Urban Maths: Mind Your S's and D's!

A. Townie

1 PEBCAK

I still spend some time writing computer code, but not half as much as I used to do when I first began my career. Back in those days, which are longer ago than I would care to mention, I typed many more lines of code than I did lines of text. And the nature of the code I was writing, and the tools that I was using, meant that I typed *string* quite a lot. Now, that wouldn't necessarily be a problem, but that sequence of characters has become permanently lodged in the muscle memory of my fingers, so much so that when I try to write *strong* I often end up with *string* instead.

Many people would, perhaps quite rightly, triage this as a PEBCAK bug, that is one where the Problem Exists Between Chair And Keyboard. However, I think that's somewhat unfair. I don't mind mistyping, but what I do mind is the fact that this particular mistype is not highlighted with a red underline. In particular, I mind that the creator of the nearly ubiquitous qwerty keyboard chose to put *i* and *o* right next to each other when, in my problem case, replacing the latter with the former still produces a valid word!

2 Decisions and simplifications

Now, I don't think it's fair to criticise without at least trying to make things better. So, I embarked on a quest to see if I could create a 'better' keyboard. In particular, I'd like a keyboard where replacing one letter with a neighbouring one didn't create valid words, or at least didn't create too many valid ones.

There are other similar issues relating to typing accuracy that I intend to ignore within this article. Some examples include:

- Reversed letters. This means I'm not interested in the difference between *goal* and *gaol*, even though it can be quite important in the footballing fraternity.
- Misplaced spaces. This means I'm not concerned with someone who *likes having jelly* and something that is *like shaving jelly*, even if that cuts down the number of parties I'm invited to.
- Additional (or missing) letters. This means that I'm not worried about an extra *t* creeping into phrases like *the busy ladies*, although I probably should be.

To further simplify my analysis, I'm only interested in mistypes that occur in the same horizontal row of the keyboard. I find the slight horizontal offsets that are used on most keyboards, which nudge the keys away from a strict grid formation, means that it's rare I unknowingly get a key from the wrong row.

And, if that wasn't enough, I'm also going to ignore the fact that some words are used much more often than others, which should mean that some mistypes are much more likely to happen than others.

3 Letter replacement

Having made those (admittedly rather significant) simplifications the analysis is quite straightforward. The first stage involves creating a dataset that describes the number of valid words that can be created by replacing one letter for another; note that, in this stage we consider every possible pair of letters, so that we then have data that supports the analysis of alternative keyboard layouts. Also note that the data used in this article is based on an analysis of just under 63,000 words, which were taken from a list that is almost always included in distributions of the Linux Operating System.

Using this list of words, we look through all 650 (26×25) combinations of pairs of dissimilar letters, counting the number of times that a letter replacement in a valid word results in another valid word. The frequency distribution of these counts (as determined by Gnuplot's smooth function) is shown in the top-half of Figure 1. It is apparent that this is a long-tailed distribution, with over half its elements falling in the first bin of the histogram.

The bottom-half of Figure 1 shows, on the horizontal scale, the actual counts themselves; the vertical scale is random jitter, so that the density of dots provides a visual representation of the frequency distribution. Again, this shows the long tail of the distribution. It also indicates that the 650 values are actually 325 repeated values; this follows from the observation that letter replacements are reversible, for example:

- Replacing the *e* in *bone* with *d* to get *bond* is matched by;
- Replacing the *d* in *bond* with *e* to get *bone*.

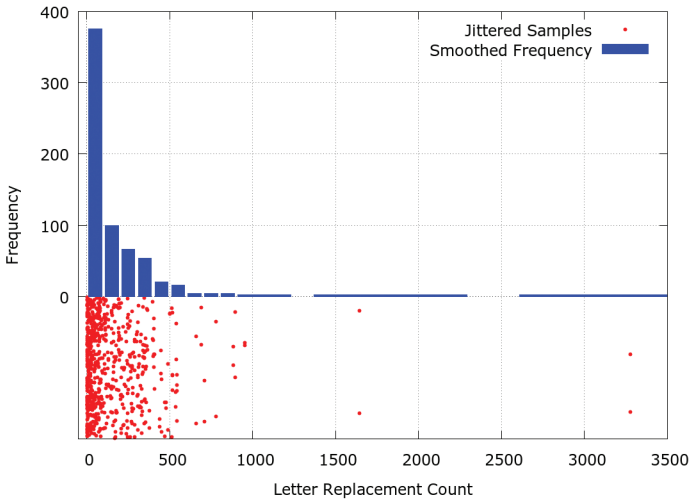

Figure 1: Distribution of new valid word counts.

Table 1 shows the five worst pairs of letters, as determined by the above analysis. Note that *d* and *s* are by far the worst pair, accounting for almost double the number of errors of the next worst pair – and these letters are placed next to each other on the

standard qwerty keyboard. (My mistyped pair of *i* and *o* isn't terribly bad, being the 12th worst pairing, with a replacement count of 538.)

Letter one	Letter two	Replacement count
d	s	3,276
d	r	1,642
r	s	951
a	o	895
a	i	883

Table 1: Worst five letter replacement pairs

4 Qwerty and Dvorak

The second, and final, stage of the analysis involves walking through each keyboard row, summing up the number of valid words that could be created by mistyping neighbouring letters. So, for the first row of the standard qwerty keyboard, we'd sum the number of valid words created when substituting:

- *q* with *w*
- *w* with *e*
- *e* with *r*
- ...
- *o* with *p*

We'd then repeat the same process for the second and third rows to give us an overall total. In the case of qwerty, the total is 6,624, and the worst case pairing is, as noted earlier, *s* and *d*, which contribute 3,276 of the total.

