21

Codes – Protecting Data Against Errors and Loss

Michael Mitzenmacher

Harvard University, Cambridge, USA

21.1 Introduction

Suppose that, after meeting a new friend at a party, you want to get his or her cell phone number – ten digits. (If you’re single, feel free to put this story in the context of meeting a special someone in a cafe or other locale.) You don’t have your phones with you, so you resort to having your friend write their telephone number on a nearby scrap of paper. Sadly, your new friend’s handwriting is rather messy, and the ink has a tendency to smudge, so you are worried you will not be able to read one (or more) of the digits later. Let us assume that if you cannot clearly read a number you will simply treat it as unknown, or erased. You might read

617-555-0?23,

where we use the question mark to mean you were not sure what that number was. You’d prefer to avoid calling multiple numbers hoping to find the right one, so you consider what you can do to cope with the problem in advance.

Your friend could write the cell phone number down for you twice, or even three times, and then you would be much more likely to be able to determine it later. By just repeating the cell number twice, you would be guaranteed to know it if any single digit was erased; in fact, you would know it as long as, for every digit, both copies of that digit were not erased. Unless you expect a great number of messy smudges, though, repeating the number seems like overkill. So here is the challenge – can you write down an eleventh digit that will allow you to correct for any single missing digit? You might first want to consider a slightly easier problem: can you write down an extra number between 1 and 100 that will allow you to correct for any single missing digit?

In this puzzle, we are trying to come up with a code. Generally, a code is used to protect data during transmission from specific types of errors. In this example, a number that is so messy or smudged that you cannot tell what it is would be called an erasure in coding terminology, so our resulting code would be called an erasure code. There are many different types of errors that
can be introduced besides erasures. I might write down (or you might read) a digit incorrectly, turning a 7 into a 4. I might transpose two digits, writing 37 when I meant 73. I might forget to write a number, so you only have nine digits instead of the ten you would expect. There are also codes for these and other more complicated types of errors.

Codes protect data by adding redundancy to it. Perhaps the most basic type of coding is just simple repetition: write everything down two or three or more times. Repetition can be effective, but it is often very expensive. Usually, each piece of data being transmitted costs something – time, space, or actual money – so repeating everything means paying at least twice as much. Because of this, codes are generally designed to provide the most bang for the buck, solving as many or as many different kinds of errors as possible with the least additional redundancy.

This brings us back to the puzzle. If I wanted to make sure you could correct for the erasure of any single digit, I could provide you with the sum of the digits in my phone number. If one digit then went missing, you could subtract the other numbers to find it. For example, if I wrote

\[ 617-555-0123 \]

and you read

\[ 617-555-0?23 \]

you could compute

\[ 35 - (6 + 1 + 7 + 5 + 5 + 5 + 0 + 2 + 3) = 1 \]

to find the missing number.

We can reduce the amount of information passed even further because, in this case, you really do not even need the tens digit of the sum. Instead of writing 35, I could just write down 5. This is because no matter what digit is missing, the sum you will get from all of the remaining digits will be between 26 and 35. You will therefore be able to conclude that the sum must have been 35, and not 25 or smaller (too low) or 45 or larger (too big). Just one extra digit is enough to allow you to handle any single erasure.

It is interesting to consider how helpful this extra information would be in the face of other types of errors. If you misread exactly one of the digits in the phone number, you would see that there was a problem somewhere. For example, if you thought what I wrote was

\[ 617-855-0123 \]

you would see the phone number and the ones digit of the sum did not match, since the sum of the digits in the phone number is 38. In this case, the extra information allows you to detect the error, but it does not allow you to correct the error, as there are many ways a single digit could have been changed to end up with this sequence. For example, instead of

\[ 617-555-0123, \]