Resolving Ambiguity in Overloaded Keyboards

Francine Evans
Schlumberger Laboratories
Houston, TX
fevans@houston.oilfield.slb.com

Steven Skiena
Dept. of Computer Science
SUNY Stony Brook
skiena@cs.sunysb.edu

Amitabh Varshney
Dept. of Computer Science
University of Maryland
varshney@cs.umd.edu

Abstract

We have developed a powerful engine which exploits statistical and grammatical constraints in English to resolve ambiguity from an input stream of overloaded characters. In this paper, we apply this engine to a variety of text-oriented user interfaces based on character overloading. We demonstrate that surprisingly accurate reconstruction is possible using such low-bandwidth input devices as telephone keypads, one-handed and minimum-motion keyboards, eye tracking devices, low-resolution image scanners, and VR datagloves.

Author's Note

Although most of us know Godfried though his work in computational geometry, his early research focused on pattern recognition. In particular, he wrote an interesting series of papers on text recognition around 1980 [15, 16, 17, 20]. We dedicate this paper to Godfried in recognition of his wide range of contributions and interests.

1 Introduction

An important goal of user interface design is minimizing the amount of user effort to express a gesture or enter information. For example, user performance on pull-down menu systems depends substantially on the order in which items appear on the menu [19].

Substantial productivity gains are possible for text-entry systems by overloading multiple characters on single keys. This requires sophisticated algorithmic techniques to resolve the ambiguity inherent in overloading. In our previous work [13], we have developed a language model which permits single-stroke text-entry on telephone keypads, as discussed in Section 3, Here, we extend our techniques to a variety of other interesting user interface applications, including:

- *Minimizing motion on QWERTY keyboards* – To speed text entry or minimize finger motion on conventional QWERTY keyboards, our techniques can be used to resolve ambiguity so that a typist need never move their fingers off the home row of the keyboard. Identifying the correct choice between the home key, the one above, and the one below is exactly analogous to selecting one of three letters on a telephone keypad. Any remaining ambiguity can be explicitly resolved by moving the finger up or down.

- *Customized keyboards for text entry* – There are a variety of special-purpose keyboards which can be made more efficient via prediction techniques, such as one-handed chord keyboards [5], small or soft keyboards for portable computers, or court stenography keyboards [3]. A particularly interesting type of keyboard is the one-handed half-QWERTY keyboard [11], designed for high-portability applications and the disabled. An advantage of the half-QWERTY keyboard is that the user can exploit typing skills developed on conventional keyboards, although left-right mode switch errors are common. Our predictive methods can detect and correct mode switch errors.

- *Keyboards for the disabled* – Severely disabled people often lack sufficient motor control to type on a keyboard of more than 6-10 keys. Further, each stroke typically requires non-trivial concentration and effort, so minimizing the number of keystrokes is critically important to creating a usable system. We propose using heavily overloaded keyboards, with the ambiguity being resolved using our methods, thus significantly reducing the effort needed to enter text.

- *Error Correction of Scanned Text* – Both handwriting and optical character recognition (OCR) systems suffer from substantial rates of character misrecognition. Using character trigram statistics or dictionaries to detect and correct such errors is a standard technique in both OCR and handwriting recognition. In this paper, we demonstrate that substantially higher error correction rates are achievable by exploiting grammatical constraints using our reconstruction engine.

In this paper, we present experimental results and human factor studies on the achievable reconstruction accuracy for all of these applications. Specifically:

- We demonstrate the practical viability of "minimum-motion" keyboards, where all typing is done with fingers left in the home position. Our system achieves character reconstruction rates of at least 98.4% for all standard keyboard layouts, including QWERTY and Dvorak.

- We study the impact of keyboard layout on reconstruction error rates. Surprisingly, we show that keyboard designs which balance character frequencies do not significantly outperform more natural alphabetic layouts.

- We study the impact that reducing the number of keys has on reading comprehension, and establish a bound on the minimum number of keys sufficient for understanding. Through human-factor experiments, we found that readers are able to accurately understand text entered using an overloaded keyboard with only 6 keys.

