Dialing for Documents: An Experiment in Information Theory

Harald Rau*
Steven S. Skiena†

Department of Computer Science
State University of New York
Stony Brook, NY 11794-4400
skiena@sbcs.sunysb.edu

January 11, 1995

Abstract

Standard telephone keypads are labeled with letters of the alphabet, enabling users to enter textual data for a variety of possible applications. However, the overloading of three letters on a single key creates a potential ambiguity as to which character was intended, which must be resolved for unambiguous text entry. Existing systems all use pairs of keypresses to spell out single letters, but are extremely cumbersome and frustrating to use.

Instead, we propose single-stroke text entry on telephone keypads, with the ambiguity resolved by exploiting information-theoretic constraints. We develop algorithms capable of correctly identifying up to 99% of the characters in typical English text, sufficient for such applications as telephones for the hearing-impaired, E-mail without a terminal, and advanced voice-response systems.

KEYWORDS: telephone keypads, information theory, telephones for the hearing-impaired, Viterbi algorithm.

*Current address: Lehrstuhl fur Informatik III, Universitat Am Hubland, 97074 Wurzburg, Germany, rau@informatik.uni-wuerzburg.de

†Partially supported by NSF grant CCR-9109289 and ONR award 400x116yip01. Patent protection is being sought for this technology.

1 INTRODUCTION

The marriage of computing and telecommunications is spawning an amazing variety of new products and services. However, only a small fraction of the population seem available as potential users, because the typical citizen does not own an alphanumeric terminal and modem. Inertia and cost make it unlikely that such a national infrastructure will develop in the near future. However, in this paper we make the case that every home in the United States already contains a simple on-line terminal, which can be made usable only by exploiting information-theoretic constraints.

Anyone who has ever placed a telephone call is familiar with the fact that the 2-9 keys of a standard phone dial are each labeled with three letters of the alphabet (the letters Q and Z do not appear). These labels have been used to provide mnemonics to aid in memorizing phone numbers (such as 1-800-FLOWERS) and in modern messaging systems for entering single word strings (such as a database of employee names in a company switchboard application). However, the overloading of three letters on a single key creates a potential ambiguity as to which character was intended.

Resolving this overloading is necessary for unambiguous text entry. For applications involving a small, predetermined vocabulary (such as the switchboard application described above), a simple hashing scheme will suffice. However, the problem becomes considerably more difficult if arbitrary text is to be entered. Existing systems all use pairs of keypresses to spell out single letters, such as using the first key to specify a triple of letters and the second to select the correct one from the triple. These systems are extremely cumbersome and frustrating to use, whereas significantly increased typing speeds can be obtained with single keypresses.

Instead, we propose single-stroke text entry on telephone keypads, with the ambiguity resolved by exploiting information-theoretic constraints. We develop algorithms capable of correctly identifying up to 99% of the characters in typical English text, as shown in Table 1. Potential applications include:

- *Communication with the hearing-impaired* – with a computer-enhanced telephone, a hearing-impaired user could make and receive calls to a conventional telephone without the intervention of an outside operator. The hearing-impaired user would communicate via a computer screen, while the other user would type on the phone keypad and hear results dictated by a text-to-speech synthesizer.
- *E-mail without a terminal* – again, the phone keypad and a text-to-speech synthesizer facilitate arbitrary interaction with a computer, enabling the user to read and write E-mail from a conventional phone without additional hardware.
- *Enhanced voice-response systems* – conventional voice-response systems record certain user-interactions, which must be transcribed before they can be acted upon. Using our techniques, a user's name and address could be entered and reconstructed with sufficient fidelity that requested literature could be mailed immediately, without human intervention.

Our "lossy" reconstruction of English text is easily understandable, as shown by the Gettysburg Address below, with all reconstruction errors highlighted:

Text	characters	characters correct	non-blanks correct	words correct	time per character
Clinton Speeches	1073593	99.04%	98.86%	97.67%	0.97ms
Herland	278670	98.24%	97.89%	97.02%	0.97ms
Moby Dick	1123581	96.85%	96.25%	94.75%	1.14ms
Bible	3961684	96.20%	95.39%	95.39%	1.33ms
Shakespeare	4558202	95.20%	94.21%	92.86%	0.99ms

Table 1: Our telephone-code reconstruction system applied to various text samples.