Given those observations, it's not that difficult to create a keyboard that has a lower, and hence better, total letter replacement count. In fact, any arrangement that separated the *s* and *d* would be better. But, before we think about trying to come up with an optimal keyboard (based on our chosen measure) we ought to consider the most significant challenger to qwerty, namely the Dvorak keyboard (Figure 2).

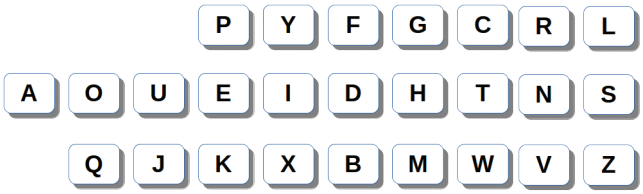

Figure 2: Dvorak keyboard (letters only).

The Dvorak keyboard was announced in the 1930s and was based on a detailed analysis of the English language (much more detailed than the simple analysis described above) along with a study of the physiology of the human hand. To say that this new layout caused some controversy would be rather like saying there has been mild disagreement between Frequentists and Bayesians on the fundamental underpinnings of probability theory. The keyboard controversy has even extended to economics since, if Dvorak is clearly better than qwerty then the efficient market hypothesis would suggest we should all be using keyboards with the vowels on the left of the middle row.

Given this tumultuous background our very simple analysis has absolutely no hope of settling the qwerty versus Dvorak debate. Despite that, it's interesting to see how Dvorak fares. It has a total score of 5,788 (as opposed to 6,624 for qwerty). So, based on our approach, Dvorak is better than qwerty, but not that much better. The worst pair of letters on the Dvorak keyboard are *a* and *o*, which contribute 895 of the total; as shown in Table 1 this is the fourth worst pairing.

5 Optimised alternatives

A number of other alternative keyboard layouts have been suggested. I certainly remember some devices that had the letters in alphabetical order, a layout that scores 2,937, comfortably beating both qwerty and Dvorak. There are also some layouts designed specifically for two hand typing; in effect these have a total of six rows, three for each hand.

To get a quick idea of what a good – I hesitate to use the word optimal – keyboard would look like, a very simple hill descent optimiser was implemented. This starts with a random keyboard (with three rows, containing 10, 9 and 7 letters, respectively) and makes the best improvement it can by swapping any two letters. This letter-swapping process is then repeated until no more improvements can be made. The statistics for the keyboard scores from 1,000 replications are illustrated in Figure 3 (which is of the same form as Figure 1).

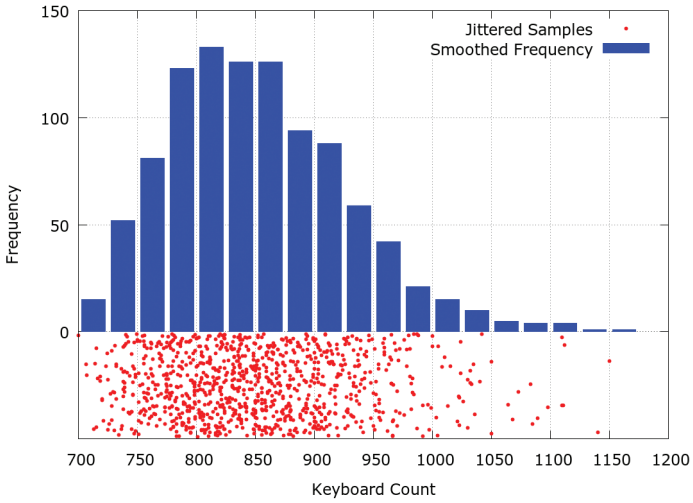

Figure 3: Distribution of hill descent optimised keyboards.

Two things are apparent from this figure. Firstly, there is a reasonably wide distribution of local minima, as found by the hill descent algorithm. Indeed, across the 1,000 replications, there were no duplications amongst the unimprovable keyboard layouts. This suggests the space we are trying to optimise over contains lots of local minima and, if we were serious about trying to find an optimal keyboard we should try a different optimising routine, perhaps simulated annealing. But, if we were really serious about finding an optimal layout, we wouldn't have introduced all the simplifications we did early on in this article!

The second thing that is apparent is that even the worst of the keyboards found by the hill descent algorithm is much better than an alphabetical layout (approximately 1,150 compared to 2,937), which in turn was much better than Dvorak and qwerty.

6 Conclusions

Despite all the simplifications and limitations inherent in this analysis it really does look as though a better keyboard layout could be created. On the one hand, the continued ubiquity of

qwerty would seem to argue that any new layout is unlikely to catch on. However, the introduction of new devices, especially those with touch screen interfaces, may provide an opportunity for new layouts to gain ground.

And, there may even be an opportunity for two different layouts to co-exist side-by-side. This is already the case for the numeric keypads on computer keyboards (which have 9 at the top-right) and the numeric keypads on telephones (which have 9 on the bottom-right). This difference has caused me to mis-dial quite a lot of phone numbers, and in that case there's no such thing as an invalid number and no red underline to help me out. So, on reflection, maybe sticking to a single layout is best after all!

7 Notes

Gnuplot is a 'portable command-line driven graphing utility'; it is available for many platforms, including Linux and Windows. More information may be found at `www.gnuplot.info/`.

The Wikipedia article (`http://en.wikipedia.org/wiki/Dvorak_Simplified_Keyboard`) is a good place to start when looking for further information about the Dvorak layout.

Acknowledgement

'Urban Maths' cartoonist: Adrian Metcalfe – `www.thisisfruittree.com`