Optimizing Small Keyboard Layout to Minimize "T9onyms"

P. Boothe

January 2, 2010

Abstract

Sending text messages on cell phones which only contain the keys 0-9 and # and * can be a painful experience. In order to make it easier, phone manufactureres have designed predictive typing schemes. In this paper we consider multiple algorithms to design an optimal map of cell phone keys to letters so that, for a given dictionary of words with associated frequencies, the expected number of characters typed is minimized. We end by describing the best mapping we found, in the (vain) hope that it can become a new alternative layout for cell phone keyboards. Also, it is quite fun to think about how one might optimize this system.

Typing a message using a keyboard which has fewer keys than the alphabet has letters can be a painful task. In an effort to minimize this pain, cellular telephone manufacturers have created the "T9" input method, which attempts to guess which letter (of the 3 or 4 possible) is intended when a user hits a single key. Enhancements of this input method also provide "speculative lookahead" to report, at any given time, what word it is that the user is most likely trying to enter and provide a completion. Unfortunately, because there are fewer keys than there are letters in the English alphabet, there are words which have different spellings but the same input sequence. As an example, in the traditional mapping of keys to characters ($abc \rightarrow 2$, $def \rightarrow 3$, etc), both "me" and "of" have the input sequence "63". Two words with the same input seuquence forces the user to press a third button to cycle through the possibilities in order from most likely ("of") to least likely ("me"). Even worse: "home", "good", "gone", "hood", "hoof", "hone", and "goof" all have the input sequence "4663". If the * key is used as the "next match" button, then the user will actually have to type in "4663******" to type in the word "goof".

When two words have the same input sequence (neglecting the *'s at the end) then these words are **t9onyms**. When two or more words are t9onyms, then the less-popular words require extra keypresses, raising the expected number of keypresses to type in a random selection of words from the english dictionary. In order to type a given word, one must press one button for every character in the word, followed by pressing the * key as many times as there are t9onyms which are more likely than the desired wored. Because typing on a cellphone keyboard is already an unpleasant experience, we would like to minimize the expected number of keystrokes. The number of keystrokes a word requires is equal to the number of letters in the word, plus the number of t9onyms which are more frequent than the word. Formally, we define the MinimalKeystrokes problem as:

Theorem 1 (MinimumKeystrokes)
Instance *An alphabet A, a set of words and their associated probabilities W, and a set of keys K ($|A| > |K|$).*

Question *What is the mapping of $A \to K$ which minimizes the expected number of characters to type a word from W?*

and the corresponding decision problem is

Theorem 2
Instance *An alphabet A, a set of words and their associated probabilities W, a set of keys K ($|A| > |K|$), and a number f.*

Question *Is there a mapping of $A \to K$ in which the expected number of characters to type a word from W is less than f?*

Before proceeding, we note that this we note that this problem is in $\mathcal{NP}$, as any assignment may be verified to have and expected number of characters of less than f in time proportional to the total length of all the words in W.

It is highly likely this problem is $\mathcal{NP}$-complete, due to a reduction from one of the optimal trie problems. But I'm not there yet.

However, our arbitrary remappings of the keyboard may be quickly deemed unusable. Therefore, we define a refinement of the problem, in which the characters are required to be kept in alphabetical order as a sequence, and our only choices are about how to divide the subsequence into key assignments. The default layout can be described as the partition $abc|def|ghi|jkl|mno|pqrs|tuv|wxyz$. Is this partition optimal? In order to decide this, we define the problem MinimumKeystrokePartition as:

Theorem 3 (MinimumKeystrokePartition)
Instance *A sequence of letters $A = \{a_1, a_2, \ldots\}$, a set of words and their associated probabilities W, and a set of keys $K = \{2, 3, 4, \ldots\}$ ($|A| > |K|$).*

Question *What is the mapping f of $A \to K$ which minimizes the expected number of characters to type a word from W, and where $f(a_1) = 2$ and if $f(a_i) = k$, then either $f(a_{i+1}) = k$ or $f(a_{i+1}) = k + 1$?*

As before, the corresponding decision problem is

Theorem 4
Instance *A sequence of letters $A = \{a_1, a_2, \ldots\}$, a set of words and their associated probabilities W, a set of keys $K = \{2, 3, 4, \ldots\}$ ($|A| > |K|$), and a*

parameter e.

QUESTION *Is there a mapping f of $A \to K$ in which the expected number of characters to type a word from W is less than e, and where $f(a_1) = 2$ and if $f(a_i) = k$, then $f(a_{i+1}) \in \{k, k+1\}$?*

Again, we note that this decision problem is in $\mathcal{NP}$, as any proposed partition may be verified, in polynomial time, to have an expected value less than e.

1 Exhaustive Search

Because our problems are $\mathcal{NP}$-hard, we move on to combinatorial enumeration and experimental results. The number of partitions of a set of size n, which we denote $p(n)$ is actually relatively small. In particular, $p(26) = 2,436$[1], which is easily enumerated and tested.

[1] `http://www.wolframalpha.com/input/?i=PartitionsP[26]`