FOURSCORE AND SEVEN YEARS AGO OUR FATHERS BROUGHT FORTH ON THIS CONTINENT A NEW NATION CONCEIVED IN LIBERTY AND DEDICATED TO THE PROPOSITION THAT ALL MEN ARE CREATED EQUAL. NOW WE ARE ENGAGED IN A GREAT CIVIL WAR TESTING WHETHER THAT NATION OR ANY NATION SO CONCEIVED AND SO DEDICATED CAN LONG ENDURE. WE ARE MET ON A GREAT BATTLEFIELD OF THAT **WAS**. WE HAVE COME TO DEDICATE A PORTION OF THAT FIELD AS A FINAL **SERVING** PLACE FOR THOSE WHO HERE **HAVE** THEIR LIVES THAT THE NATION MIGHT LIVE. IT IS ALTOGETHER FITTING AND PROPER THAT WE SHOULD DO THIS. BUT IN A LARGER SENSE WE CAN NOT DEDICATE WE CAN NOT CONSECRATE WE CAN NOT HALLOW THIS GROUND. THE BRAVE MEN LIVING AND DEAD WHO STRUGGLED HERE HAVE CONSECRATED IT FAR ABOVE OUR POOR POWER TO ADD OR DETRACT. THE WORLD WILL LITTLE NOTE NOR LONG REMEMBER WHAT WE SAY HERE BUT IT CAN NEVER FORGET WHAT THEY DID HERE. IT IS FOR US THE LIVING RATHER TO BE DEDICATED HERE TO THE UNFINISHED WORK WHICH THEY WHO FOUGHT HERE HAVE THUS FAR SO NOBLY ADVANCED. IT IS RATHER FOR US TO BE HERE DEDICATED TO THE GREAT TASK REMAINING BEFORE US THAT FROM THESE HONORED DEAD WE TAKE INCREASED DEVOTION TO THAT CAUSE FOR WHICH THEY HERE **HAVE** THE LAST FULL MEASURE OF DEVOTION THAT WE HERE HIGHLY RESOLVE THAT THESE DEAD SHALL NOT HAVE DIED IN VAIN THAT THIS NATION UNDER GOD SHALL HAVE A NEW BIRTH OF FREEDOM AND THAT GOVERNMENT OF THE PEOPLE BY THE PEOPLE FOR THE PEOPLE SHALL NOT PERISH FROM THE EARTH.

Below, we discuss previous work on entering text via telephone keypads. We then use information theory to demonstrate that one keypress per character suffices to reconstruct English text. Finally, we describe our algorithms to do the reconstruction and demonstrate their effectiveness. We also present the results of modest human-factor experiments to show user acceptance of our approach.

2 PREVIOUS WORK

Human-factor issues associated with telephones have been extensively studied. The most relevant work concerns conventions for text entry, human factors on the use of telephone keypads, and telephones for the hearing-impaired.

A variety of different conventions have been proposed for text entry with telephone keypads, to resolve the ambiguity inherent in overloaded characters. Flanagan's convention [9] presses the given key k times, if the given character labeled in the kth position on the key. Therefore 'L' would be represented *5-5-5-0*, with 0 an escape key to denote the end of the character. Hashimoto [14] proposes using repeated presses of

$\boxed{\#}$ to distinguish between overloaded characters. Dargan [4] suggests using Boolean exclusion questions to resolve multiple interpretations of a sequence of phone codes. Several inventors, including [8, 17, 24] uses statistical data and/or dictionaries to resolve ambiguities. This is in the spirit of our work, however, it does not address the grammatical constraints which are crucial to the success of our system. These methods for entering alphabetic characters using the telephone keypad with a multiple keypress-per-letter have been assessed by [19] and [7]. Intensive user instruction is required to avoid high input-error rates. Davis [5] proposes an approach similar to ours, but without as extensive an analysis.

Our system permits text entry using one keypress per letter. One necessary design decision concerned the characters (Q and Z) that usually do not appear on the keypad (although the AT&T 7506 has 'PQRS' and 'WXYZ' on single keys). Several different conventions have been used here as well. For example, Tsakanikas [27] proposed the '1' key to represent Q and Z, with '0' as a space. Marics [20] found that users preferred to type '*' to represent Q or Z, although many subjects instead chose the key which was labeled with the alphabetic neighbors of the missing characters. We adopted Marics' prefered convention for our work.

Clearly, physical characteristics of the telephone and the layout of the telephone keypad has a significant effect on user performance and speed. Deininger [6] showed that the keying error rate varied between 1.3% and 7.1% with different key layouts. The key label layout and force-displacement characteristics of the telephone also have significant influence on the user performance. Error rates can be reduced with adequate training and instructional materials. Marics [20] showed that speed increases significantly after a short period of training. Kramer [19] recommends two hours of training to reduce error rates and suggests that a feedback mechanism might improve the keying accuracy, such as a tone that confirms the keypress. In our text-entry application, we propose that synthesized speech be used to enable the sender of the message to confirm a sentence decoding or to choose alternatives.

Finally, we note that several patents have been issued for telephones for the hearing-impaired, including [2, 3, 10, 18, 22, 23, 28].

3 INFORMATION-THEORETIC BOUNDS

Before attempting to reconstruct English text from single-key phone codes, we must establish that there is enough information available to make such a reconstruction possible. Formally, an information source S with a repository of possible messages M_k, $1 \leq k \leq n$ selects one message for transmission. Information theory defines the information content or *source entropy* $H(S)$ of a set of messages in terms of the probability $P(M_i)$ that this message is chosen by the source, as the average value of information associated with each of the messages:

$$H(S) = -\sum_{i=1}^{n} P(M_i) \log_2 P(M_i)$$

The source entropy $H(S)$ denotes the average information content conveyed per message, in units of bits. The source entropy reaches its maximum for the case of equally likely messages, when $H(S) = \log_2 n$.

