Part 7 – A few bits of Information Theory

What do we mean by "Information"?
& how do we measure it?

"They were brusque to the world of manners, they had built their hope of heaven on the binary system and the computer, 1 *and* 0*, Yes and No – "*
Norman Mailer
The Armies of the Night

A few points to be aware of

The material in this part is

a. **NOT** covered in the module textbook (see p. 470 for discussion).

b. **NOT** examinable.

c. Included for sake of interest:
Information Theory is a fundamental topic in CS.

Some motivating background

At the centre of most computational activity there will (eventually) arise a need to **send** "data" (eg text) to another party.
Consider sending an e-mail:

a. We *hit* a **keypad** on a **keyboard** (eg 'S')
b. This action results in the S being added and **stored** in the text.
c. The message itself will be sent through some "**channel**": eg a mobile-network frequency, optical cable, coaxial cable, copper wire,
d. On arrival the message is **processed** (ie the **content** is determined).
e. The text is then "**displayed**" in some form.

What problems arise?

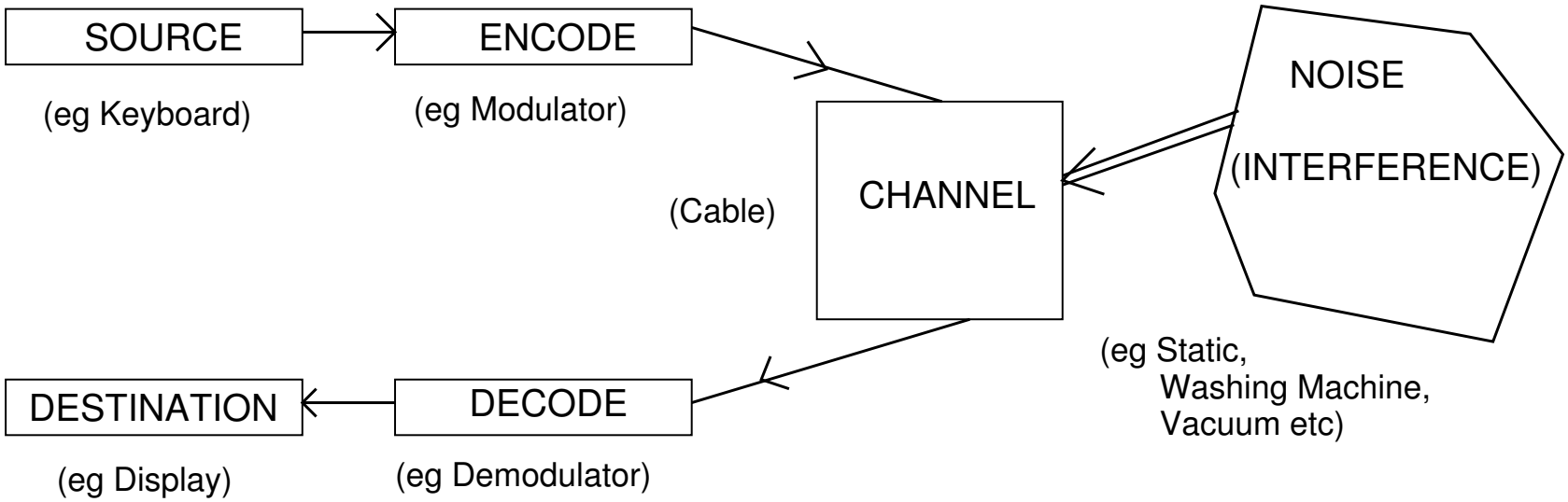
a. The characters (keypads) have to be **encoded** (usually using *binary*).
b. When a message is sent through a channel:
   P1. How much time will it take? ("*transmission rate*")
   P2. What if there is interference? ("*noisy*" channels)

Information Theory

Information Theory is the field of Computer Science that studies questions such as these. It offers robust solutions that answer:

I1. How to define **information content** and **uncertainty** in information.
I2. What (if anything) limits **speed** and **data compression**.
I3. Is it **always** possible reliably to transmit data irrespective of noise?
I4. What steps can be taken to ensure data is **received** correctly?

The Shannon Paradigm for Communication



The model depicted is from
C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423,623–656, July, October 1948
Shannon's paper is the origin of Information Theory as a scientific discipline: it formulates the key issues of interest, the basic supporting framework and derives fundamental results in the field.

Information & Uncertainty

"The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point." C.E. Shannon, *op. cit.*, p. 379

What is information?

