

T12: AN ADVANCED TEXT INPUT SYSTEM WITH PHONETIC SUPPORT FOR MOBILE DEVICES

Naushad UzZaman
BRAC University
Bangladesh
naushad@bracuniversity.net

Mumit Khan
BRAC University
Bangladesh
mumit@bracuniversity.net

Abstract

The popular T9 text input system for mobile devices uses a predictive dictionary-based disambiguation scheme, enabling a user to type in commonly-used words with low overhead. We present a new text input system called T12, which in addition to providing T9's capabilities, also allows a user to cycle through the possible choices based on phonetic similarity, and to elaborate commonly used abbreviations, acronyms and other short forms. This ability to cycle through the possible choices acts as a spelling checker, which provides suggestions from the dictionary with similar pronunciation as the input word.

Key Words: T9, T12, Text Input, wireless communication service, SMS, email, phonetic similarity, spelling checker.

etc. When using an email application, the user may desire that these be written using the short form, but the input scheme translate these to the elaborated form for the recipient. With T12, the user is able to write the short form, and then use the *Next* key to cycle through the possible choices and pick the elaborated form if available in the dictionary. When using another application, the requirement may be quite the opposite however. In the case of SMS (Short Message Service), the requirement is that the messages be as short as possible, due to the limit of 128 or 160 characters per message (3). The T12 solution then is to convert the words to similar sounding words shorter forms or to acronyms by cycling through the choices using the *Next* key, reducing the length of the messages. For example, the words *phone* and *formula* converted to *fone*, *Amula* by cycle through. T12 also allows the user to check the spelling using the phonetic similarity measure.

1. INTRODUCTION

The tremendous increase in mobile usage in recent years has created a large demand for efficient text input systems for key-starved mobile devices. The text input schemes are needed not just for SMS (Short Message Service), but also email, Internet access, contacts, calendar, notes, task list and many more applications. One popular scheme for text input is T9 (1), which allows the user to use the 9 keys on the mobile keypad to enter text, and then to cycle through the possible choices using another key, called the *Next* key. While T9 allows disambiguation based on dictionary and frequency list, it does not allow the user to elaborate commonly used acronyms, abbreviations and short forms that mobile users often use, MacKenzie et al (2). It also does not allow a mechanism for prediction based on phonetic similarity, which is quite useful in languages where the misspelling are often phonetic in nature due to the language-specific orthographic rules, and also when using phonetically similar short forms. T12, our proposed text input scheme, extends T9 with these functionalities.

The commonly used phonetically similar short forms range from using *2morrow* instead of *tomorrow*, *4* instead of *for*, *ppl* instead of *people*, *r* for *are*, etc. The acronyms and abbreviations are also quite common, such as *BTW* instead of *by the way*, *IMO* for *in my opinion*,

2. TEXT INPUT ON MOBILE DEVICES

Unlike QWERTY keyboards, mobile device keyboards has 12 keys most of the cases. So such keyboards have to use one key to represent multiple inputs. On these keyboards, characters are typically grouped into sets and the sets are bound to a particular key. Most used layout is, ABC bound to key 2, DEF to key 3, GHI to key 4, JKL to 5, MNO to 6, PQRS to 7, TUV to 8 and WXYZ to 9. There are currently two ambiguous input methods used on mobile devices: single-tap and multi-tap (4). In the multi-tap mode the input character cycles with every press of its related key. Using the above layout, the letter H can be entered by pressing key 4 twice. When the user presses a different key, or waits for a timeout delay, the previous character is fixed in the input. In the single-tap method each key-press can represent any of its associated characters. Sequences of key-presses can then represent any word that can be constructed from the characters in the order they were entered. If more than one word is associated with a key-sequence (e.g. 7468 produces "pint" and "riot" amongst others) the user can cycle through the available set using another key on the keyboard. In terms of word-entering efficiency, single-tap is a considerable improvement over multi-tap, although this efficiency is dependent on: (i) whether the word wanted by the user is in the single-tap database, and (ii) the order in which the words tied to a key-

sequence is presented to the user. The first of these issues can be tackled through a careful choice of the corpus used to build the single-tap database. If we assume that the database contains most of the words required by the user, then the issue of word selection order provides the biggest source of potential inefficiency for the user, Hawes and Keller (5).

3. WHY IS IT CALLED T12?

Single Tap entry described above is T9, which means Text in 9 Keys. We propose T12, which means Text using 12 Keys. Purposes of key 2 to 9 are described above from Hawes and Keller (5). Among the other keys, 1 is bound to punctuations and symbols (. , - ? ! ‘ @ etc); key 0 is bound to space; key * is bound as the *Next* key to cycle through the possible choices in the T9 text input scheme, producing words with the same key sequence; and key # bound to cycle through for T12 method, producing words with the same pronunciation, acronyms, and elaborated forms. Figure 1 shows the key layout for our proposed T12 text input scheme.