In our application, the message source is the English text itself, and measuring the true entropy of English is a non-trivial problem [1]. If one imagines the message source to consist of 26 letters, the space and the period, each of which are used with equal probability, then the source entropy is $\log_2 28 = 4.81$ bits per letter. Since the information content which can be transmitted by using the phone keypad is at most $\log_2 10 = 3.32$ bits per keystroke, there remains a considerable deficit which must be overcome.

English, however, does not use all letters with an equal probability. Using the probabilities from the Brown Corpus, the source entropy of first-order English drops to 4.16 bits per letter. However, standard assignment of letters to phone keys is not optimally balanced (as shown in Table 2), and lowers the transmitted entropy to approximately 3.07 bits per letter, leaving a deficit of over one bit per symbol.

However, a better measure of the entropy of English comes from considering entire words or sentences as messages, rather than the individual letters. On a word-by-word basis, estimates of the entropy of English drop to below 2.2 bits per letter [25]. In a series of experiments on full English text, Shannon obtained an estimate of the entropy of English as between 0.6 and 1.3 bits per letter.

Since the information capacity of telephone keycodes exceeds Shannon's estimate of entropy, in principle it is possible to accurately reconstruct the source text by exploiting context and other linguistical constraints.

4 RECONSTRUCTION ALGORITHMS

In this section, we describe our algorithms for performing telephone code reconstruction. The system takes as input a stream of digits, and outputs a maximum-likelihood reconstructed English source text.

We assume the following input convention. To supplement the traditionally-labeled keys 2-8, we use the asterisk $\boxed{*}$ to substitute for the letters `Q`, `Z` and ␢, the blank character. The pound key $\boxed{\#}$ serves as sentence delimiter. The $\boxed{0}$ and the $\boxed{1}$ key have no meaning and could be used for additional features. Hyphens and other punctuation should be omitted or substituted by blank characters. Ordinal and cardinal numbers are entered by spelling the numbers out.

Our convention ends each sentence with a #, and so our reconstruction proceeds on a sentence-by-sentence basis in four phases:

Recognizing Blanks: The first phase partitions the input stream into tokens by recognizing the blank characters, and is rendered non-trivial because of overloading the $\boxed{*}$ key with Q and Z. Since blanks are much more common than `Q` and `Z`, simply marking *all* asterisks as blank would be correct 98.99% of the time. However, we can do a better job of identifying the real blanks.

We base the decision of whether an asterisk represents a blank on the character context of the asterisk. A window of three characters preceding and following

Letter	*Frequencies*	*Key*	*Frequencies*
a	6.70%		
b	1.26%	2 ABC	10.57%
c	2.62%		
d	3.12%		
e	10.29%	3 DEF	15.35%
f	1.94%		
g	1.58%		
h	4.34%	4 GHI	11.99%
i	6.07%		
j	0.16%		
k	0.51%	5 JKL	4.09%
l	3.42%		
m	2.13%		
n	5.92%	6 MNO	14.34%
o	6.30%		
p	1.69%		
q	0.09%	7 PRS	12.40%
r	5.20%		
s	5.51%		
t	7.65%	8 TUV	10.71%
u	2.20%		
v	0.86%		
w	1.49%	9 WXY	3.07%
x	0.16%		
y	1.42%		
z	0.08%	* QZ$\not{b}$	16.70%
$\not{b}$	16.53		
.	0.78%	# .	0.78%

Table 2: obtained by examining the Brown Corpus. By our convention `Q`, `Z` and the space are entered by $*$ and the period is indicated by $\#$.

