differences ($t = 0.007$) to the CPC data of the second experiment using the unconstrained keypad design. Also, both sessions of the third experiment using normal and context-aware predictive disambiguation methods have statistically significant differences ($t < 0.05$) in CPC metric to data sets of other sessions including the first experiment using unconstrained and telephone standard keypad designs, and the second experiment using constrained keypad design. None of the other pairs of CPC data sets were found to be statistically significantly different than each other.

### 5.5 Discussion

The CPC values for the text entry method used in our experiments with different keypad designs and predictive disambiguation methods are high because of the nature of dictionary-based predictive disambiguation methods. Users will normally not be able to notice an incorrect key press until they finish the whole word. If the desired word does not appear in the candidate list, users usually delete all the...
correct key presses after that incorrect one. This process makes the cost for correcting one single incorrect key press quite high. Therefore, the CPC values reflect this high cost.

The high CPC values are also partly due to the large number of “wasted” control key presses that such predictive disambiguation text entry methods require. Usually users do not realize errors until they have cycled through the entire candidate word list, which tells them that the desired word is not available. These “next” keystrokes become wasted once the correction process begins. Non-ambiguous text entry methods, such as using a full keyboard, should not have as many wasted control keystrokes and therefore yield lower CPC values.

Additionally, high CPC values (which are not statistically different) for constrained and unconstrained keypad designs used in the first and second experiments show that keypad design is not the dominant factor for the efficiency of a text entry error correction process. Larger differences may be observed if CPC metrics for different text entry methods (e.g. full keyboard) are evaluated.

The third experiment tested dictionary-based predictive disambiguation text entry methods that used a word-based error correction design. Incorrect word input can be deleted with only one “Del” key press. However, because of the much smaller keypad design, thus much longer candidate word lists and more wasted control keystrokes, although more characters can be corrected with fewer fixing key press, it can be seen from Table 5.2 that both sessions of the third experiment using different predictive disambiguation methods still have the highest CPC values of 10.80 and 10.67. This shows the robustness of the CPC error metric for showing the real costs of different text entry method designs.

Because of the different error correction designs (character based versus word based), in addition to the higher average CPC values, statistically significant differences are also found between the results of the third experiment and those of the first two experiments.

Column TERR (of Table 5.2) shows the Total Error Rate values described by Soukoreff and Mackenzie [74] for data from different experiments. While the Total Error Rates for experimental sessions using the
constrained keypad designs are higher than those for sessions using the unconstrained or international standard keypad designs, the differences are not significant. Instead, CPC values clearly demonstrate the effects of different keypad designs and error correction mechanisms by showing very close CPC values for data from the first two experiments and much higher CPC values for data from the third experiment.

CPC can be a useful metric for showing the nature of error corrections for text entry methods. This may include the quality of feedback that influences the chance of errors being noticed, the amount of effort required by the designs of error correction methods, or even the usage habits of a particular user. Higher CPC values generally mean greater effort needed for the error correction task. Although the reasons for high CPC values need to be investigated in greater depth, the CPC metric itself seems to be a good starting point for future work in this area.

5.6 Conclusions and Future Work

The CPC error metric developed and studied as part of this dissertation work provides an additional level of detail on the types of errors that occur during text entry. The metric is also capable of showing the differences between different error correction mechanisms. Analysis using the CPC metric on previously collected data demonstrated its use and advantages. Results show the unique features of the CPC metric and its potential for use in research and usability studies.

Future research may include extending this error metric to incorporate the analysis of incorrect but not corrected characters in the final transcribed text. Additional testing with different text entry methods is also needed to see how the CPC metric differs from other types of error analysis metrics.
Chapter 6

Conclusions

6.1 Contributions of this Dissertation

Text entry on mobile devices has always been an interesting and important problem for HCI researchers. One challenge which differentiates this particular research area from more generalized text entry research is the ever-decreasing sizes of mobile devices. Another challenge is the “anyplace/anytime” use of mobile devices. Finding input modalities and interaction techniques that work well with these devices makes this problem space challenging and difficult. However, progress in this area will potentially benefit the millions of people who use mobile devices, and may also lead to advances in universal usability of information systems.

This dissertation work describes several advances made by the author to improve dictionary-based predictive disambiguation text entry methods. Dictionary-based predictive disambiguation text entry methods are usually recognized as efficient methods for entering texts on mobile devices with small keypads. However, such methods still have limitations that arise from the ambiguity that comes from keypad (versus keyboard) use, the limited size and relevance of dictionaries used on devices, and relatively large learning curves that can result going from keyboard text entry to keypad text entry.