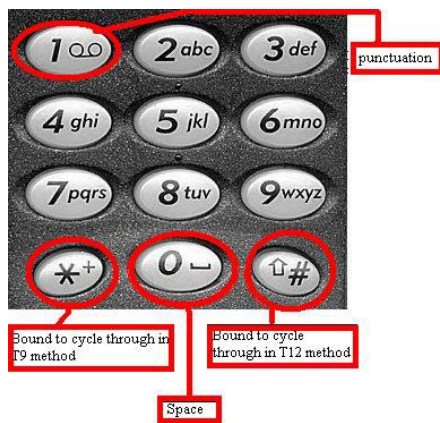


Figure 1: Image of a mobile keypad with the proposed functionality

4. CHALLENGES FOR T12

As described in the previous sections, T12 allows the user to use either the short form that can be elaborated, or the long form that can be shortened. There are quite a few challenges in implementing this capability, and we describe some of these below.

1. Given the non-phonetic nature of English, and its complex orthographic rules, the misspellings are often phonetically similar to the intended word. In the Internet culture, especially when using instant messaging and chat applications, users often intentionally use the misspelled version as well. For example, *fone* for *phone*, *grate* for *great*, *kat* for *cat*, etc.

2. Users will often omit the vowels when spelling certain commonly used words. For example *r* for *are*, *ppl* for *people*, etc.
3. There is also a tendency of using digits to replace similar sounding consonants or sequence of letters. For example, *4* for *for*, *2morrow* for *tomorrow*, etc.
4. There is an ever-increasing number of “Internet acronyms” which are in common use. For example *AKA* for *also known as*, *BTW* for *by the way*.
5. Sometimes abbreviations and acronyms include punctuations and sometimes not. For example *I.B.M.* instead of *IBM*, *USA* instead of *U.S.A.*
6. The use of apostrophe in informal language for contraction (6). For example, *haven't* for *have not*, *won't* for *will not*, etc.

5. THE T12 TECHNIQUE

The T12 text input scheme depends on a certain amount of “pre-knowledge” in the form of an enhanced dictionary, word list with frequency, and phonetic encoding information, as well as the actual algorithm described below.

5.1 Pre-knowledge

The T12 technology uses a dictionary and a word frequency list to create a final word list with the help of a phonetic encoding. This final word list includes the entries needed for translating the commonly used acronyms based on phonetic similarity. The T12 algorithm, described in Sec. 5.2, then uses the final word list.

5.1.1 T9 dictionary

T12 uses the same basic dictionary as the T9 input scheme found in (1), extended with the short forms commonly used in various Internet applications such as chat and email. Table 1 shows some of the commonly used ones that will be used to create the final word list.

Table 1: Sample of T9 dictionary

Shorthand	Phrase
2Day	Today
ASAP	As soon as possible
@	At

5.1.2 Word list with frequency

A word list with frequency information, typically collected from corpus analysis, is used to sort the suggestions when cycling through the choices using the *Next* key. For our implementation, we have used the one found in (7). Table 2 shows a sample word list with frequency information.

Table 2: Sample of word list with frequency

Word	Frequency
Today	263
Years	902
Orange	14

5.1.3 Phonetic Encoding

Phonetic Encoding encodes words based on their pronunciation. Among the many phonetic encodings in use for English, we have found Metaphone encoding, Philips (8), to be the most appropriate for T12. Metaphone partitions the English alphabet to 16 consonant sounds:

B X S K J T F H L M N P R Ø W Y

The Ø code represents the 'th' sound.

Metaphone uses the following transformation rules in its encoding:

Doubled letters except "c" -> drop 2nd letter.

Vowels are only kept when they are the first letter.

- B -> B unless at the end of a word after "m" as in "dumb"
- C -> X (sh) if -cia- or -ch-
S if -ci-, -ce- or -cy-
K otherwise, including -sch-
- D -> J if in -dge-, -dgy- or -dgi-
T otherwise
- F -> F
- G -> silent if in -gh- and not at end or before a vowel
in -gn- or -gned- (also see dge etc. above)
J if before i or e or y if not double gg
K otherwise
- H -> silent if after vowel and no vowel follows
H otherwise
- J -> J
- K -> silent if after "c"
K otherwise
- L -> L
- M -> M
- N -> N
- P -> F if before "h"
P otherwise
- Q -> K
- R -> R
- S -> X (sh) if before "h" or in -sio- or -sia-
S otherwise
- T -> X (sh) if -tia- or -tio-
Ø (th) if before "h"
silent if in -tch-
T otherwise
- V -> F
- W -> silent if not followed by a vowel
W if followed by a vowel
- X -> KS
- Y -> silent if not followed by a vowel

Z -> Y if followed by a vowel
S

Initial Letter Exceptions

- Initial kn-, gn- pn- ac- or wr- -> drop first letter
- Initial x- -> change to "s"
- Initial wh- -> change to "w"

5.1.4 Normalization

The input words are normalized to convert these to a base form. For example, 4 is converted to *four*, *asap* to *as soon as possible*, *I.B.M* to *IBM*, *Burger* to *burger* (conversion to lower case), etc.

