1 Noiseless Coding

Noiseless coding is the removal of redundancy in sent information. **Compression** is a good example of this. Noisy coding or **error correction coding** adds redundancy to make things better. First result in noiseless coding is that the **entropy** of a memoryless souce gives a lower bound on the length of a code working on the source. The **average word length** is bounded in terms of the entropy.

3.1 Noiseless Coding

A memoryless source using a set W of source words is anything which throws these words away with prescribed probabilities. ie:

$$w \in W$$

does not depend on what came before it. Formally, it is a sequence: $X_1, X_2, ...$ of **independent adn identically distributed random variables set on a probaility space**. These take values in a set W of source words.
An example,

$$P(X_i = 0) = \frac{1}{2}$$
$$P(X_i = 1) = \frac{1}{2}$$

This is an example of the stream of 0s and 1s and the distribution.
Most general type of source ie. one with no assumptions about interdependence is a **stochastic source**. The other is a **Markov source**. Which I won't expain.
An **alphabet** $\sum$ is just a finite set. The elements fo the set are called **characters** of the alphabet. we denote $\sum^*$ to be the set of all finite strings from characters in $\sum$.
A **code** or **encoding** f of a memoryless source say $S = X_1, X_2, ...$ (emitting source words in a set W) into codeword strings over an alphabet $\sum$ is a map:

$$f : W \to \sum^*$$

Example (Morse Code)

the alphabet here is $\sum dot, dash$ and the collection of 'words' W can simply be the english alphabet. More commonly used words are likely to have shorter encodings.

Example (Ascii)

encoding the english alphabet with numberals, punctuation, and other characters into Ascii (0-255). This is from symbol to numbers of course. Source words

are just single characters rather than 'words' in the traditional sense for $\sum$ we have 4 options binary, octal, decimal, hexadecimal.

In this couse we will only consider **uniquely decipherable** codes. Codes in which two different messages can't be encoded the same way. This is so no information is lost in the encoding. This also means:

$$f : W \to \sum^*$$

is injective. Think like a JPEG file format is loses some information that barely is needed. This means not uniquely decipherable.

$$s = s_1 s_2 \dots s_m$$
$$t = t_1 t_2 \dots t_n$$

in $\sum^*$, say that s is a **prefix** of t if s is an initial piece of t, ie $m \leq n$ and

$$t_1 = s_1, t_2 = s_2, \dots, t_m = s_m$$

a code $f : W \to \sum^*$ is an **instantaneous** or **prefix** code if for all words w, w' in the set W of source words,

$$f(w) \text{ is not a prefix of } f(w') \text{ for } w \neq w'$$

If a code is instantaneous, then it can be decoded without **lookahead**. This means the correct decoding of codewords can be determined without looking ahead or comes after (hence lookahead). Natural languages don't have this. Think blue and bluetooth blue is a prefix in bluetooth.

Example

The code with words $00, 01, 110$, and 001 is not a prefix code. This is because the first codeword, 00 is the prefix to the last 001.

Prefixes are an essential part of Compression and all this. We can add an 'ending character' but that would add more data and less efficiency so these prefixes are important.