This dissertation has made significant contributions to three major areas in the field of mobile text entry research, focusing primarily on improving current dictionary-based predictive disambiguation text entry methods. The major contributions are 1) alternative keypad designs that promote novice usability in addition to high performance, 2) improvements to predictive disambiguation method itself, and 3) a metric that provides new insight into the analysis of mobile text entry errors.
6.1.1 Alternative Alphabetically Constrained Keypad Designs

The first contribution has been made to the keypad design problem of the dictionary-based predictive disambiguation text entry methods. Optimal keypad designs of various sizes with the constraint that characters must be placed under alphabetical order were developed. Computer simulations and human usability studies have shown that 1) although a little worse than their corresponding unconstrained designs, the alphabetically constrained keypad designs achieved reasonably good theoretical disambiguation performances, 2) however in the usability studies, the alphabetically constrained designs significantly outperformed the unconstrained ones because of the saved visual searching efforts with the alphabetical order, 3) usability studies also confirmed that the alphabetically constrained keypad designs (no matter with or without advanced controls for numbers, capital letters, and punctuations) provide greater ease of learning and require fewer attention demands from novice users.

This part of the dissertation work also made several contributions by finding that different optimal keypad designs will be derived for different theoretical performance metrics (DA and KSPC), for different word lists (Spoken, Written, and Short Message), and for different text entry methods (Dictionary-Based Predictive Disambiguation and MultiTap).

Some other contributions of this dissertation work include the development of a GA-based heuristic for finding optimal unconstrained keypad designs for dictionary-based predictive disambiguation text entry methods, and a dynamic programming algorithm for finding optimal constrained keypad designs for Multi-Tap text entry method under KSPC metric.

This work has investigated and focused on the keypad design problems for improving usability and performance for novice users. The objective was achieved through using an alphabetical order of characters across keys.
6.1.2 Improved Predictive Disambiguation Methods

The second part of the dissertation work dealt with improving current dictionary-based predictive text entry methods. The contributions include a novel semantic relatedness model and a part-of-speech model for measuring the semantic and syntactic likelihood of a candidate word being the user desired word. These models can be linearly combined with the common frequency model to achieve better disambiguation accuracy and fewer keystrokes per character. An empirical study was completed to validate the proposed context-aware predictive disambiguation methods. Significantly faster text entry performances with comparable error rates were observed for the method that utilized the context-aware predictive disambiguation method. Also, participants of the usability study did not report noticeable extra attention demands. As a related contribution, a method for finding the best linear combination (in KSPC metric) of the three models was also implemented.

Since the semantic relatedness model is trained on individual words, it has a relatively larger size compared with that of the part-of-speech model. Therefore as another contribution, a cutoff threshold method was investigated to filter out those entries below a certain threshold to reduce the size of the model. The performance of the reduced semantic relatedness model was found to also decrease, but with a much slower speed.

6.1.3 Novel Metric for Mobile Text Entry Error analysis

Lastly, contributions were made to the mobile text entry error analysis problem. The CPC error metric provides additional insights into the types of errors that occur and the types of error correction mechanisms that are used. Analysis was completed using the CPC error metric with the data collected during studies performed for dissertation work. The results demonstrated unique features and promising future uses of the new error metric.
6.2 Future Work

The marketing of new touch-screen types of mobile devices (e.g., Apple’s iPhone) opens a new horizon for existing text entry research work. The virtual keypad, instead of the fixed physical keypad design, brings brand new opportunities for the mobile text entry research described in this dissertation.

In terms of the keypad design problem, constrained keypad designs that adapt themselves according to recent usage history could be a good topic for further study. Existing models that predict user performances with mobile text entry methods on physical keypads need to be refined and further developed. Another limitation to be addressed is the lack of text corpora that capture a more realistic picture of users’ text entry on mobile applications, such as during text messaging. Such corpora are important for understanding the words (and their related frequencies) that people use with mobile devices so that the best solutions to the mobile text entry problem can be found. Finally, as previously stated, most mobile phone keypad users are also familiar with other widely adopted character layouts, such as the QWERTY keyboard design. In a recently published book by MacKenzie [58], several QWERTY-like mobile phone keypad designs were summarized and compared with other types of designs. Comparable or better performances were observed. Interesting future work would be to investigate possible improvements in text entry performance by replacing the alphabetical constraint that was used in our work with another layout constraint, such as keeping letters close to a QWERTY design.

In terms of future work in the area of context-aware predictive disambiguation methods, larger text corpora may be used to train the semantic relatedness model to see whether more performance improvements can be gained by using more training data, as word level language models cannot be easily saturated and are capable of continuous growth.