When we communicate (in speech, texting, or (proper) writing) the basic units are **words**. A word is just a (recognised by some authority) *sequence* of *symbols* from an *alphabet*.
Thus, at the most elementary level:

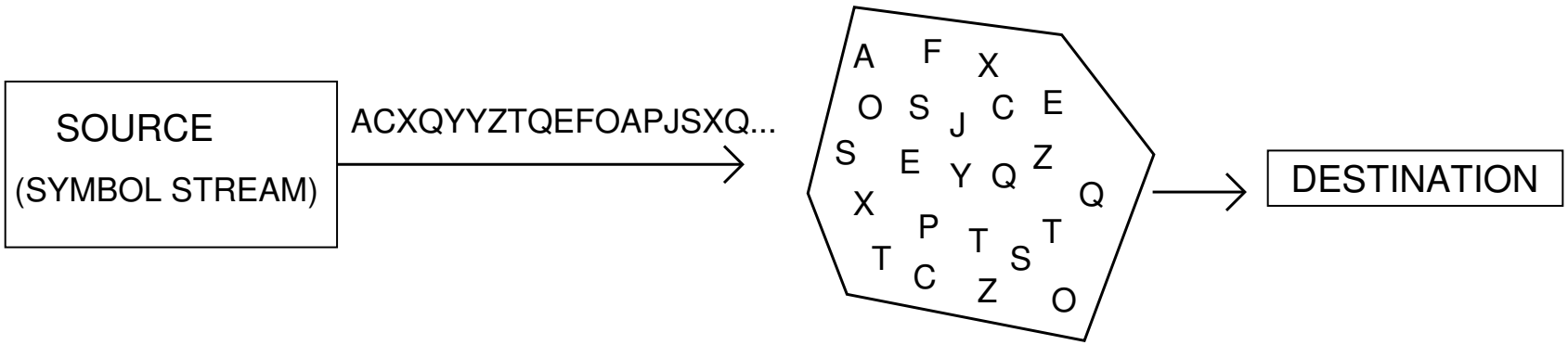
C1. "**communication**" involves "sending" "*information*"

C2. "**information**" is provided as a *sequence of symbols*.

In order effectively to communicate, the receiver must be able to recognise individual symbols in the stream of symbols that have been sent.

Information and the element of "surprise"

Basic view of symbol stream



The receiver will see a stream of symbols:
some will be "more common" than others;
some symbols will be quite rare.
For example, in written English the letters 'E' and 'T' occur often; the letters 'Z' and 'J', however are rarer.
In total a receiver will be "less surprised" to see some symbols and "more surprised" to see others.

Surprise is information gained
Information gained is uncertainty reduced

In English the letter 'Q' is almost always followed by the letter 'U'.
When a receiver sees the letter 'Q' in an input stream of text, "almost surely" the next symbol seen will be 'U'.
Very few words in English begin with the letter 'X'.
So if a symbol for the space character is seen (' ') a receiver would be "surprised" if the following symbol is an 'X'.

In summary we have an **alphabet**, $S = \{s_1, s_2, \ldots, s_n\}$ of n **symbols**. A **message** is a sequence of symbols. We have two questions:

Q1. How do we measure the "uncertainty" (equivalently "degree of surprise" or "information content")?

Q2. How should we "encode" messages (using binary) to achieve desirable properties?

Messages as "sequences of **events**"

In a stream of symbols (as noted earlier) we might expect to see some symbols "very frequently" and others "only rarely".
Thus we can think of attaching a **probability**, P_i, to each symbol s_i.
When the receiver sees s_i "information" has been "gained"
This is because from a state of being "unsure" what the next symbol would be, the observer's "uncertainty" has decreased.

$u_i = -\log_2 P_i$ informally "the decrease in uncertainty when s_i is seen"
ie "the information gained from seeing s_i"

Very small example

If we have an alphabet in which one symbol (s_1 say) is the *only* symbol that ever appears then $P_1 = 1$, so that $u_1 = 0$.
There is **no uncertainty** (only s_1 will appear) and
no increase in information (if anything is seen it can only be s_1).

Shannon's Uncertainty Formula & Limits on Source Coding

What is meant by "Source Coding"?

The symbols we use to compose messages cannot be "sent directly".
eg Alphabetic symbols need to be **encoded** as *binary sequences*.
Various standards (ASCII, UNICODE etc) are used for this purpose.
The (general) **source coding problem** has three principal questions:
For a given source (stream of symbols from an alphabet)

Q1. Is it **possible** to *compress* the number of bits in the data stream?
If this **is** possible

Q2. On average, how many bits are **needed** to represent any symbol?

Q3. What are good algorithms that maximise compression.