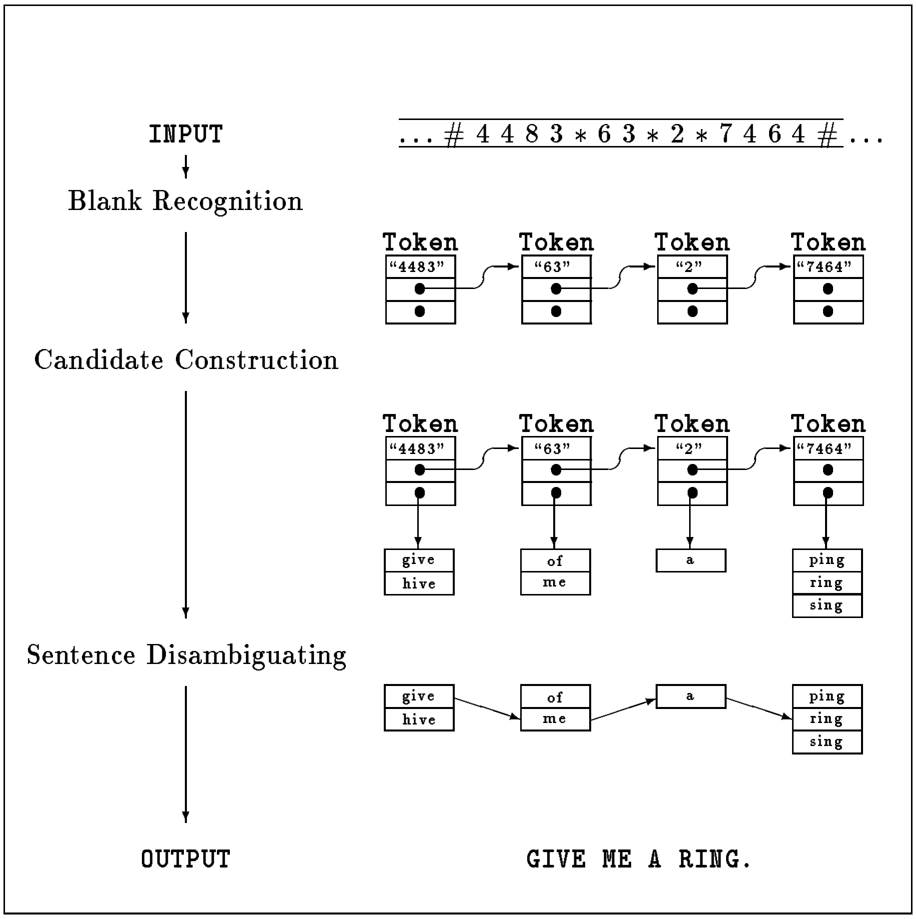

Figure 1: The phases of the reconstruction process

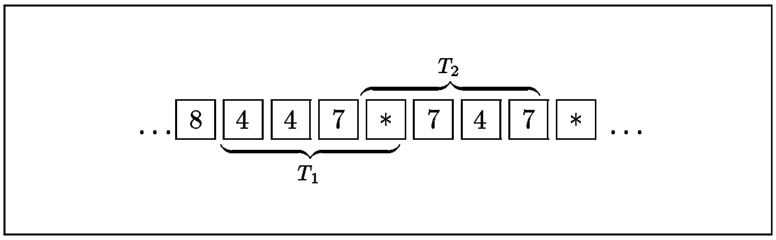

Figure 2: Pre- and post-asterisk-trigrams for identifying blanks.

the asterisk is considered as context, as in Figure 2. These pre-asterisk and post-asterisk trigrams are composed of phone codes, resulting in 10^3 possible combinations for each trigram. For each trigram, we tabulate how often the asterisk was intended as Q, Z, or blank, yielding six quantities:

- T_1^c: How often did the letter $c \in \{\texttt{Q}, \texttt{Z}, \text{b}\!\!/\}$ occur as the fourth character of the pre-asterisk trigram?
- T_2^c: How often did the letter $c \in \{\texttt{Q}, \texttt{Z}, \text{b}\!\!/\}$ occur as the first character of the post-asterisk trigram?

We define the blank confidence:

$$C(\boxed{*} \leftarrow \text{b}\!\!/) = \frac{T_1^{\text{b}\!\!/} + T_2^{\text{b}\!\!/}}{\sum_{c=\texttt{Q},\texttt{Z},\text{b}\!\!/}(T_1^c + T_2^c)}$$

The asterisks which are classified with high confidence as b/ are now so labeled, while other asterisks remain unidentified until later phases of our algorithm. In setting the confidence threshold $C(\boxed{*} \leftarrow \text{b}\!\!/) = 1.0$, we find 81.63% of all asterisks are marked as b/, while making no mistakes. Using lower confidence thresholds to classify the asterisks significantly increases the number of blanks correctly marked, at a cost of a small number of errors. The best accuracy for reconstructing asterisks is obtained for a marking threshold of 99.997%. The overall blank recognition accuracy is then 99.91%.

Construction of Word Candidate Sets: Each token is labeled with a set of possible character candidates, using a dictionary of English words which hash to the key codes of the token.

Unknown Word Analysis: If no dictionary entry exists for the token, a character interpretation is constructed using partial dictionary matches, affix/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. A dynamic programming algorithm is applied to eliminate candidates which are unlikely because of word-pair frequencies and grammatical constraints, and returns the most likely word sequence.

Each phase of the system is discussed below in detail.

4.1 Dictionary Matching

The blank recognition process partitions the input stream into sequences of code strings. Each code sequence represents a small number of consecutive English words, separated by the remaining unlabeled asterisks.

The dictionary matching phase seeks to map each code sequence with a unique word in our dictionary of nearly 50,000 words. Most of the words in the dictionary map to

Number of coinciding codes	*Words*	*Code Strings*
1	41122	41122
2	4896	2448
3	1476	492
4	588	147
5	300	60
6	174	58
7	42	6
8	48	6
9	9	1
10	0	0
11	11	1

Table 3: Mapping English words into phone codes causes collisions.

unique phone code sequences, however, approximately 16% of the phone codes collide with other dictionary words. The distribution of collisions is shown in Table 3. As an extreme case, the code string "22737" collides with eleven distinct English words, as shown in Table 4.

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 [21], enhanced by other common words. Abbreviations such as BMW and ACM were removed, yielding a dictionary with 53324 words hashed to 44474 code entries. Our lookup succeeds for the chosen dictionary for about 80% of all tokens.