  This observation has important implications for eye-tracking systems based on electro-oculography (EOG) [8], a less expensive technology than expensive reflectance-based eye-tracking methods. Current technology only provides sufficient resolution to implement a 2×3 button keyboard. Our ambiguity resolution techniques result EOG systems which are substantially easier to use than those employing traditional hierarchical menus.

- We demonstrate that grammar-based disambiguation methods significantly out perform standard dictionary lookup in error recovery for OCR and handwriting recognition. Specifically, our grammar-based methods correctly reconstruct roughly 83.6% of unknown characters, where only 76.3% accuracy is possible using dictionary-based methods.

Our paper is organized as follows. In Section 2, we discuss previous work on text entry and keyboard design. Section 3 describes the reconstruction algorithm underlying our engine. Our experimental results begin in Section 4, where we consider the impact of keyboard design on reconstruction accuracy. Section 5 presents our results on reconstruction accuracy achievable with minimum-motion versions of standard keyboard layouts. Section 6 describes our human factors experiments to study the impact of reconstruction-error rate on reading comprehension. In Section 7, we apply our techniques to OCR systems. Finally, in Section 8 we suggest avenues for further research.

2 Keyboard Designs

The QWERTY (or Sholes) layout has been standard on almost all keyboards since the invention of the typewriter by Christopher Sholes in 1868. The impact of keyboard design on typing performance was recognized immediately; popular legend holds that the QWERTY layout was designed to slow typing so as to minimize jamming on the original typing mechanism.

Previous research on optimizing keyboard layout lead to alternate designs that seek to improve typing speed, reduce physical discomfort, minimize learning time, and reduce keying errors. The Dvorak simplified keyboard places the ten most frequently used letters on the home row, with vowels on the left and consonants on the right, so as to minimize finger movement. The Maltron keyboard [10] also places the ten most frequently used letters on the home keys with the exception of letter 'e', which is placed on its own separate thumb key. Malt [10] has done an extensive comparison of the three layouts and concluded that this keyboard layout minimizes keying error rates. Table 1 illustrates the QWERTY, Dvorak, and Maltron layouts. In Section 5, we consider 'minimum-motion' versions of each of these

Hand / Finger	QWERTY	Dvorak	Maltron
Right / First	j u h y n	f g d h b	v m d t
Right / Middle	k i m	c t w	u h w
Right / Ring	l o	r n v	z o
Right / Pinkie	p	l s z	l r x
Left / First	f g t r b v	p y i u k x	c b s f g
Left / Middle	d e c	e j	y i j
Left / Ring	s w x	o	p n
Left / Pinkie	a q z	a	q a
Left / Thumb	␢	␢	e
Right / Thumb	␢	␢	␢

Table 1: QWERTY, Dvorak, and Maltron keyboard layouts.

keyboards, where each symbol is specified only by any motion of the appropriate finger, and ambiguity resolved using our reconstruction engine.

Several recent keyboard have strove to minimize hand movement so as to reduce the risk of repetitive stress injury, including carpal tunnel syndrome. These include the Data-Hand keyboard which adjusts to the user's hands, and the Kinesis Ergonomic Keyboard which arranges keys in vertical columns and not in diagonals as on a traditional keyboard. Other special-purpose keyboard designs include one-handed chord keyboards [5], small or soft keyboards for portable computers, and court stenography keyboards [3]. A particularly interesting type of keyboard the one-handed half-QWERTY keyboard [11], designed for high-portability applications and the disabled. Our predictive methods have potential to enhance any of these keyboard designs.

Previous work on predictive keyboards and keyboard accelerators includes [2]. Our work differs significantly since we propose typing on inherently ambiguous keyboards to facilitate special-purpose input devices and new applications, while previous work focuses on interactive prediction and escape mechanisms to speed text-entry on unambiguous keyboards.