Other methods for ordering the matching candidate word list, such as alphabetical order, which should greatly reduce the mental demands from users, could also be worth investigating. Possible improvements to the predictive disambiguation methods may also include ways of allowing reasonable user errors in the keystroke sequences.
The CPC error metric, as stated previously, could be further extended to incorporate the analysis of more types of information contained in the final transcription. More testing could also be done to experiment with the metric and validates its usefulness.
Appendices

Appendix A. The BNC Basic Tagset

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AJ0</td>
<td>Adjective (general or positive) (e.g. good, old, beautiful)</td>
</tr>
<tr>
<td>AJC</td>
<td>Comparative adjective (e.g. better, older)</td>
</tr>
<tr>
<td>AJS</td>
<td>Superlative adjective (e.g. best, oldest)</td>
</tr>
<tr>
<td>AT0</td>
<td>Article (e.g. the, a, an, no)</td>
</tr>
<tr>
<td>AV0</td>
<td>General adverb: an adverb not subclassified as AVP or AVQ (see below) (e.g. often, well, longer (adv.), furthest)</td>
</tr>
<tr>
<td>AVP</td>
<td>Adverb particle (e.g. up, off, out)</td>
</tr>
<tr>
<td>AVQ</td>
<td>Wh-adverb (e.g. when, where, how, why, wherever)</td>
</tr>
<tr>
<td>CJC</td>
<td>Coordinating conjunction (e.g. and, or, but)</td>
</tr>
<tr>
<td>CJS</td>
<td>Subordinating conjunction (e.g. although, when)</td>
</tr>
<tr>
<td>CJT</td>
<td>The subordinating conjunction that</td>
</tr>
<tr>
<td>CRD</td>
<td>Cardinal number (e.g. one, 3, fifty-five, 3609)</td>
</tr>
<tr>
<td>DPS</td>
<td>Possessive determiner-pronoun (e.g. your, their, his)</td>
</tr>
<tr>
<td>DT0</td>
<td>General determiner-pronoun: i.e. a determiner-pronoun which is not a DTQ or an AT0.</td>
</tr>
<tr>
<td>DTQ</td>
<td>Wh-determiner-pronoun (e.g. which, what, whose, whichever)</td>
</tr>
<tr>
<td>EX0</td>
<td>Existential there, i.e. there occurring in the there is ... or there are ... construction</td>
</tr>
<tr>
<td>ITJ</td>
<td>Interjection or other isolate (e.g. oh, yes, mhm, wow)</td>
</tr>
<tr>
<td>NN0</td>
<td>Common noun, neutral for number (e.g. aircraft, data, committee)</td>
</tr>
<tr>
<td>NN1</td>
<td>Singular common noun (e.g. pencil, goose, time, revelation)</td>
</tr>
<tr>
<td>NN2</td>
<td>Plural common noun (e.g. pencils, geese, times, revelations)</td>
</tr>
<tr>
<td>NP0</td>
<td>Proper noun (e.g. London, Michael, Mars, IBM)</td>
</tr>
<tr>
<td>ORD</td>
<td>Ordinal numeral (e.g. first, sixth, 77th, last)</td>
</tr>
<tr>
<td>PNI</td>
<td>Indefinite pronoun (e.g. none, everything, one [as pronoun], nobody)</td>
</tr>
<tr>
<td>PNP</td>
<td>Personal pronoun (e.g. I, you, them, ours)</td>
</tr>
<tr>
<td>PNQ</td>
<td>Wh-pronoun (e.g. who, whoever, whom)</td>
</tr>
<tr>
<td>PNX</td>
<td>Reflexive pronoun (e.g. myself, yourself, itself, ourselves)</td>
</tr>
<tr>
<td>POS</td>
<td>The possessive or genitive marker 's or '</td>
</tr>
<tr>
<td>PRF</td>
<td>The preposition of</td>
</tr>
<tr>
<td>PRP</td>
<td>Preposition (except for of) (e.g. about, at, in, on, on behalf of, with)</td>
</tr>
<tr>
<td>PUL</td>
<td>Punctuation: left bracket - i.e. ( or [</td>
</tr>
<tr>
<td>PUN</td>
<td>Punctuation: general separating mark - i.e. , !, : ; - or ?</td>
</tr>
<tr>
<td>PUQ</td>
<td>Punctuation: quotation mark - i.e. ' or &quot;</td>
</tr>
<tr>
<td>PUR</td>
<td>Punctuation: right bracket - i.e. ) or ]</td>
</tr>
<tr>
<td>TO0</td>
<td>Infinitive marker to</td>
</tr>
<tr>
<td>UNC</td>
<td>Unclassified items which are not appropriately considered as items of the English lexicon.</td>
</tr>
<tr>
<td>VBB</td>
<td>The present tense forms of the verb BE, except for is, 's: i.e. am, are, 'm, 're and be [subjunctive or imperative]</td>
</tr>
<tr>
<td>VBD</td>
<td>The past tense forms of the verb BE: was and were</td>
</tr>
<tr>
<td>VBG</td>
<td>The -ing form of the verb BE: being</td>
</tr>
</tbody>
</table>
Appendix B. The BNC Ambiguity Tagset

