Lesson plan: error-correcting codes

Guess who, part 1: 11:20 am - 12:00 pm

1. *QR codes recap*
   (a) **Ask**: What are QR codes used for? Why are they effective? **Answer:** QR codes are special because they can takes an input and masks it through some optical pattern. A computer can then convert it into a sequence of bits. QR codes contain some extra information, which is used to both detect and correct errors. So if a QR code is damaged by dust or dirt, we can usually still scan it. Recall that we could remove some Legos and still scan the QR code.
2. *Redundancy, error correction, and detection*
   (a) **Ask**: What does the word redundancy mean?
   (b) Codewords use this idea of extra data, redundancy, to read damaged data. The big idea is that the extra information counteracts the damaged information.
   (c) We can look out for special properties of codes like interleaving to make sure that a small amount of damage doesn't diffuse into the rest of the code.
   (d) Today we'll be talking about error correction in particular: what makes an error-correcting code work and what properties of codes are best in practice.
3. *Guess Who*
   (a) **Ask**: Has anyone ever played Guess Who before?
   (b) Rules of the game, strategies. A really important question in coding theory is how fast can we win? how many questions should we ask before we know our guess is correct? Or, more concretely, how much data do we need to identify a character?
4. *Round one, no lies*
   (a) Ask the following four questions to a volunteer: Does this character wear a hat or head covering? Does this character wear (sun)glasses? Does this character have an unhappy facial expression? Does this character have facial hair?
5. *Your answers are bits of information!*
   (a) Bits correspond to answers. What is Roger's corresponding bit string?
   (b) Now I can guess who you picked based on the answers you provided. These four questions uniquely determine the person you picked.
6. *But what if someone lied?* **Ask**: Raise your hand if you think we can still determine who the choice was.
7. *Round two, one lie* Find another volunteer, play on phone.
8. *Rounds three and four, one lie/two lies* Try it yourself! Use the link provided.
9. *Transmitting data*
   (a) Note that we only needed to ask four questions in our first game, we needed seven for our second game. In other words, we need seven guesses to correct a single error bit.
   (b) In our third game with two lies, seven questions was not enough to detect and correct two error bits.
   (c) This is called a [7, 4] linear code correcting one error. Think of the 7 as representing the number of questions we need to ask if there is one error, and the 4 represents the number of questions if there is no error.
   (d) After lunch, we'll talk about what kinds of codewords we should be using in practice.

Guess who, part 2: 1:30pm - 3:45 pm

Now that we know what error-correcting codes are, we're going to learn how to build error correcting codes.

1. *Create your own characters*
2. *Instructions*
3. *Properties of successful codes* **Ask**: Raise your hand if you won. Describe your code.
4. *Distance*
   (a) A code that works has a reasonable distance, which is the number of positions in which two codewords differ.
   (b) What is the distance between Roger and Gwen, from our example in the morning?
   (c) The code we just built is a [5, 2] linear code.
5. *What distance works?*
   (a) A commonly known fact in coding theory is that a code can correct t errors if the minimum distance between two codes is at least $2t + 1$. So what is the smallest distance we need to correct one error?
   (b) We also need an upper bound.