A high level of compression increases "throughput" (more data sent in shorter time) **but** "compression" must allow the receiver to **decode** the text (uncompress) "easily".

The Uncertainty Formula & Source Entropy

Symbols appearing "indepedently"

Recall that we have introduce P_i as "the probability of seeing s_i in a stream". For the "simplified" development being given we assume that these are "independent": ie should s_i appear at time T the probability of seeing s_j at time $T+1$ is still P_j.
Shannon derives (**NOT** "defines") the
"**Uncertainty in a data source from alphabet** S" as

$$H(S) \quad = \quad -\sum_{i=1}^{n} P_i \log_2 P_i$$

$H(S)$ is sometimes referred to as the **entropy** of the source S.
Entropy depends on the **distribution** P not the **alphabet** S.

The Source Coding Theorem

For $L(S)$ ("average number of bits to encode a symbol"): $L(S) \geq H(S)$.

Some examples

$S = \{A, B, C, D\}$

If $P_i = 0.25$ for each symbol (ie each is equally likely to be seen) then

$$H(S) \;=\; -4 * (0.25) * \log_2(0.25) \;=\; -\log_2(0.25) \;=\; 2$$

We need at least two bits for some symbol. Could use

$$A = 00 \;\;;\;\; B = 01 \;\;;\;\; C = 10 \;\;;\;\; D = 11$$

With this scheme, sending a stream of 104857600 characters (100Mega) require $200 Mbits$.
What if $P_A = 0.5$, $P_B = 0.25$, $P_C = P_D = 0.125$?
We *could* use the **same** coding scheme as before.
We know, however, that the entropy is now:

$$(-0.5 \log_2 0.5) + (-0.25 \log_2 0.25) + 2(-0.125 \log_2 0.125) = 1.75$$

Exploiting the reduction in entropy

What if we now use the code

$$A = 1 \;\; ; \;\; B = 01 \;\; ; \;\; C = 000 \;\; ; \;\; D = 001$$

Given the symbol probabilities, in our 100Mega stream we "expect" there to be:
104857600/2 = 52428800 As (50Megabits);
104857600/4 = 26214400 Bs (2×26214400, ie 50Megabits)
26214400 Cs and Ds ($3 \times 26214400 \sim$ 75Megabits)
In total we expect to transmit 175Megabits

In order to be effective the estimate of the relative frequency of different symbols must be accurate. For a "large" data file this can be determined exactly so allowing data **compression** with no loss of information.
It is assumed, of course, the receiver will be informed (or know) of the code scheme used.

A Bit more on Source Coding Schemes

The Source Coding Theorem gives a lower bound on the number of bits that might be needed for some symbol (on average).
This lower bound is **optimal** and (to within a small additive term) actually **achievable**.
To do so involves careful study of **coding schemes**.

Coding schemes

A binary code, C, is a set of **codewords**

$$C = \{c_1, c_2, \ldots, c_t\}$$

We think of individual **alphabet** symbols s being assigned (coded) using a particular $C(s) = c_i$ in C

Some Example Codes

Original ASCII

Lower and upper case alphabetic characters, digits, punctuation symbols, so-called "CONTROL (CTRL)" characters (128 in total) are mapped to 7-bit binary codes. For example

Symbol	*ASCII*	*Symbol*	*ASCII*
A	1000001	0	0110000
a	1100001	9	0111001
!	0100001	%	0100101
ESC	0011011		0100000

The set of all 128 binary sequences containing 7 bits defining ASCII is one example of a **prefix code**.

The next stage - Channel behaviour

The preceding slides have considered one aspect of Information Theory: the elements concerning what "happens" when the sender of a message "prepares" the message for transmission.
That is to say, the processes of transforming symbols into binary, how much compression is possible without information being lost, etc.
Although it is over-simplifying, the mechanisms involved when a message is **received** mirror the actions involved in sending: in essence decoding the binary sequence to recover the text.

In the final section we look at what affects the "middle" of this activity: messages are sent through a "channel" (wire, ether, frequency etc).

Q1 How can the **sender** be "sure" the message has arrived "uncorrupted"?

Q2 How can the **receiver** be confident that the message received is what was **intended** to be **sent**?