As in the first version of the BNC, a limited number of ambiguity tags were introduced, to deal with particular cases where the tagger has difficulty in distinguishing two categories, and where incorrect taggings would otherwise result rather frequently. The ordering of tags is significant: it is the first of the two tags which is estimated by the tagger to be the more likely. Hence the interpretation of an ambiguity tag X-Y may be expressed as follows: ‘There is not sufficient confidence to choose between tags X and Y; however, X is considered to be more likely. The list of ambiguity tags is:

<table>
<thead>
<tr>
<th>Ambiguity tag</th>
<th>Ambiguous between</th>
<th>More probable tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>AJ0-NN1</td>
<td>AJ0 or NN1</td>
<td>AJ0</td>
</tr>
<tr>
<td>AJ0-VVD</td>
<td>AJ0 or VVD</td>
<td>AJ0</td>
</tr>
<tr>
<td>AJ0-VVG</td>
<td>AJ0 or VVG</td>
<td>AJ0</td>
</tr>
<tr>
<td>AJ0-VVN</td>
<td>AJ0 or VVN</td>
<td>AJ0</td>
</tr>
<tr>
<td>AV0-AJ0</td>
<td>AV0 or AJ0</td>
<td>AV0</td>
</tr>
<tr>
<td>AVP-PRP</td>
<td>AVP or PRP</td>
<td>AVP</td>
</tr>
<tr>
<td>AVQ-CJS</td>
<td>AVQ or CJS</td>
<td>AVQ</td>
</tr>
<tr>
<td>CJS-AVQ</td>
<td>CJS or AVQ</td>
<td>CJS</td>
</tr>
</tbody>
</table>
Appendix C. Groupings of Part-of-Speech Tags from the BNC Basic and Ambiguity Tagsets Used by the improved Predictive Disambiguation POS Model

<table>
<thead>
<tr>
<th>Category</th>
<th>POSes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sentence Start</td>
</tr>
<tr>
<td>1</td>
<td>AJ0, AJC, AJS, AJ0-NN1, AJ0-VVD, AJ0-VVG, AJ0-VVN</td>
</tr>
<tr>
<td>2</td>
<td>AT0</td>
</tr>
<tr>
<td>3</td>
<td>AV0, AVP, AVQ, AV0-AJ0, AVP-PRP, AVQ-CJS</td>
</tr>
<tr>
<td>4</td>
<td>CJC, CJS, CJT, CJS-AVQ, CJS-PRP, CJT-DT0</td>
</tr>
<tr>
<td>5</td>
<td>DPS, DT0, DTQ, DT0-CJT</td>
</tr>
<tr>
<td>6</td>
<td>EX0</td>
</tr>
<tr>
<td>7</td>
<td>ITJ</td>
</tr>
<tr>
<td>8</td>
<td>NN0, NN1, NN2, NP0, NN1-AJ0, NN1-NP0, NN1-VVB, NN1-VVG, NN2-VVZ, NP0-NN1</td>
</tr>
<tr>
<td>9</td>
<td>PNI, PNP, PNQ, PNX, PNI-CRD, PRP-AVP, PRP-CJS</td>
</tr>
<tr>
<td>10</td>
<td>PRF, PRP</td>
</tr>
<tr>
<td>11</td>
<td>TO0</td>
</tr>
<tr>
<td>12</td>
<td>VBB, VBD, VBG, VBI, VBN, VBZ</td>
</tr>
<tr>
<td>13</td>
<td>VDB, VDD, VDG, VDI, VDN, VDZ</td>
</tr>
<tr>
<td>14</td>
<td>VHB, VHD, VHG, VHI, VHN, VHZ</td>
</tr>
<tr>
<td>15</td>
<td>VM0</td>
</tr>
<tr>
<td>16</td>
<td>VVB, VVD, VVG, VVI, VVN, VVZ, VVB-NN1, VVD-AJ0, VVD-VVN, VVG-AJ0, VVG-NN1, VVN-AJ0, VVN-VVD, VVZ-NN2</td>
</tr>
<tr>
<td>17</td>
<td>XX0</td>
</tr>
<tr>
<td>18</td>
<td>ZZ0</td>
</tr>
<tr>
<td>19</td>
<td>UNC</td>
</tr>
</tbody>
</table>
Bibliography


Augmentative and Alternative Communication, 3, 192-195.