Normalization Algorithm

if in the shorthand of T9 dictionary (Table 1)

then convert it to phrase

//asap to as soon as possible

if there are any digit

then convert it to digits

//4mula to formula

if any punctuation

then exclude the punctuations

//I.B.M to IBM

convert all to lower case

//Burger to burger

We generate a table with words, normalize the list, and create the corresponding metaphone encoding. Table 3 shows the result of this step.

Table 3: Sample of word, normalized form of that word and its Metaphone encoding

Word	Normalized word	Metaphone encoding
What	what	WT
Kat	kat	KT
4mula	fourmula	FRML
asap	as soon as possible	SSNSPSBL

5.1.5 Approximate string matching algorithms

One the word list is normalized and the corresponding Metaphone encodings are computed, we use two different approximate string-matching algorithms to create the list of suggestions that the user can cycle through. The two algorithms are ED (Edit Distance) (9) and LCS (Longest Common Subsequence) (10).

act	0.33	0.33	0.33	0.66	0.66	0.66	0.495	59
acute	0.2	0.2	0.2	0.66	0.66	0.66	0.43	23
cat	0.66	0.66	0.66	0.66	0.66	0.66	0.66	39
caught	0.33	0.33	0.33	0.66	0.66	0.66	0.495	86
coat	0.5	0.5	0.5	0.66	0.66	0.66	0.58	34
code	0	0	0	0	0	0	0	52
cut	0.33	0.33	0.33	0.33	0.33	0.33	0.33	29
equity	0.166	0.166	0.166	0.33	0.33	0.33	0.248	20
gate	0.5	0.5	0.5	0.66	0.66	0.66	0.58	35
god	0	0	0	0	0	0	0	36
good	0	0	0	0	0	0	0	25
got	0.33	0.33	0.33	0.33	0.33	0.33	0.33	932
guide	0	0	0	0	0	0	0	12
kid	0.33	0.33	0.33	0.33	0.33	0.33	0.33	16
quid	0	0	0	0	0	0	0	13
quiet	0.2	0.2	0.2	0.33	0.33	0.33	0.265	62
quite	0.2	0.2	0.2	0.33	0.33	0.33	0.265	412
quote	0.2	0.2	0.2	0.33	0.33	0.33	0.265	10

7. We assume a threshold of 0.65, so accepted words are ones with scores > 0.65 are:

- 1) kit
- 2) cat

7. Ranked according to the average of EDscr max and LCSscr maximum values:

- 1) kit (0.66)
- 2) cat (0.66)

8. Both have the same value, so ranked according to frequency:

- 1) cat (39)
- 2) kit (9)

So, with the *Next* key #, when we cycle through the choices, we will get our first suggestion as *cat*, and then *kit* for the input word *kat*.

The key sequence for the word *kat* is 528. The T9 cycle-through with this key sequence produces the list *lat*, *lau*, *lav*, *jav*, *kat*, *kau*, and *jau*. The T12 cycle-through, given the above threshold, produces the list *cat*, and *kit*.

Example 2:

Suppose the input word is *gr8*.

1. word = gr8

2. normalized word = great (since it is found in T9 dictionary, hence in the normalization process, it will be converted to corresponding phrase)

3. Metaphone encoding = KRT

4. words with the encoding KRT:

- a. gr&d
- b. gr8
- c. great
- d. agreed
- e. card
- f. cared
- g. carried
- h. court
- i. create
- j. cried
- k. crowd
- l. crude
- m. grade
- n. great
- o. grid
- p. guard