Rau and Skiena [13] developed an system exploiting statistical and grammatical constraints to reconstruct text entered on a standard overloaded telephone key pad at one keystroke per symbol. In experiments, they were able to correctly reconstruct up to 99% of characters in standard texts that were pressed on a telephone keypad. This is significantly easier than conventional protocols which employ pairs of key presses to disambiguate which letter was intended.

The same problem has been addressed by Litvak and Shamir[9], who use a weighted tree structure called a probabilistic history tree to convert overloaded symbols to text. This method does not achieve near the accuracy of Rau and Skiena, although their underlying model is smaller and they provide more immediate feedback. Litvak and Shamir [9] have applied this technology to handwriting recognition.

3 The Reconstruction Engine

The reconstruction engine used in our studies is modified from the original Rau-Skiena [13] engine. We have enhanced it so that the user can specify an arbitrary keyboard layout, and to permit overloading the same symbol on multiple different keys, a feature necessary to support applications to OCR.

Here we describe the Rau-Skiena algorithm for overloaded symbol reconstruction. It takes as input a stream of digits, and outputs a maximum-likelihood reconstructed English source text, after the following phases:

Construction of Word Candidate Sets: The dictionary matching phase seeks to map each key sequence with a unique word in our dictionary. Because of overloading, multiple dictionary words can map to the same key sequence. For example, Table 2 demonstrates that certain code strings for minimum-motion QWERTY collides with 15 distinct English words.

Designing a good dictionary involves tradeoffs. All common words must be in the dictionary, although adding very seldom-used words might decrease the overall accuracy due to increased collisions with other words. Our dictionary was built from the UNIX `ispell` word list [12], enhanced by other common words. Abbreviations such as BMW and ACM were removed, yielding a dictionary with 53324 words hashed to 44474 code entries. With each word in the dictionary is a frequency count from the Brown Corpus, discussed below.

Hits	Digits	Words
15	5335	BOOT BOOR BOLT BOOB VOLT GOOF GOLF TOOT BLOT FOOT BLOB ROOT ROOF GLOB FLOG
15	585	RAT TAT TAR RAG VAT TAG TAB GAG GAB BAT BAR BAG FAT FAR FAG
14	5315	GOUT TOUT TOUR BLUR FOUR BONG BOHR ROUT GLUT GONG FONT TONG FLUB BOUT
14	5856	FARE RAGE GATE BABE RARE TATE FATE GAVE RATE BARE BARD GAGE RAVE BATE
13	535	ROT TOT TOR ROB TOG GOT GOG GOB BOG BOB FOR FOG FOB
12	581	RAY RAN TAU RAJ TAN VAN GAY BAY BAN BAH FAY FAN
12	5336	BOLE BOLD GOOD GOLD FOOD BLOC TOLD FOLD ROOD ROLE FLOE FLOC
11	525	FIB TIT RIG RIB GIG BIT BIG BIB FIT FIR FIG
11	5857	VATS RAGS FATS BAGS RATS TABS BARS GAGS TAGS FAGS BATS
11	5855215	TABBING GABBING TARRING BARGING TAGGING BATTING GAGGING BARRING RAGGING BAGGING TATTING

Table 2: Code strings with the most interpretations for minimum-motion QWERTY.

Unknown Word Analysis: If no dictionary entry exists for the token, a character interpretation is constructed using partial dictionary matches, prefix/suffix analysis, and by optimizing over character probabilities and transitional letter probabilities.

Sentence Disambiguation: A sentence is composed of a list of tokens terminated with a period. Each token is now labeled with a list of one or more character interpretations, defining a set of possible sentence interpretations. There may be an exponential number of possible sentence candidates. Grammatical constraints are employed within the Viterbi algorithm to find the most likely sentence for the phone code string. These constraints can be formulated in terms of transition probabilities:

$P(Z_i/X_i)$, the probability that the word Z_i was intended when the code X_i was observed.

$P(Z_i/Z_{i-1})$, the transitional probability that the word Z_i is observed when the previous word was Z_{i-1}.

