Let Your Fingers do the Spelling: Implicit disambiguation of words spelled with the telephone keypad

James Raymond Davis
The Media Laboratory
Massachusetts Institute of Technology
E15-325
Cambridge, MA 02139
jrd@media-lab.media.mit.edu

Abstract

One way to enter words into an interactive computer system is to spell them with the letters on a telephone keypad. Although each button has three letters, the system designer can often supply the system with enough additional information that it can select the intended word without additional input from the user. This is called implicit disambiguation. This paper examines the obstacles to implicit disambiguation and describes two different kinds of knowledge that can make it possible.

Introduction

Designers of telephone-based computer systems for voice interaction often provide the user with the ability to choose among several alternatives. If the list of alternatives is both small and known in advance (e.g., just the words "yes" or "no") then this ability can be provided through speech recognition. If not, another possibility is to use the twelve push buttons on the telephone keypad. There are several ways of using the keypad. One of them, applicable when the object to be selected has a name, is to spell the name using the letters on the keypad.

A frequently raised objection to such "telephone spelling" is that each button has three different letters on it. Some kind of DISAMBIGUATION is required in order to know which letter is intended. (A second problem, discussed below, is that the letters 'Q' and 'Z', as well as punctuation, are missing.)

Many system designers have chosen EXPLICIT disambiguation, which requires the user to do something extra to tell the system which letter is intended. There are two main ways of doing this. In the first, the user presses a key once, twice, or three times times to select one of the three letters on it (for example, to select 'B' one presses the '2' key twice). This method is used by both experimental systems (e.g., [3]) and implemented services (e.g. the BayBanks Brokerage "voiceQuote" service, certain NEC cellular phones). It has the disadvantage that it requires an explicit delimiter between successive letters when they appear on the same button. The other means of explicit disambiguation is to use one or more keys as "shift" keys, e.g. buttons 1, 2, and 3 select the first, second or third letter on the following key. Both of these alternatives are discussed in [12], and [2] reports on the speed and accuracy of such systems. Neither one is very attractive, since both are unnatural. Nobody could use such a system without first being told how.

Sometimes it is possible to use IMPLICIT DISAMBIGUATION, where the system figures out for itself which letter is intended. Even though a *single* button press is ambiguous, a *sequence* of button pushes need not be ambiguous. Since a given button has three letters, it might seem impossible to know which one is intended. Indeed, if there is no constraint on the possible inputs, that is, if every letter is equally possible in every position, then this is so. From the standpoint of information theory, a single button press contributes about 3.2 bits (if nine keys are used), and to select a single letter from the set of 26 requires 4.7 bits. But additional knowledge about the possible choices, if available, can substitute for the "missing bits". For instance, if the input is known to be a word from English, fewer bits should be required, since a single letter in English is estimated to contribute only about 2.3 bits[10].

Another way to look at disambiguation is as a coding problem. Consider the eight digits (2 to 9) as a reduced alphabet in which words are to be re-written. For example, the word "the" becomes "843" in this alphabet. Even though the single button press for '8' is ambiguous, the string '843' is not ambiguous if no other word has the same spelling in the reduced alphabet. When two words have the same spelling in the reduced alphabet they are said to COLLIDE. When a collision occurs, the system will need some kind of further disambiguation to decide which word was intended, but on the word level (which word does this string stand for) rather than the letter level (which letter does this digit stand for).

A second problem with telephone spelling (in addition to ambiguity) is the problem of the missing letters. The letters 'Q' and 'Z' do not appear on the telephone keypad. In the word described in this paper, the letter 'Q' is on the key marked 7 (where it fits, between P and RS), and 'Z' is on 9 (after WXY). Both are also assigned to the '1' key, because that is a plausible guess that the user might make, and also because several other systems do that, too. Systems should be useable without documentation or prompting, therefore they should be designed to do what a (reasonable) user would guess they should do. The design proposed here is likely to be only partially successful, because a recent study by Marics[7] shows that naive users show no consistent pattern of usage — indeed, many subjects guessed that these characters would be on the "star" key. (Marics makes some valuable suggestions for how systems should cope with these guesses, but to consider them here would take us beyond the scope of this paper.)

This paper discusses two experiments on implicit disambiguation of words. In each, an additional knowledge source is provided to the computer so it can (attempt to) perform implicit disambiguation. In the first, the knowledge source is a list of all the possible inputs, in the second, it is statistical knowledge about letter combinations in English.

Using word lists for disambiguation

One way to add knowledge to a system is to supply it with a list of all valid inputs. For most applications, this is no problem at all, since such a list will usually already be available. One early use of word lists for implicit disambiguation is an experimental directory assistance system built at Bell Labs in 1973 [8]. (This system used first and middle initials and state in addition to last name.) Word list disambiguation was also used in the Speech Filing System [4], the Olympic Message System [5], Direction Assistance [1], and an experimental directory assistance system built by Reich [9].

