4. Codings, Slips, Defences, and Repairs

Why we've included this: emphasis on transcription and on avoiding errors and sometimes repairing them

A coding is a transcription. Instead of a postal area spelled out in words, we can use a zip code or a postal code. Our interest is in transcription errors: what happens when somebody writes down the wrong code, for example? Discovering that there has been error is only half the story: what happens next? how easy is it to guess what was intended and fix it – the repair length? We shall look at three coding schemes which have different degrees of defence against slips.

Resistor coding

Our first example is a coding scheme that has almost no built-in defences against slips.

Electrical resistors are marked with 3, 4, or 5 coloured rings showing their resistance value and their manufacturing precision. As information structures go, such codings are very simple, intended to support only the activities of search (find a resistor of value X) and transcription (turn a number, like 68,000±5%, into a set of coding rings, or vice versa).

A resistor looks like this:

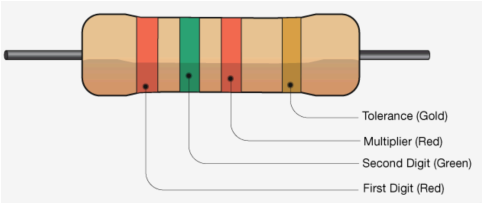

Figure 14. A typical resistor. This one is 2.5KΩ ± 5%.

The 3 coloured bands to the left indicate the resistance in ohms, and the separated band on the right indicates the tolerance or manufacturing precision. The band colours indicate digits: black = 0, brown = 1, red = 2, etc. Reading left to right, the first two bands show 2 digits and the third band is a multiplier showing the number of zeros to add. In Figure 14 the first two bands are red (2) and green (5). If the third band were black, that would be a 25 ohm resistor. As it's red, we add two zeros, giving 2500 ohms, usually written 2.5KΩ (but see below). The tolerance band is gold, indicating 5% tolerance.

Defence

How to defend against slips? By international convention, resistors are not allowed to have any old arbitrary value, but instead each decade is divided into 12 steps, starting at 1 and increasing by 21% at each step, rounded off: 1– 1.2 – 1.5 – 1.8 – 2.2 – 2.7 – 3.3 – 3.9 – 4.7 – 5.6 – 6.8 – 8.2 – 10. So there are resistors of 33Ω, 330Ω, 3.3KΩ, etc, and of 47Ω, 470Ω, 4.7KΩ, etc, but there are no resistors of 40Ω. This restriction greatly reduces the number of possibilities to choose between and allows a reasonably discriminable set of colours to be used.

There is a second type of defence, not so universal. Printed circuit diagrams are often badly printed, or get photocopied numerous times, and decimal points may get lost or intrusive specks may be read as decimal points. To avoid the problem the numbers can be printed as 5K6 (for 5.6KΩ) or 3M3 (for 3.3MΩ). This is not universal practice but seems to us an excellent move.

Repairs

In the typical scenario of use, someone is building a circuit from a circuit diagram such as Figure 15, and needs to pick out the right resistors; so they need to work out the colour bands for 510Ω, 10KΩ, and 470Ω. Picking out the wrong colours would constitute a transcription error, and the effect would be that the circuit vibrated at the wrong speed or not at all (or possibly burnt out the transistors, though that's unlikely with a six volt power source). Errors in the first two bands might not be serious, but errors in the third band, the multiplier, would mean that a selected resistor would be at least ten times larger or smaller than intended, which is quite a big difference. 'Repair', in this scenario, means no more than observing malfunction and carefully checking all the components. A slight error may never be noticed.