The effect of "noisy" channels

Some examples of garbled messages

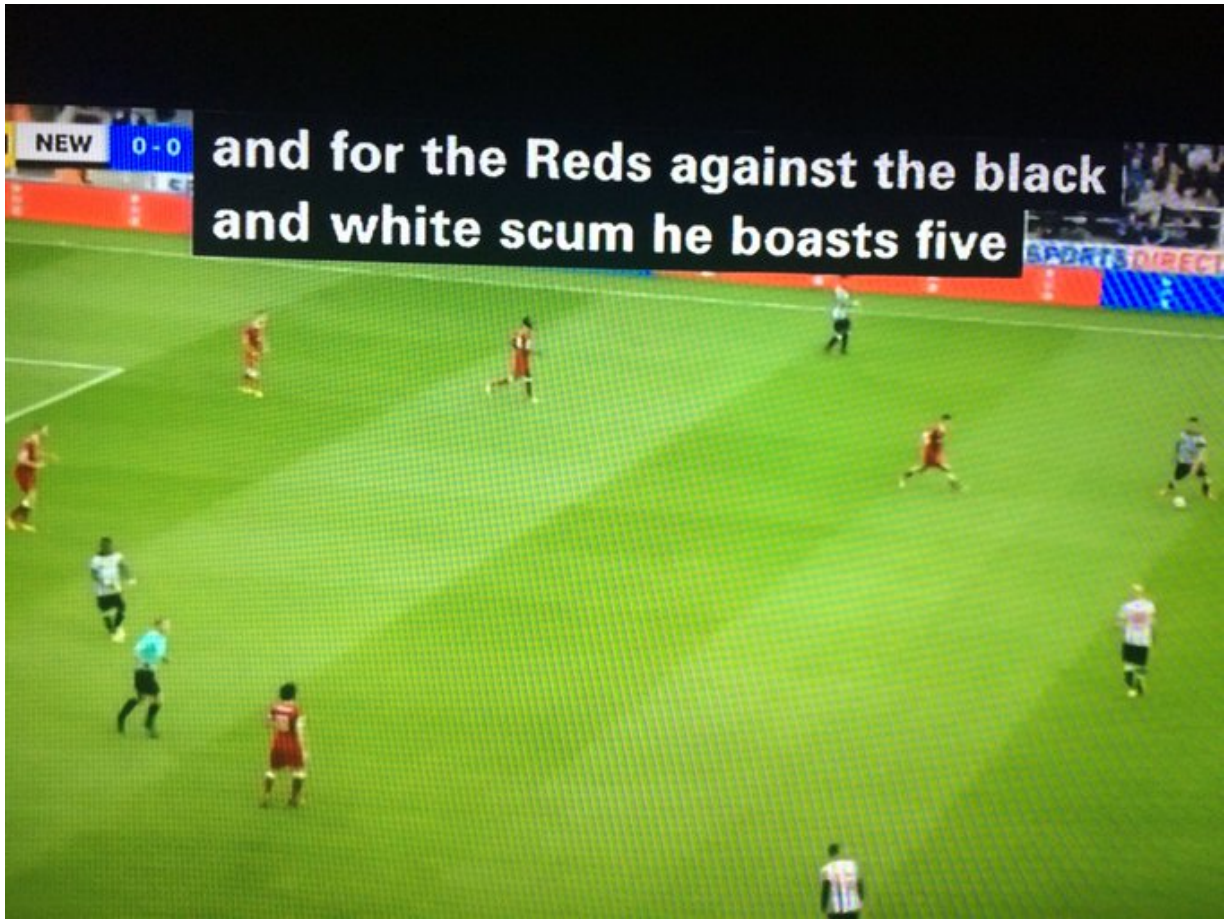
1. "*Send three-and-fourpence we're going to a dance*"
   Field-telephone message as relayed to British army HQ (WW1).
2. Coach-hire company: "*How many people do you need buses for?*"
   Reply: "*aboot six tae seven*"

The "real" messages

1. "*Send reinforcements we're going to advance*"
2. "*about six to seven*" (ie 6–7 **not** 67)

And a more recent example: BBC MOTD2 2/10/2017

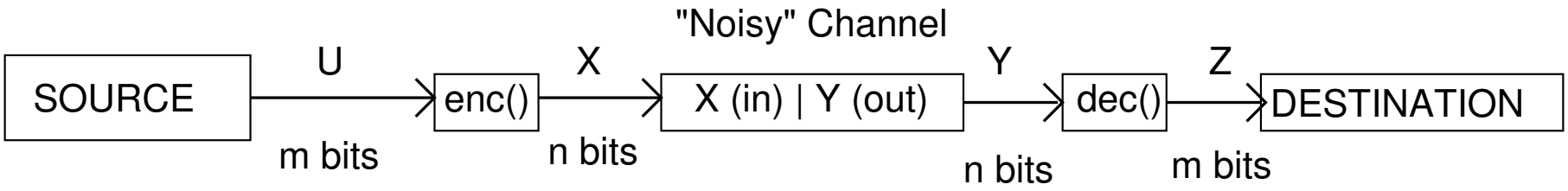
Commentary to subtitle transcription



What was actually said

"*and for the Reds against the black-and-white***s COMMA** *he*"