A directed graph represents the word lattice that forms the set of sentence candidates. Each of the nodes and edges have costs assigned with them. The node costs are the word probabilities for the observed code, while the edge costs are the transition probabilities between word pairs. Any path between the start and end node represents a possible sentence. The Viterbi algorithm finds the maximum-cost path in the trellis using dynamic programming. See [18] and [7] for detailed descriptions of the Viterbi algorithm.

To increase the flexibility of the reconstruction engine, we changed the main dictionary data structure from a hash table to a trie. By doing a depth-first traversal of the appropriate portion of the trie data structure, we can efficiently retrieve all dictionary strings which partially match a query string. Hash tables cannot efficiently support such queries without hashing multiple copies of each string, which requires a prohibitive amount of space. Further, this trie structure could be built roughly four times faster than an analogous hash table.

Our trie data structure enables us to include unknown "wildcard characters" needed to support OCR applications. Our trie data structure contains only one entry in the trie per dictionary word. To search for a word that contains a wildcard character, we do a depth-first traversal from the root. Whenever we come to the wildcard position, we traverse all the nodes at the point we are up to in the tree, pruning the search on all mismatched characters.

4 Impact of Keyboard Layout on Reconstruction

In many of our proposed applications, such as eyetracking systems for the disabled, no standard keyboard layout exists. Thus we have the flexibility to design our virtual keyboard to facilitate maximum reconstructibility. To assess the impact of layouts on reconstructibility, we studied two classes of keyboard designs:

- *Sequential layouts* – A particularly easy-to-learn keyboard layout places runs of consecutive letters on successive keys. Thus knowledge of alphabetical order enables one to identify the correct key for any symbol. Telephone keypads observe this convention, by placing runs of 3 consecutive letters on each key.

  Our sequential layouts are consistent with that of the telephone layout. We placed from 4 to 8 consecutive letters on each key, as shown in Table 3.

- *Bin-packing layouts* – Information-theoretic principles dictate that maximum reconstructibility should result when each key is pressed with equal frequency. For this reason, we experimented with bin-packing approaches which balance the symbol probabilities for each key. Because bin-packing is an NP-complete problem, we used a heuristic based on randomization. After first assigning symbols randomly to the specified keys, our bin-packing algorithm repeatedly selects random pairs of symbols and swaps the letters between bins if this helps to balance the sum of the probabilities across the bins. Table 4 shows the actual bin packing layouts that resulted.

Number of Keys	9 (Phone)	7	5	4	4	3
Sequential Layout	abc def ghi jkl mno prs tuv wxy qz	abcd efgh ijkl mnop qrst uvwx yz	abcde fghij klmno pqrst unwxyz	abcdef ghijkl mnopqrs tuvwxyz	abcdefg hijklmn opqrstu vwxyz	abcdefgh ijklmnopq rstuvwxyz
Clinton %Char Correct	99.04%	98.25%	96.51%	94.52%	92.61%	88.84%
Clinton %Words Correct	97.30%	95.60%	92.40%	88.69%	86.66%	80.50%
Moby %Char Correct	96.95%	94.75%	90.81%	87.80%	84.20%	79.05%
Moby %Words Correct	92.56%	88.84%	82.91%	77.92%	74.55%	66.87%

Table 3: Sequential layouts and accuracy of reconstruction.

Our experiments were conducted on two sample texts. *Clinton* is a collection of President Clinton speeches, consisting of about 1 million characters and 200,000 words. *Moby* is a similar-sized excerpt from Moby Dick consisting of about 1.1 million characters. Tables 3 and 4 give the percentages of characters and words which were successful reconstructed for each document on each keyboard.

For the most part, the bin packing layouts lead to slightly greater reconstruction accuracy over the sequential layouts for the corresponding number of keys for both texts. However, the performance difference between the two layouts was smaller than we initially expected, indeed it is probably not sufficient in most applications to justify the resulting steeper learning curve. On all layouts, we were able to perform significantly better on *Clinton* than on *Moby*, not surprising since *Moby* contains archaic words and a higher percentage of proper nouns which are difficult to reconstruct.