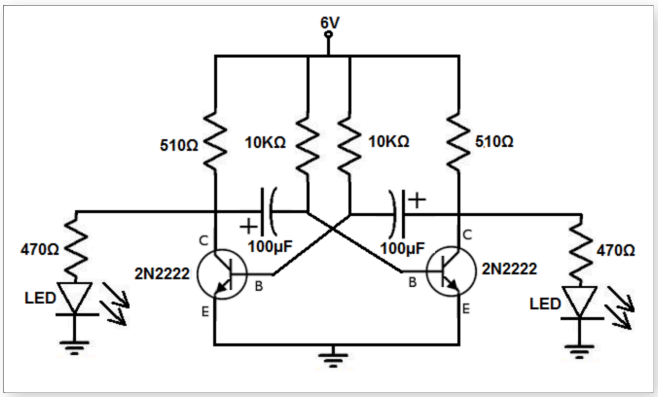

Figure 15. A typical small circuit diagram, showing, among other components, resistors, indicated by zig-zag lines. (This circuit is a 'multivibrator', from http://www.learningaboutelectronics.com/Articles/Astable-multivibrator-circuit-with-transistors.php)

We have absolutely no data about correcting mistakes made with resistors .... TBC lots of guess work here, maybe we can talk to someone??

Long numbers, specifically ISBN

Contemporary society uses long numbers in many ways: bank routing numbers, passport numbers, credit cards, airline tickets, product numbers, and even library membership numbers. All of these numbers are protected by a 'check digit', because when a person reads them to another person or types them on a keyboard, mistakes – transcription errors – are entirely possible. Typical transcription errors are:

- mistyping or misreading a digit – 1234 becomes 1284
- transposing neighbouring digits – 1234 becomes 1243
- doubling the wrong digit – 1223 becomes 1233.

The purpose of the check digit is to detect the commonest such mistakes.

Part 5

Defence against slips

Different defence schemes are used in different coding systems. We shall focus on ISBN numbers, 13-digit unique identifiers for books, because they are reasonably familiar and not so complex as some of the coding schemes. The current ISBN scheme uses the 13th digit as a check digit, calculated from the previous 12 digits in a manner reminiscent of the 'casting out nines' check of divisibility by 9: the first 12 digits are multiplied alternately by 1 and by 3 and the products are added. Take the last digit of the total (so if the total comes, to say, 123, the last digit is 3). If the last digit is 0, then the check digit is 0, and otherwise the check digit is (10 - that last digit) – so if the last digit is 3, then the check digit is 7.

If you wish to see this in operation, you can visit https://tgworkshop.me.uk/bookdemos/isbndemo/ where you can check the validity of 13-digits numbers. Try it with a valid ISBN and with slightly garbled versions.

The ISBN scheme detects the most frequent single errors: if a digit is mistyped, or if adjacent digits are transposed, the check digit will not correspond to what it should be. Some other errors are not detectable.

Repairs

If an ISBN number is invalid, how do we repair it? We have absolutely no data on this, but we imagine that the commonest scenario is simply someone trying to locate a book and mistyping the ISBN. Our experience is that if you type a 13-digit ISBN into a search engine, that's all it needs to locate the book and give you all the other information, but if you deliberately make a mistake, you get a message along the lines of "Your search - 9781781574040 - did not match any documents". So you don't get the wrong book, which is a good start, but you don't get any help in repairing the error. Still, in the scenario we gave, that's not much of a problem; if you type a number and the search says "sorry, no" you just have to retype the number. So the repair length is quite acceptable.

On the other hand, consider a slightly different scenario: you read a promising review and you note the ISBN number – and nothing else. If you wrote the number down wrong and you've no longer got access to the book or the review or wherever you found it, you're in trouble.

Somewhat to our surprise we could find no 'auto-suggest' facility on the web. Given that one of the most transcription errors for long numbers is transposing adjacent digits, it would be easy to make suggestions when an ISBN number fails to validate. In a 13 digit number where the 13th digit is assumed to be correct, there are only 11 possible transpositions. If any of those yields a number that validates against the check digit, it could be suggested to the user as a possible correction. (Going slightly farther, it would also be possible to check ISBN data sources to see whether the resulting 13-digit number was in fact an assigned ISBN, and if so, to present the associated metadata – title, author, etc.)