For the part of tokens that can not be resolved in the previous phase a series of further reconstruction attempts is made. If the dictionary lookup fails, one reason might be that the code contains more than one English word. Thus an attempt is made to find a dictionary entry for a segment of the code which is delimited by an asterisk and starts at character position 0. If the attempt succeeds the code string is split at the asterisk position and a new token is created for the rest of the string. Single letters are not split off. For the new token with the code remainder the process restarts. Many tokens contain only two English words, and would be found in the dictionary had the separating asterisk had been interpreted as blank.

If the split fails, the code string is checked for affixes from Table 5. Suffixes and prefixes which are found in the code are stripped off, and the remaining string is looked up in the dictionary. If multiple entries exist, the most frequent entry is used. If no entry is found, the affix is discarded and the code is passed to the next subphase.

Code	*Word Candidates*	*Count*
22737	cases cares cards capes caper bases baser bares barer bards	11
46637	inner hoofs hoods hones honer homes homer goods goner	9
269	coy cox cow boy box bow any amy	8
76737	sores sorer roses ropes roper poses poser pores	8
7283	scud save sate rave rate pave pate Saud	8
729	say sax saw ray raw pay pax paw	8
72837	saves saver sates raves rates rater paves pater	8
2273	case care card cape base bare bard acre	8
7867	suns sums stop runs rump puns pump	7
752837	slaves slaver slates slater skates skater plates	7
4663	hoof hood hone home goof good gone	7

Table 4: The selection of code strings with the greatest number of interpretations

Prefix List				
afro-	ever-	iso-	negro-	stereo-
all-	ex-	kilo-	non-	sub-
anti-	fast-	last-	out-	super-
auto-	fore-	less-	over-	supra-
best-	geo-	long-	photo-	tele-
bio-	half-	magneto-	poly-	thermo-
contra-	high-	meta-	pre-	trans-
counter-	hyper-	micro-	pro-	ultra-
cross-	ill-	mid-	pseudo-	un-
dis-	in-	milli-	psycho-	under-
double-	inter-	mis-	re-	well-
electro-	intra-	mono-	self-	
en-	ir-	multi-	semi-	
Suffix List				
-ability	-ed	-illy	-ity	-ment
-able	-es	-ily	-ivism	-ness
-ably	-hood	-ing	-ize	-s
-al	-ibility	-ism	-ized	-top
-aly	-ic	-ist	-led	
-ally	-ies	-istic	-less	
-ary	-ific	-ists	-like	
-d	-ify	-ities	-ly	

Table 5: The affixes applied to construct a dictionary candidate for a code string. It is an enhanced version of [21].

4.1.1 Recursive Match

For words which are completely unknown to the dictionary, such as proper names and foreign terms, a recursive match procedure is used. The decode suggestion is assembled from partial matches of the code and dictionary entries. The final candidate is chosen from all possible matches according to the following preference list:

1. The candidate with the most reconstructed characters wins.
2. The candidate with the fewest partial matches wins.
3. The candidate with the most preserved word boundaries wins, ie. asterisks which turn to blanks.

This match routine is not able to reconstruct certain kinds of words that can not be constructed by English words, such as foreign names. To catch these cases a rejecting rule was applied – a string is accepted if it consists of at most two more partial matches than the number of contained blanks. Otherwise it is rejected.

For the remaining unknown words a dynamic programming algorithm was applied. A second-order transitional probability model was used, based on the probability of observing letter L_i given previous letters $L_{i-2}L_{i-1}$, as estimated using the Brown Corpus. The Viterbi algorithm is a dynamic programming algorithm for finding a global optimum for the given probabilities. It will be further described below, applied to words of a sentence rather than letters of a word.

4.2 Sentence Disambiguating

The input to this final phase is a list of tokens corresponding to a sentence of the text input. Each token contains a list of possible interpretations for its code string. Depending on the size of these interpretation lists, there may be an exponential number of possible sentence candidates. Grammatical constraints can be employed within the Viterbi algorithm to eliminate unlikely or implausible candidates, to find the most likely sentence for the phone code string.

4.2.1 The Viterbi Algorithm

Let $X = X_0X_1 \cdots X_mX_{m+1}$ denote a given sequence of phone code tokens, where the X_i are the isolated word codes and X_0 and X_{m+1} are the sentence delimiters. Further, let $Z = Z_0Z_1 \cdots Z_mZ_{m+1}$ be an English sentence consistent with X, where Z_0 and Z_{m+1} are periods. The Viterbi algorithm seeks the interpretation Z that maximizes $P(Z/X)$, the probability that Z caused X. It takes into account:

$P(Z_i/X_i)$, 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 or *trellis* represents the word lattice that forms the set of sentence candidates. Each of the nodes and edges have costs assigned with them. The node