These experiments demonstrate that reconstruction accuracy decreases slowly as the number of keys is reduced from 26 to 6, beyond which a more dramatic decrease in accuracy occurs. Even when using only three keys over 86% of characters on *Clinton* are reconstructed correctly; however this seemingly impressive rate obscures the fact that such text is essentially

Number of Keys	9	8	7	6	5	4	3
Bin packing Layouts	dn grc ap e ybmh li wt jsuxqzk vfo	ipd lsg omc e avfk hn jbwyr tuzxq	ge hskb djuo yrn vilw axqcp tmfz	syo ewzfj dhxnk gual cqir mvtpb	pvmuztc irdf alwyjbg ekqn xhso	oubfpaqjzx csdtvy lekn wghrmi	aeo cdgjlmptuwxy rbvfhiknqsz
Clinton %Char Correct	99.18%	99.05%	98.74%	97.93%	96.91%	92.48%	90.22%
Clinton %Words Correct	97.02%	97.12%	96.41%	94.85%	93.04%	85.21%	82.38%
Moby %Char Correct	97.70%	97.56%	96.31%	94.52%	92.20%	85.78%	77.16%
Moby %Words Correct	93.13%	93.26%	90.79%	87.97%	85.36%	74.84%	63.01%

Table 4: Bin packing layouts and accuracy of reconstruction.

incomprehensible. In Section 6, we report on human factor experiments to measure the effect on reading comprehension as the number of keys decrease.

5 Minimum Motion Keyboards

Minimum motion keyboards are an intriguing idea for high-speed text entry. By using ambiguity resolution, we eliminate the need to move fingers from the home position on a keyboard. In addition to the obvious potential for higher data-entry speeds, such minimum motion keyboards appear likely to reduce to risk of repetitive stress injuries like carpel tunnel syndrome.

We evaluated minimum-motion versions of each of the three standard keyboard designs of Table 1 on four texts, including those of Section 4. The results are summarized in Table 5.

The minimum-motion QWERTY keyboard reconstructs over 98.7% of characters correctly on *Clinton*, the most modern and representative text. Similar recognition rates were achieved on the Dvorak and Maltron keyboards. We were surprised that minimum-motion QWERTY outperformed minimum-motion Dvorak on all four texts. The Dvorak layout did not perform as well because since similar consonants were often mapped to the same finger.

The Maltron layout achieved slightly better character-recognition rates than either the QWERTY or Dvorak layout. This is not a completely fair comparison, however, since the Maltron layout uses an additional finger (distinguishing between the left and right thumbs) for the letter ‘e’. Maltron’s character recognition rate (but not necessarily its word recognition rate) exceeded QWERTY on all texts. This difference is not sufficiently compelling to justify abandoning minimum-motion QWERTY, however.

Layout	Text	Char	Char %	Word %
QWERTY	Clinton	1084630	98.71%	96.29%
	Moby	1137518	96.43%	90.69%
	Bible	4007829	95.68%	89.29%
	Shakespeare	4645103	94.45%	87.69%
Dvorak	Clinton	1084630	98.40%	94.91%
	Moby	1137518	95.68%	88.20%
	Bible	4007829	95.32%	85.76%
	Shakespeare	4645103	94.25%	85.12%
Maltron	Clinton	1084630	98.82%	96.08%
	Moby	1137518	97.01%	91.44%
	Bible	4007829	96.85%	90.70%
	Shakespeare	4645103	95.40%	88.44%

Table 5: Results of minimum-motion keyboard with QWERTY, Dvorak and Maltron layouts.

Finally, we studied resolving the ambiguity inherent using the one-handed half-QWERTY keyboard [11]. In this layout, there are 2 letters on each key, plus a mode-switch control key to signify whether he/she is typing a left- or right-hand symbol. Through ambiguity resolution, we can dispense with the mode switch key, since our algorithm can accurately reconstruct the original text. In our experiments, we are able to correctly reconstruct 99.32% of characters for *Clinton* and 97.82% characters for the Moby Dick excerpt. That we achieved such high accuracies is not surprising, since the degree of overloading is less that of the previous keyboards we have discussed. The half-QWERTY layout uses 15 keys, each of which contains at most two letters, and some of which only contain one letter and a punctuation symbol.