The effectiveness of word list disambiguation is determined by the number and size of collisions, since collisions will most likely require additional user action. Table 1 shows the number of collisions for six different word lists. (The source files are the names of Boston area streets, an otherwise unidentified list of last names, the names of all MIT students in 1989, all words in the Brown text corpus, a dictionary discussed by Witten (in [12] page 249), and Webster's Seventh Dictionary.)

Database	size	average letters per word	number of collisions	percent of words colliding
Boston streets	2433	7.4	79	3.2
job applicant names	6138	6.9	365	5.9
student names	14,246	6.8	1622	11.4
Brown words	19,837	6.3	2549	12.8
Witten word list	24,500	7.3	2000	8.0
Webster dictionary	48,102	8.1	5289	11.0

Table 1: Number of collisions for seven databases

What factors affect the number of collisions? Clearly, the percentage of collisions rises as the database size increases. There is also an effect due to length. As words become longer, the likelihood of a collision decreases, since a collision requires that every letter of two words have the same digit. Word lists with longer words are less likely to collide. For example, words in the Brown corpus range in length from one to 19 letters. Words of length four or less are only 12% of the total words, but account for 57% of all collisions.

Consider now the size of the collisions. For the Boston street name database, of the 79 names that collide, 64 collide with only one other name, and the remaining 15 collide with two names. For the Brown corpus word list the collision sets are bigger. Table 2 shows their sizes. Even for this word list, most collisions are with only one or two other words.

Implicit disambiguation appears to be feasible for selecting a word from a list. Collisions are not a major obstacle. Even for large vocabularies, collision rates are low, and collision set sizes are small.

Size	Count	Percent
1	17288	87.2
2	1690	8.5
3	531	2.7
4	200	1.0
5	70	.4
6	36	.2
7	14	.1
8	8	.0
Total	19837	100.1

Table 2: Collision set sizes for Brown Corpus. Size is the number of words in the collision, so size 1 is no collision. The eight-way collision is for the words acre, bard, bare, base, card, care, case.

Sometimes a name is not enough information

In many of the potential applications for telephone spelling, the items of interest are not the words per se, but rather the people (or other individuals) who have these names. The name is just a means of identifying the individual, and this can lead to a problem when more than person has the same name. This problem applies to any kind of spelling system, whether it uses the telephone or not. Even if there is no collision in selecting the name, that will not help select an individual if there is more than one person with the given name. It turns out that this non-uniqueness can be a much worse problem than name collision.

The preceeding section considered only the problem of selecting a name from a set of names. This section considers that each name stands for at least one person. Table 3 shows the name sharing in the two databases of names discussed above.

database	total people	distinct names	unique names	%	shared names	%	colliding names	%
applicants	9028	6138	4719	52	3586	40	723	8
students	25292	14246	10321	41	9704	38	5267	21

Table 3: Name sharing in two list of people's names. Note that roughly half the names in each list are unique, but a large minority are also shared (i.e. more than one person with that name, but no other name has the same 'telephone spelling').

In both databases there are more shared names than colliding names. The first database has 38 people surnamed "Smith"; the second has 290 entries for "Lee".

This problem has nothing to do with the means of entering the name. It would be just as bad if the last name were to be entered with a conventional keyboard. Other studies have found similar results. Lesk, with a database of 213,071 people, found that first and last name alone identified half the individuals [6]. Adding the name of the town raised this to 82%, and adding the street name brought it to 97%. In general, after obtaining the first five or so letters of the last name, it is much more useful to get some other kind of information, such as the first name, street, or town. Marics (personal communication) has similar results. Last name uniquely identified only 16% of the people in her database. Adding the first name raised this to 88%, and adding state brought it to 94%.

Entering English text may be possible

Implicit disambiguation using word lists appears feasible. Can one then consider free text entry as a problem of selecting a word from a very large word list – namely, a full dictionary? Note that there is no problem with synonyms, as there was with like-named people. It does not matter that there are two distinct words named "wind". But consider the number and size of the collisions for a full dictionary, as shown in table 4.

Size	Count	%
1	42813	95.0
2	1713	3.8
3	349	0.8
4	123	0.3
5	35	0.1
6	11	0.0
7	6	0.0
8	4	0.0
9	1	0.0

Table 4: Collision set sizes for Webster's Seventh Dictionary

More than three quarters of the collisions involve just two words, and the size falls off rapidly after that. From this we can conclude that it is feasible to enter words from the dictionary. Only rarely will there be a collision, and there will usually be only two alternatives. This suggests, for instance, the possibility of an online dictionary to provide definitions and pronunciations for words entered on the telephone keypad.