An abstract model of noisy channel behaviour



Description

After encoding message, U with m-bits some binary sequence ($enc(U) = X$) with n-bits is passed to the "channel"
If "noise" is present what will emerge is some binary sequence (Y, also n-bits) which **may differ** from X.
The word Y is decoded ($dec(Y) = Z$) and Z (m bits long) sent.
We wish to "minimise" the potential for errors by encoding (in n bits) messages that use m bits.
Ideally, n should be "small" relative to m: we don't want to "pad out" the message with "too much" "redundancy".

Noise as the "probability of errors"

The abstraction on the previous slide, in effect, describes the action of the channel as a **function** mapping (encodings, X) of input binary words (U) to transmitted words (Y) that decode to final texts (Z).
Given the nature of "noise" we cannot **predict with certainty** what will appear as Y when X is given as input.
(eg we cannot tell **exactly how long** it will take for the person braying into their mobile phone to scream "*I'm on the TRAIN*")

Binary Channels

We focus on a particular class of "noisy" channel where the stream of source symbols U are $\{0, 1\}$.
A sequence of symbols, U is translated as $X = enc(U)$ for transmission through a channel which outputs Y. This is then used to give $Z = dec(Y)$ as the message received.
Suppose now we have values p $(0 \leq p \leq 1)$ and $0 \leq q < 0.5$ for which:

U. $P[U = 0] = p$; $P[U = 1] = 1 - p$. That is the input is a 0 with probability p; 1 with probability $1 - p$.

Y. Given an input bit X the channel leaves this unchanged with probability $1 - q$ and "flips" (ie negates) this bit with probability q.

Problems when sending m bits

With the scenario of the previous slide if we send U without making any changes (ie $enc(U) = U$) basic probability show us that $\sim qm$ bits will be corrupted by noise in the channel.
If q is "large" (eg very close to 0.5) the message received (Z) is unlikely to resemble the message sent (U).
Thus, P_e the probability of error, ie that $U \neq dec(enc(U))$ will be "quite high".

Avoiding the problem: building redundancy

Now consider an alternative:

E1. Given U, $X = enc(U)$ just "repeats" U, say 3 times, eg if $U = 0$, $enc(U) = 000$, if $U = 011$, $enc(U) = 000111111$.

D1. Z is found by $dec(Y)$ where $dec(Y)$ reports the ith bit of Z ($0 \leq i < m$) as that occurring in the majority of the positions $y_{3i}y_{3i+1}y_{3i+2}$, eg if $X = enc(U) = 000111111$ and $Y = 010101110$ then $Z = dec(Y) = 011$.

Error probabilities and "repetition coding"

In the approach described we reduce the chance of a transmitted text being corrupted by repeating its content and applying a "majority" vote scheme to the result. Before (making no change to U) with "one-bit-at-a-time" transmission through the channel we have $P_e = q$. When we repeat each bit 3 times,

$$P_e = P[\text{At least 2 of the 3 bits are corrupted}] = 3(1-q)q^2 + q^3 < q$$

We reduce the error probability **BUT** at the cost of *reducing the bit rate*
BEFORE: Bit rate with no change to U: $n/n = 1$; $P_e = q$;
AFTER: Bit rate with 3-repeats: $n/3n$; $P_e < q$

The transmission progress depends on the encode-decode convention. Using e for $enc()$ and d for $dec()$ the pair (e, d) is called a **scheme**. The measure of "transmission speed" is called the **scheme rate**, denoted R, and describes the *number of bits* sent each time the channel is used. Messages with m bits are "padded" to n bits: the rate of a scheme is m/n

Information Theory – Summary

1. The material in this part offers only an extremely introductory discussion of the key concerns of Information Theory:
   a. Methods for "encoding" symbols as binary sequences so as to realise properties such "lossless data compression" (*Source Coding*).
   b. Methods for modeling noise and techniques for reducing its effect (*Channel Coding*).
2. Schemes in 1(a) try to **reduce** the number of bits sent in a message. Schemes in 1(b) try to minimize the number of bits **added** in order to send a message reliably.
3. *Coding Schemes* (of both types) are an important (if rather more advanced) area of CS. Some of these (eg Huffman Codes, Liv-Zempel Codes) feature later in the programme on COMP309.