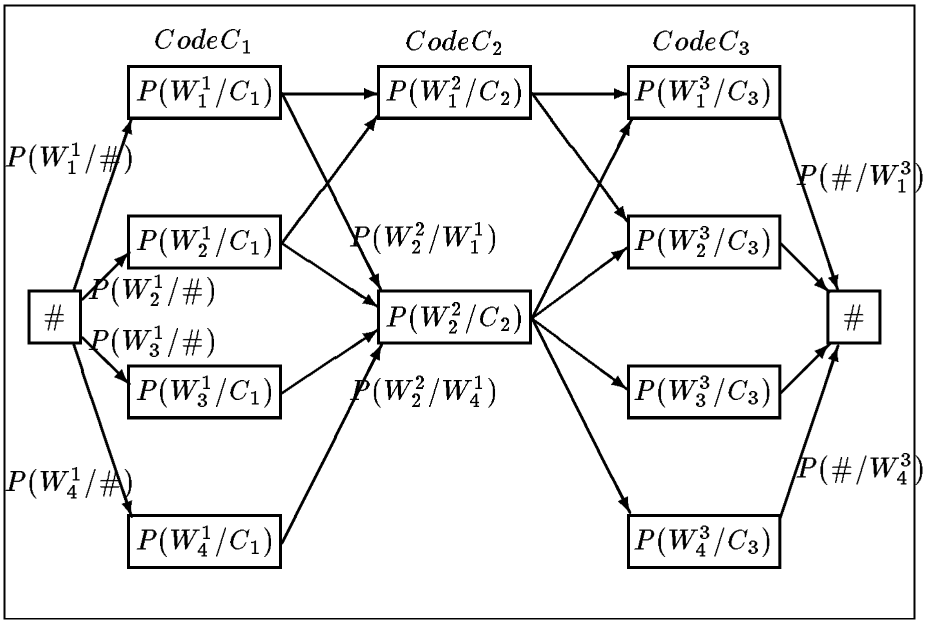

Figure 3: The Viterbi algorithm finds the maximum-cost path through a trellis

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 [26] and [16] for detailed descriptions of the Viterbi algorithm.

To compute the word transition probabilities, we examined all word pairs in the Brown Corpus. Each entry in the dictionary contains the list of words which follow it, and the corresponding transitional probabilities. To account for word pairs which did not occur in the corpus, we use

$$P(W_{i+1}/W_i) := 1/\text{count}_{W_i}^2$$

to estimate the probability of word W_{i+1} given W_i, which itself occurred count_{W_i} times in the corpus. Since the transition probability is at least

$$P(W'_{i+1}/W_i) = 1/\text{count}_{W_i}$$

if word pair $W'_{i+1}W_i$ occurres in the corpus, this gives preference to the pair which actually appeared. However, W'_{i+1} might have a larger stand-alone frequency than W_{i+1}, and thus lead to a more costly path and hence better interpretation.

In our dictionary implementation, 54.22% of all dictionary entries carry a word-pair list. The average length of this list is 8.52 entries.

List of Word Tags		
Tag	Description	Examples
ABL	pre-qualifier	*quite, rather*
ABN	pre-quantifier	*half, all*
AP	post-determiner	*many, next*
AT	article	*a, the, no*
CC	coordination conjunction	*and, or*
DT	singular determiner	*this, that*
DTS	plural determiner	*these, those*
JJ	adjective	*compellable*
NN	sing. or mass noun	*powder*
NNS	plural noun	*drafts*
NR	adverbial noun	*home, west*
VB	verb, base form	*take*
VBD	verb, past tense	*took*
VBG	verb, present participle, gerund	*taking*
VBN	verb, past participle	*taken*
VBZ	verb, 3rd singular present	*takes*
WDT	wh- determiner	*what,which*

Table 6: A selection of word tags used in the Brown Corpus. Different tags might be combined to specify the category of a words.

4.2.2 Grammatical Word Categories

The Viterbi algorithm takes word and word-pair frequencies into account, but no grammar rules are considered. Thus the selected sentence may not be grammatically plausible. To ensure implausible sentences are rejected, each word in the dictionary is augmented with a tag that indicates the corresponding grammatical category.

The Brown corpus contains about 30 different word tags, which again might be combined with each other for word categories (see [11]). Examples are shown in Table 6. A total of 132 combinations were used, each of which became a word category in the underlying dictionary. The words that appear with more than one category became multiple entries in the dictionary each with a different tag and with its own frequency count. Word-pair frequencies were collected for the words in regard to their tags.

A corpus was created solely of tags and the period. From that collection of sentences, bigram (word pair) frequencies for the tags were collected. From that data, transitional probabilities for the tags were derived, to be taken into account by the Viterbi algorithm as additional edge costs.

The word and word-pair frequency costs of the standard Viterbi algorithm must be combined with the tagged costs. For some constant $0 \leq c \leq 1$,

$$\mathrm{Cost}_{Path} = (1 - c) \cdot \mathrm{Cost}_{stand} + c \cdot \mathrm{Cost}_{tags}$$

trades off the grammatical categories of the word nodes with their transitional probabilities. After experimentation, we found $c = 0.4$ gave the best results.

The Brown corpus is the source for our word categories. Unfortunately, our augmented dictionary contained words which did not appear in the corpus, and were hence untagged, effecting performance. To reduce the amount of words without tags, some tagging 'by hand' was performed on the dictionary.