6 Effect of Errors on Reading Comprehension

Character error rates provides a somewhat misleading measure of the fidelity of reconstruction, since intelligibility suffers substantially at seemingly low error rates. We performed a modest human-factors study to assess the impact of error rates on intelligibility.

Our methodology was as follows. We selected a brief passage and the associated reading comprehension questions from a sample TOEFL exam. We gave our exam to 139 students in computer science classes at SUNY Stony Brook in one of six versions:

- A perfect exam, taken straight from the TOEFL sample.
- A version of the exam which was reconstructed using the telephone keypad layout.
- Reconstructed texts using the bin packing layouts for 7,6,5, and 4 keys.

Our results are presented in Table 6, which shows that the number of errors and "can't determines" for each test. Our results show that reading comprehensibility drops off substantially when using 5 or fewer keys. In particular there is no significant decline in the percentage of exams with no mistakes or the average number of errors between the perfect exams and the exams encoded with 6 or more keystrokes.

	Orig	Phone	6-key	5-key	4-key	3-key
0 wrong	14	9	9	1	0	0
1 wrong	5	7	8	2	6	1
2 wrong	4	2	3	9	5	4
3 wrong	2	2	1	4	8	9
4 wrong	0	2	0	4	4	4
5 wrong	1	3	0	0	0	6
Can't	6	13	7	16	13	44
students	26	25	21	20	23	24
Avg errors	0.9	1.6	0.8	2.4	2.4	3.4

Table 6: Results of reading comprehension exam.

7 Implications for OCR Systems

Printing artifacts and other noise causes serious problems for optical character recognition (OCR) or handwriting recognition systems. Typically, such systems report such obscured characters as “unknown”, however more advanced systems use dictionaries to perform post-processing on the output stream. By guessing that the correct interpretation of the unknown characters yields a word in the dictionary, we can increase the recognition rate and compensate for these defects, at a cost of a certain amount of errors.

Conventional OCR post-processing is limited to character trigram frequencies and dictionary lookups. However, our reconstruction engine provides the possibility of exploring the impact which grammatical constraints can improve reconstruction. Indeed, our reconstruction engine has been employed in a research OCR system [14].

We conducted a series of experiments to measure to what extent context and grammatical constraints help in identifying characters left unrecognized by an OCR system. In our experiments, we randomly selected a given fraction of the words in the text, and from each randomly selected one character to be turned into an “unknown” character. To model the impact of different levels of image noise, we varied the fraction of affected words from 10% to 100% on 11 chapters of the novel *Herland*, consisting of about 53,000 words.

Our reconstruction rate exceeds 83% even when half of the words contain errors. As the error rate gets larger, the effect is to weaken the power of grammatical constraints since the part-of-speech of neighboring terms is more likely to become ambiguous. In the limiting case where 100% of the words contain single character errors, we reconstruct only 76% of these characters correctly.

The differential performance as the image error rate increases can only be attributed to the weakened power of grammatical constraints. For this reason, we are convinced that our engine provides more powerful error correction than employed in conventional OCR systems.

8 Conclusions and Future Work

In conclusion, our results suggest several interesting new ways to improve the power of text-oriented user interfaces. Further human factors studies are necessary to assess the practicality of the minimum-motion QWERTY keyboard, but we would advocate encorporating it as an optional mode in word-processing systems in order to properly assess user’s reaction to it.

Our results demonstrate that other applications, particularly for the severely disabled, we clearly achieve high enough accuracy to merit more widespread implementation.

We have also employed this reconstruction engine for back end processing in a "virtual keyboard", implemented using data gloves to measure physical hand movement and a head-mounted display to provide visual feedback to the user [4]. Since VR data gloves do not provide precise feedback of small finger-motions and multiple fingers move on each keystroke, ambiguity resolution is necessary to create practical virtual keyboards.