But English text is not composed of words selected at random, and the collision table is misleading, as it is not weighted for frequency of use. Many of the most common words have ambiguous spellings. In addition, there are many words missing from the wordlist. To determine whether entry of actual sentences is possible, one must examine actual texts.

I examined several collections of texts (the Wall Street Journal for 1988, and three Sherlock Holmes stories), counting each word as either "okay" (having a unique spelling), "ambiguous" (non-unique spelling) or "missing" (not in the Webster's Seventh dictionary). Table 5 shows the results:

To be sure, these are somewhat unusual texts — the Wall Street Journal has a fair

Source	Number of words	Unique words	%	Ambig words	%	Missing words	%
Wall Street Journal	20,691,239	8,049,923	38.9	8,633,941	41.7	4,007,375	19.4
A Study In Scarlet	44008	15132	34.4	23992	54.5	4884	11.1
"Baskervilles"	59699	21570	36.1	32854	55.0	5275	8.8
The Sign Of Four	43741	15572	35.6	24119	55.1	4050	9.3

Table 5: Distribution of words from text corpuses shows that about half have ambiguous telephone spellings, and a substantial number are altogether missing from the dictionary.

number of company names, acronyms, and abbreviations, and the Arthur Conan Doyle stories use an English different from modern usage in the USA, but they were the best I had, and I believe the following conclusions still apply. Clearly the number of missing words alone makes full text entry impractical. (It would be most interesting to determine how many of these missing words were simply derived forms — plurals, verb tenses, and such, and how many were proper nouns.) Even worse though is the number of ambiguous words. Each ambiguous word will require an action from the user. No matter how easy this action is, it will be unbearable to be required to make it on every other word, and if the vocabulary is expanded (to pick up missing words), this can only get worse. It is possible than additional knowledge (syntax) could be applied to provide further disambiguation at the word level, but that is a topic for further research.

Implementation

The Direction Assistance program [1] includes a module for telephone spelling. In that module, a word list is stored as a REDUCED LETTER TREE. Where an ordinary letter tree has 26 branches at each node, a reduced tree has one branch for each digit used in the reduced telephone alphabet. Given a digit sequence $[D_1, D_2, ...D_i]$, one begins at the root, taking the D_1'st branch, and so on for each digit. A path through the tree encodes a spelling in the telephone alphabet. At each node are stored a list of all words with the spelling. For certain vocabularies it may be economical to store words at the node corresponding to the shortest unique substring that spells the word, rather than spelling the word in full. Such a representation makes it easier to provide completion where the user enters only enough letters to uniquely identify the desired word.

Disambiguation by using statistics

An alternative knowledge source to a word list is information about the statistical combinations of letters in English. It ought to be possible to eliminate at least some strings simply because they can't form words, for example, "843" can't be "VGF" since those three letters never occur together in a word. There are many different kinds of knowledge about English that one might use, for instance one might notice that "VGF" has no vowel, so it can't be a word (although it might be an acronym). In the ideal case, by using knowledge of this sort one could pick out all and only the possible English words for input digit sequence. If this worked, this would allow input of full English text, without needing a dictionary.

Is it possible to use statistical knowledge about English for free text entry? I investigated the use of trigram letter probabilities as a source of knowledge for disambiguation. A trigram is a sequence of three letters $L_1L_2L_3$, and a trigram probability is just the probability that L_1L_2 will be followed by L_3. For any two initial letters, the sum of all trigram probabilities that begin with those two letters (27, one for each of the 26 letters, plus one for end of word) is 1, no matter how rare the letter pair is. A trigram frequency table records some of the redundancy in English — for example, having seen the letters "TH" you would expect to see a vowel or perhaps "R", but "X" would be unlikely in most styles of writing[1].

Constructing a trigram frequency table requires a large body of text. I used the Wall Street Journal text mentioned above. To apply the trigrams to phone spelling, I collapsed the table according to the placement of letters on the keypad. The collapsed table shows the probability for each pair of letters L_1L_2 and for each digit D, that the next letter will be L_3, given that L_3 is one of the three (or four) letters assigned to the D key. This then allows me to estimate the probability of a given letter string being the intended string for a supplied digit string. It is simply the product of the reduced trigram probabilities for each letter. It is then straightforward to write a program to enumerate all possible letter strings for a supplied digit string, compute the probability of each, and collect the most likely ones.