4.3 Output Analysis

We give a performance analysis of the reconstruction system in Table 7, we describe the percentage of characters marked, errors made, and time spent for each phase of the algorithm. 71% of our total errors were made in reconstructing words not in the dictionary, although we correctly reconstructed over 87% of the characters in such words. Most of the computing time is consumed during the sentence disambiguating phase. The procedure profiles were monitored with `gprof`.

Several text samples were encoded by the telephone keypad code and reconstructed by the presented system. These samples were passed through a filter to spell out ordinal and cardinal numbers which occurred in the texts. The system does remarkably well, correctly decoding more than 95% of all characters in all samples, and greater than 99% on the Clinton speeches which is most representative of modern English.

Table 1 gives an analysis of the effectiveness of the different phases of our algorithm. The unknown-word analysis accounts for the largest share of our errors, which is not surprising since none of these words occur in our dictionary. Still, we correctly identify over 87% of the characters in these words. Sentence disambiguation (ie. selection of the correct dictionary word in cases of ambiguity) accounts for most of the computing time and almost all of the remaining errors, however our accuracy is still remarkably high. We do not believe significantly higher accuracy rates are possible without actual comprehension of the text, which is clearly an unsolved and possibly unsolvable problem.

5 HUMAN FACTORS AND USER ACCEPTANCE

The conventional two-stroke method to enter text on a telephone is extremely painful to the user. Our one-stroke method is clearly more comfortable, but we conducted a modest human-factor experiment to assess keying accuracy, speed, and the user's discomfort in using a telephone keypad to enter text.

Subjects – 13 subjects, 2 female and 11 males, participated in the experiment. All subjects can be rated as "intermediate" to "expert" computer users. All subjects were graduate students of either Computer Science, Electrical Engineering or the Business School.

Material – The subjects were asked to type three sentences (see Table 8) using the one keystroke per letter method. The second and the third sentence contained one Q each. No other special character was used. The subjects entered the sentences on

Phase	Characters identified	Errors of total	Accuracy	Time
Total	3155498	29875	99.05%	1.03 ms/char
Blank				
Recognition	15.82%	0.45%	99.97%	12.88%
Unknown Word	5.31%	71.28%	87.28%	0.74%
Split	0.64%	0.57%	99.15%	0.03%
Affix	1.58%	12.04%	92.79%	0.07%
Match	2.41%	37.12%	85.41%	0.35%
Viterbi	0.68%	21.55%	70.03%	0.29%
Sentence Disambiguation	78.88%	28.27%	99.66%	86.11%

Table 7: Performance analysis of the reconstruction algorithm.

S1: This is my first sentence on this device.
S2: Excessive bail shall not be required nor excessive fines imposed nor cruel and unusual punishments inflicted.
S3: Fourscore and seven years ago our fathers brought forth on this continent a new nation conceived in liberty and dedicated to the proposition that all men are created equal.

Table 8: Test sentences for human-factor experiments.

a TELEMAX desk set with a touch-tone keypad. The telephone was connected to a DTMF-ASCII converter, which in turn was connected to a computer. For each typing session the key pressed and the cumulative time for each sentence were recorded. Unfortunately, the telephone keypad used in the experiment was not ideal, with medium force-displacement characteristics and small letters located beside the number. Subjects complained about the keypad design when using it. Using a standard keypad might improve performance substantially.

Procedure – The subjects were introduced to the keying conventions and asked to type the sentences. Each sentence was displayed on the computer screen. The entered keys were displayed as feedback on the same screen. To terminate a sentence the subjects were asked to type the # key twice. Only one sentence was displayed at a time. The entry time was recorded for each sentence and was measured between the first keystroke and the delimiter keystroke. When the subjects were finished, they were asked *"How did you like using the telephone as alphabetic input device?"*, grading on a scale from 1 to 4 as follows:

	Character				*Word*			*Blank*	
	Ins	Sub	Omi	Rev	Ins	Sub	Omi	Ins	Omi
S1	2	1	4	0	0	1	1	0	1
S2	2	1	9	1	0	1	1	1	0
S3	4	8	15	2	1	0	3	1	5

Table 9: Typing Errors for each Sentence

	Characters	Correct	Worst	Entry time	Worst	Comfort
S1	43	98.6%	95.3%	1.78s	3.53s	1.77
S2	109	99.0%	94.5%	1.75s	2.78s	2.31
S3	173	98.4%	94.2%	1.55s	2.22s	3.15

Table 10: Experimental Results: accuracy of entered text, typing time per character, and average subject comfort.

(1) "it was fun"
(2) "it was ok"
(3) "somehow annoying"
(4) "really annoying"

To evaluate the subjects on accuracy, we classified typing errors into the following cases: (1) substitution errors – where a character is erroneously substituted for another; (2) insertion errors – where a character is erroneously inserted; (3) substitution errors – where a character is erroneously substituted for another; (4) omission errors – where a character is erroneously omitted; and (5) reversal errors – where the positions of adjacent characters are erroneously interchanged.