More speculatively, we believe that these ideas can impact text entry for languages such as Chinese, which use character sets with thousands of symbols. A common text entry technique [1, 6] for Chinese has the user type in a phonetic pronounciation of each character on a conventional western keyboard, with the user explicitly resolving the remaining ambiguity by selecting the character from a menu. Sufficient information exists to disambiguate the characters from the pronounciation, since this can be done by any native listener. We believe that much of the ambiguity can be resolved using our sentence reconstruction techniques.

References

[1] Kim Hang Chung. Conversion of chinese phonetic symbols to characters. In *MS Thesis, Hong Kong University of Science and Technology*, August 1993.

[2] J. J. Darragh, I. H. Witten, and M. L. James. The reactive keyboard: A predictive typing aid. *IEEE Computer*, 23(11):41–50, 1990.

[3] A. M. Derouault and B. Merialdo. Natural language modeling for phoneme-to-text transcription. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-8:742–749, 1986.

[4] F. Evans, S. Skiena, and A. Varshney. System and method for entering text in a virtual environment. U.S. Patent 6,407,679, 2002.

[5] D. Gopher and D. Raij. Typing with a two-handed chord keyboard: Will the QWERTY become obsolete? *IEEE Transactions on Systems, Man, and Cybernetics*, 18(4):601–609, July-August 1988.

[6] Loke Soo Hsu and Zhibiao Wu. Chinese character prediction by recurrent network. In *Computer Processing of Chinese and Oriental Languages*, pages 6(2):179–194, December 1992.

[7] J. J. Hull and S. N. Srihari. Experiments in text recognition with binary n-gram and Viterbi algorithms. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-4:520–530, 1982.

[8] A. Kaufman, A. Bandopadhay, and G. Piligian. Apparatus and method for eye tracking interface, November 1 1994. United States Patent Number 5,360,971.

[9] S. Litvak and R. Shamir. Efficient algorithms for constructing and employing variable-length markov models of language. Technical report, Dept. of Computer Science, Tel-Aviv University, Israel, 1996.

[10] Lillian Malt. The effect of keyboard layout on error rate and error patterns. Technical report, Malt Applied Systems, UK.

[11] E. Matias, I. MacKenzie, and W. Buxton. Half-QWERTY: A one-handed keyboard facilitating skill transfer from QWERTY. In *Proceedings of the INTERCHI '93 Conference on Human Factors in Computing Systems*, pages 88–94, 1993.

[12] M. D. McIlroy. Development of a spelling list. *IEEE Trans. Communications*, COM-30:91–99, 1982.

[13] H. Rau and S. Skiena. Dialing for documents: An experiment in information theory. *Journal of Visual Languages and Computing*, pages 79–95, 1996.

[14] G. Sazaklis, E. Arkin, J. Mitchell, and S. Skiena. Geometric decision trees for optical character recognition. In *Proc. 13th ACM Symp. Computational Geometry*, 1997.

[15] R. Shinghal and G. T. Toussaint. A bottom-up and top-down approach to using context in text recognition. *International Journal of Man-Machine Studies*, 11:201–212, 1979.

[16] R. Shinghal and G. T. Toussaint. Experiments in text recognition with the modified viterbi algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-l:184–193, 1979.

[17] R. Shinghal and G. T. Toussaint. The sensitivity of the modified viterbi algorithm to the source statistics. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-2:181–185, 1980.

[18] S. N. Srihari, J. Hull, and R. Choudhari. Integrating diverse knowledge sources in text recognition. *ACM Trans. Office Info. Sys.*, 1:68–87, 1983.

[19] M. A. Toleman, J. Welsh, and A. J. Chapman. An empirical investigation of menu design in language-based editors. *SIGSOFT Software Engineering Notes*, 17-5:41–46, 1992.

[20] Godfried T. Toussaint. The use of context in pattern recognition. *Pattern Recognition*, 10:189–204, 1978.