If this representation captures sufficient information about English, then one should be able to take a word, reduce it to a string of digits, and collect the most likely strings for that digit string. If the target word is often the first in the list, the scheme is likely to succeed. If the word appears low in the list, or worse, does not appear at all, the scheme will fail. I examined a word list of 473 words ($\leq$ ten letters long) chosen randomly from the Brown Corpus word list to see whether this would be so. I found that 100 of the words were rated either highest, or second highest. Unfortunately, many of the other words were lower in the ordering, so the mean position was 6.86, and 133 of the words were not found at all by the algorithm (that is, the estimated probability was below an ad-hoc cutoff value). I conclude that trigram frequencies are not sufficient information for disambiguation.

In retrospect, this should have been obvious, since Shannon's estimate of the information content of a single letter (in a trigram model) is 3.3 [10], and each digit contributes but 3.17 bits. It may be, though, that the trigram spelling algorithm could be improved by adding additional knowledge about English — such as the frequencies of word initial and word final di- or trigrams; or a filter to discard strings which can't be broken into syllables; or a larger, and more accurate, set of letter frequency data.

[1] One exception to this rule is the work of George Lucas.

Discussion

Implicit disambiguation is a useful technique for spelling words with the letters on the telephone keypad. To use it, the system designer must identify enough constraints on the possible inputs to reduce the information per key to below 3.2 bits. Whether this is possible depends on the application. If every, or almost every, string of letters is a possible input (as for example, in stock market abbreviations or computer passwords) then implicit disambiguation is impossible.

This paper has considered only the entry of letters. Some applications might require users to enter non-alphabetic data, e.g., numbers or punctuation. It is possible that some such applications could employ implicit disambiguation as well, for example, using syntactic constraints on the form of arithmetic expressions.

In many cases implicit disambiguation will be only partial, and there will be more than one possible word for a given input. When this happens the system will need to either employ explicit disambiguation on the word level, or look for some higher level knowledge source to provide additional implicit disambiguation. How often this will be required depends on the vocabulary, in particular on its size and the average word length.

Implicit disambiguation seems to be practical for selecting from sets of named entities (streets, words, people) when a word list can be provided. Designers of practical systems need to consider two additional issues, namely completion (allowing the user to specify only part of the name) and spelling correction. Implicit disambiguation does not appear feasible for entering unrestricted English text. Free text often uses words not found in dictionaries, and many common words are ambiguous. It is possible that further research could identify sufficient additional knowledge about English to make this possible. It is also possible that some applications may use sufficiently constrained subsets of English that free text entry would be possible. This remains a subject for further research.

Acknowledgments

I thank Chris Schmandt and Mike Hawley for providing databases and helpful comments. Comments from Barry Arons, Debby Hindus, and Sanjay Manandhar are also appreciated. Of course the author takes all responsibility for any folly which remains. This work was supported by a grant from the Nippon Electric Company Home Electronics Division.

References

[1] James R. Davis and Thomas F. Trobaugh *Direction Assistance* MIT Media Laboratory Speech Group Technical Memo 1 December, 1987

[2] M. Detweiler and R. Schumacher, Jr. and N. Gattuso Jr. Alphabetic input on a Telephone Keypad *Proceedings of the Human Factors Society– 34th annual meeting*, 1990

[3] L. Fast and R. Ballantine Dialing a name: Alphabetic Entry Through a Telephone Keypad *SIGCHI Bulletin* 20 (2) : 34. 1988

[4] John D. Gould and Stephen J. Boies Speech filing — An office system for principals *IBM Systems Journal* Vol 23 No 1 1984 65–81

[5] John D. Gould et al The 1984 Olympic Message System: A Test of Behavioral Principles of System Design *Communications of the ACM* Vol 30 Number 9 September 1987 758–769

[6] Mike Lesk Name Overlaps in Touch-Tone Coding in a Large Directory. Bell Laboratories unpublished memo 1979

[7] M. Marics How Do You Enter "D'Anzi-Quist" Using a Telephone Keypad? *Proceedings of the Human Factors Society– 34th annual meeting* 1990

[8] Samuel P. Morgan Minicomputers in Bell Laboratories Research *Bell Laboratories Record* Vol 51 July Aug 1973 pp 194–201

[9] David L. Reich An Automated Telephone Book Using Synthetic Speech. Research Report RC 9958 #44137 April 25 1983 IBM Thomas J Watson Research Center, Yorktown Heights, NY 10598

[10] C. E. Shannon Prediction and Entropy of Printed English *Bell System Technical Journal* 30 50–64 1951 Reprinted in *Key Papers in the Development of Information Theory*, edited by D. Slepian, IEEE Press 1974

[11] John C. Thomas and John C. Carroll Human factors in communications *IBM Systems Journal* Vol 20 No 2 1981 237–263

[12] Ian H. Witten *Principles of Computer Speech* Academic Press, 1982

Jim Davis received his PhD from the Media Laboratory in September 1989. His research interests include computer generation of natural language, voice interaction with computers, and developing computers as intelligent partners for musical performance.