5. Described in Table 6.

6. Described in Table 6.

Table 6: Example of ED and LCS for input *gr8*

word	ED with word gr8	ED with normalized word great	Max ED	LCS with word gr8	LCS with normalized word great	Max LCS	Average	Freq
gr&d	0.5	0.4	0.5	0.66	0.5	0.66	0.58	9
gr8	1	0.4	1	1	0.66	1	1	9
great	0.4	1	1	0.66	1	1	1	9

agreed	0.33	0.5	0.5	0.66	0.6	0.66	0.58	10
card	0.25	0	0.25	0.33	0.25	0.33	0.29	57
cared	0.2	0.2	0.2	0.33	0.4	0.4	0.3	13
carried	0.142	0.142	0.142	0.33	0.4	0.4	0.271	138
court	0.2	0.2	0.2	0.33	0.4	0.4	0.3	285
create	0.16	0.66	0.66	0.33	0.8	0.8	0.73	82
cried	0.2	0.2	0.2	0.33	0.4	0.4	0.3	30
crowd	0.2	0.2	0.2	0.33	0.2	0.33	0.265	43
crude	0.2	0.2	0.2	0.33	0.4	0.4	0.3	13
grade	0.4	0.4	0.4	0.66	0.6	0.66	0.53	19
grid	0.5	0.4	0.5	0.66	0.5	0.66	0.58	11
guard	0.4	0.2	0.4	0.66	0.4	0.66	0.53	25

7. We assume a threshold of 0.65, so the accepted words with scores > 0.65 are:

1. gr8
2. great
3. create

8. Ranked according to the average of EDscr and LCSscr maximum values:

1. gr8 (1.0)
2. great (1.0)
3. create (0.73)

9. Both have the same values, so ranked according to frequency:

1. gr8 (9)
2. great (9)
3. create (82)

So, with the *Next* key #, when we cycle through the choices, we will get the first suggestion as *gr8* or *great*, and then *create* for the input word *gr8*.

The key sequence for the word *gr8* is 478. The T9 cycle-through with this key sequence produces the list *gru* and *irv*. The T12 cycle-through, given the above threshold, produces the list *great* and *create*.

6. DISCUSSION

Our proposed T12 input scheme extends T9's capabilities using phonetic support that allows features such as checking spelling and elaborating short-forms using phonetic similarity measures, while maintaining complete backward compatibility with T9. For example, input words such as *kat*, *fone*, and *ppl* produce the expected *cat*, *phone*, and *people* respectively using the T12 scheme.

T12 also compresses messages, an important feature when considering applications with limited message size such as SMS, by allowing the user to cycle through the choices and picking the short forms. For example, the user is able to choose *2night* for *tonight*, *4* for *for*, etc, shrinking the message size. In some cases, it may be

necessary to add the short form to the dictionary first before the user is given the choice. For example, to be able to use the short form *msg* for *message*, it is necessary to add *msg* to the dictionary by using the standard MultiTap methods.

T12 performs at least as well, and better than in some cases, than T9, using the keystroke per word metric. When considering short form elaboration or contraction, it is likely to perform better than T9. Table 7 shows the relative performance in terms of keystrokes needed for a set of commonly used words, showing the performance improvement in some of the cases.

Table 7: Examples of Key press needed in T9 and T12

word	T9	T12
be right back	13 key press	brb - #: 4 key press
as soon as possible	19 key press	asap - #: 5 key press
before	6 key press	be4 - #: 4 key press
message	7 key press	msg - #: 4 key press
today	5 key press	2day - #: 5 key press
great	5 key press	gr8 - #: 4 key press
read	4 key press	read: 4 key press
not	3 key press	not: 3 key press

7. CONCLUSION

We propose a new text input system the adds to the capabilities of the popular T9 scheme by using phonetic similarity to detect spelling errors as well as to elaborate acronyms and short forms commonly used in Internet

applications. The examples shown above hopefully illustrate the usefulness the T12 text input system for key-starved mobile devices.

ACKNOWLEDGEMENT

This work has been supported by BRAC University.

REFERENCE

1. <http://www.t9.com>
2. I. S. MacKenzie, H. Kober, D. Smith, T. Jones, and E. Skepner, *LetterWise: Prefix-based disambiguation for mobile text input*, ACM Symposium on User Interface Software and Technology, pp. 111-120, (2001).
3. Long SMS definition, available online at <http://www.phonescoop.com/glossary/term.php?gid=132>
4. W. Soukoreff and I. S. MacKenzie, "Text entry for mobile computing: Models and methods, theory and practice", *Human-Computer Interaction*, 17. p. 147-198, (2002).
5. Nick Hawes and John Kelleher, "Context-Sensitive Word Selection for Single-Tap Text Entry", 16th European Conference on Artificial Intelligence (ECAI'04). Valencia, Spain, IOS Press, (2004).
6. Detail of Apostrophe, available online at <http://www.uottawa.ca/academic/arts/writcent/hypergrammar/apostrph.html>
7. Word list with frequency, available online at http://www.comp.lancs.ac.uk/ucrel/bncfreq/lists/1_2_all_freq.txt
8. Lawrence Philip's Metaphone Algorithm, available online at <http://aspell.sourceforge.net/metaphone/index.html>
9. Levenshtein edit distance algorithm, available online at <http://www.nist.gov/dads/HTML/Levenshtein.htm>
10. T. T. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*, MIT Press, Cambridge, MA, 1990.