Table 9 summarizes the raw data. The error classes were also applied for entire words. Word errors are basically caused by cognitive factors rather than just by mistyping. Table 10 shows that benefits of experience already appear after typing just one sentence. Although subjects were told how to type special characters before starting with the first sentence, still 8 subjects asked how to type Q while typing the second sentence. In one case Q was entered by typing the # key. In two cases a different letter was entered as asterisk. Five subjects typed asterisks before terminating the sentence with the # key. One 'o' was entered as 0 (zero). This behavior was noted by [20] as well.

The recorded typing discomfort increased with sentence length. Fortunately, in the proposed applications it seems likely that comparatively short sentences are chosen. One subject typed "machine" instead of "device" in the first sentence, demonstrating that they were typing the sentence from memory. These errors in our experiment are irrelevant in practice, since our proposed applications require interaction instead of transcription.

The character error-rates recorded in our experiment were not insignificant, although they will decrease with experience. Still, they seem frequent enough to potentially reduce the accuracy of our reconstruction algorithm. Dealing with such errors is a logical focus of future work.

6 CONCLUSIONS

In conclusion, we have presented a telephone code reconstruction system based on word-wordpair frequency analysis and grammatical/linguistic constraints. It has been trained on a large corpus of English text, and reconstructs text with remarkable accuracy. We believe our technique significantly more powerful than any other which have appeared in the literature, and believe it can be used to in a variety of applications.

We believe that our information-theoretic approach can be fruitfully applied a wide variety of user interfaces, indeed any interface where multiply operations must be performed to specify a single action. This is the case in Chord [12] and other special keyboards. Similar problems [13, 15] occur in word processing systems for languages with large character sets, like Chinese and Japanese, for multiple keystrokes are necessary to specify each character, making typing very inconvenient. Finally, reducing entry complexity is an important goal in keyboards for with severe mobility limitations, like quadrapaligics. In all of these applications, a powerful context-based prediction engine would result in significantly more powerful interfaces.

References

[1] T. Bell, J. Cleary, and I. Witten. *Text Compression.* Prentice Hall, Englewood Cliffs, 1990.

[2] A. Chieffo. Telephone communication system for the deaf. US Patent 3,585,303, June 15, 1971.

[3] G. Coles. Telephone-coupled visual alphanumeric communication device for deaf persons. US Patent 4,191,854, March 4, 1994.

[4] J. Dargan. Telephonic data access and transmission system. US Patent 4,677,659, June 30, 1987.

[5] J. R. Davis. Let your fingers do the spelling: Disambiguating words spelled with the telephone keypad. *AVIOS Journal*, 9:57–66, March 1991.

[6] R. L. Deininger. Human factors engineering studies of the design and use of pushbutton telephone sets. *Bell System Tech. Journal*, 39:995–1012, 1960.

[7] M. C. Detweiler, R. M. Schumacher, and N. L. Gattuso. Alphabetic input on a telephone keypad. In *Proc. 34th Annual Meeting Human Factors Society*, 1990.

[8] R. Feinson. Interpretive tone telecommunication method and apparatus. US Patent 4,754,474, June 28, 1988.

[9] J. Flanagan, J. Kronmeyer, and R. Nelson. Communications system for alphanumeric information employing audio tone signalling. US Patent 3,675,513, July 11, 1972.

[10] S. Fowler and A. Brown. Telecommunications device for the hearing impared. US Patent 4,608,457, August 26, 1986.

[11] W. N. Francis. *Frequency analysis of English usage: lexicon and grammar*. Houghton Mifflin, Boston, 1982.

[12] 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.

[13] Kim hang Chung. Conversion of chinese phonetic symbols to characters. MS Thesis, Hong Kong University of Science and Technology, August 1993.

[14] K. Hashimoto. Push button phone capable of sending to the telephone line alphabetical characters. US Patent 4,918,721, April 17, 1990.

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

[16] 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.

[17] L. James. Processor-assisted communication system using tone-generating telephones. US Patent 4,650,927, March 17, 1987.

[18] G. Kondraske and A. Shennib. Character pattern-recognition and communications apparatus. US Patent 4,674,112, June 16, 1987.

[19] J. J. Kramer. Human factors problems in the use of pushbutton telephones for data entry. In *International Symposium On Human Factors In Telephony*, Berlin, 1970. VDE-Verlag GmbH.

[20] M. A. Marics. How do you enter "d'anzi-quist" using a telephone keypad? In *Proc. 34th Annual Meeting Human Factors Society*, 1990.

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

[22] J. Meyer. Telephone communications system for the deaf. US Patent 4,012,599, March 15, 1977.

[23] D. Nielson, D. Fylstra, and R. Stehle. Portable telephone communication device for the hearing impaired. US Patent 4,268,721, May 19, 1981.

[24] B. Riskin. Method and apparatus for identifying words entered on DTMF pushbuttons. US Patent 5,031,206, July 9, 1991.

[25] C. E. Shannon. Prediction and entropy of printed english. *Bell System Technical Journal*, 30:50–64, 1951.

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

[27] P. Tsakanikas. Telephonic alphanumeric data transmission system. US Patent 4,427,848, January 24, 1994.

[28] E. Underkoffler. Telephone communications device for the hearing-impared person. US Patent 4,426,555, January 17, 1984.