In the ISBN scenario, we hazard a guess that the error-detection scheme is adequate for detecting transcription errors, since the errors it detects are most frequent. Searching for error reports on the web, it seems that most problems are bureaucratic ones – wrong numbers assigned, numbers assigned twice, conflict between ISBN and other book-numbering schemes – so perhaps transcription errors are sufficiently defended against.

How true is that of other scenarios where long numbers have to be transcribed? What about transcribing a financial number wrong and having someone lose a lot of money, for example? In that scenario, and others where the costs of errors are high, there are schemes for error-*correcting* codes; if a single digit is mistyped, or if to adjacent digits are transposed, there is enough redundancy in the coding scheme to let the mistake be repaired automatically.

What3Words

Our last example is a new, and very original coding scheme: every 3 metre square of the world has been given a unique combination of three words. This is very much more precise than a post code or a zip code, more precise even than a street address, much easier to use under adverse conditions than a map grid reference. The scheme is called What3Words and on their website, https://what3words.com/, they claim what uses?? We are impressed by this idea, and it is becoming more widely used.

Defences

Words are easier to get right than strings of numbers, but even so, clearly people can get them wrong. The developers write:

> People make errors – lots of errors – and what3words is optimised to recognise and correct mistakes by both the sender and receiver of a 3 word address. The what3words AutoSuggest system picks up errors in spelling, typing, speaking, mishearing and misremembering 3 word addresses.
>
> We have shuffled all of our similar sounding 3 word locations as far away from each other as possible, so we can use your location to intelligently guess where you meant. [Whereas] with street addresses, similar sounding addresses are near enough to each other that it can be very confusing, and often leads to huge delays whilst people work out what has gone wrong and what the right address actually is.
>
> source
>
> https://intercom.help/what3words/en/articles/2212828-what-if-i-spell-or-say-a-word-incorrectly-or-get-the-words-in-the-wrong-order

This is the code of a place in the UK: `shoelaces.paintings.friday` . If you try slightly different words you get interesting results. `shoelaces.printings.friday` is in Mexico. `shoelaces.painting.friday` (singular painting) is somewhere in the States, while `shoelaces.panting.friday` is in Alaska again. This is exactly what the developers intended, because the most likely transcription errors will give results that are clearly inappropriate because they are so very far away.

The app comes with an auto-suggest feature that tries to help by offering three alternatives locations for whatever is typed or spoken. After speaking `shoelaces.paintings.friday` the auto-suggest offered `shoelaces.paints.friday` and `shoelace.paintings.friday`, neither of which is remotely near the intended target – so I can be sure that those are not really where I'm going.

All of the above examples use words that are known to the app. When an unknown word is tried, such as `fishcake`, the auto-suggest offers words that have some similarity – `pancake` and `fishlike`.

Repairs

If an emergency call to a help centre in the UK is given a W3W code that is misheard, can they they find the place intended? W3W tries to put likely confusions or mishearings into very different places – Alaska instead of UK – so the basic repair method is simply to look at the places offered and discard ones that are clearly not appropriate. We wonder whether that is adequate, and whether the auto-suggest should offer more than three alternatives; or possibly the user could indicate which continent the target is expected to be on, and the auto-suggest could then refine its suggestions.

Nevertheless, although the repair may not be as easy as the developers hoped (and once again, we have absolutely no data on this) what impresses is that the developers have very consciously taken into account error-proneness and repair. Their auto-suggest is a **design manoeuvre** that could well be adopted in other situations.

What we have learnt

Coding schemes highlight one particular type of error-proneness, the transcription error. We looked at examples with little defence against transcription errors, the resistor code; with error detection built into the code, but not much help in error repair, the ISBN coding; and with sophisticated design against errors, intended to not only to make errors glaringly obvious but also offering some help in repairing possible errors, theWhat3Words app. The lesson to be learnt is, obviously, that it makes sense to consider likely errors and to try to defend against them and to reduce the repair length.