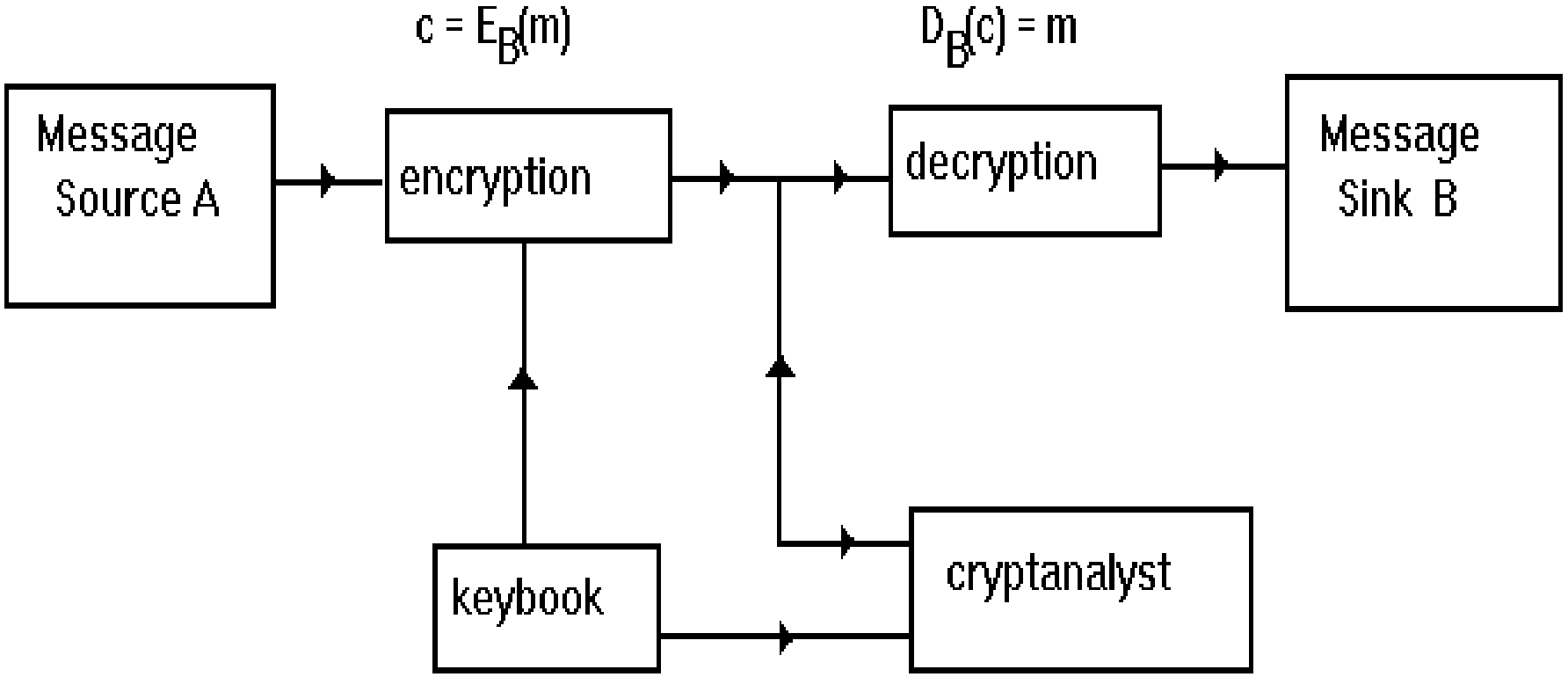
Modern Cryptology

Redundancy and Information Theory

Introduction



In the beginning ...

Early computers were huge mechanical monsters whose reliability was low compared to the computers of today. Based, as they were, on banks of mechanical relays, if a single relay failed to close the entire calculation was in error. The engineers of the day devised ways to detect faulty relays so that they could be replaced. While R.W. Hamming was working for Bell Labs, out of frustration with the behemoth he was working with, the thought occurred that if the machine was capable of knowing it was in error, wasn't it also possible for the machine to correct that error. Setting to work on this problem Hamming devised a way of encoding information so that if an error was detected it could also be corrected. Based in part on this work, Claude Shannon developed the theoretical framework for the science of information theory.

Bell Labs

Quoted from *From Error-Correcting Codes through Sphere Packings to Simple Groups*,Thomas M. Thompson, Carus Mathematical Monographs (MAA) #21 (1983) pp.16-17.

The Model V, the penultimate relay computer to be built by Bell, was the largest in the series. Containing over nine thousand relays and over fifty pieces of teletype apparatus, it occupied about one thousand square feet of floor space and weighed some ten tons. Incidentally, the same computing power three decades later is to be found in the more sophisticated hand-held calculators. (Today, in most common wristwatches.)

Bell Labs

Quoted from *From Error-Correcting Codes through Sphere Packings to Simple Groups*, pp.16-17.

... the Model V ... used various k-out-of-six codes, particularly for input and output. These were transmitted essentially in blocks and permitted extensive self-checking. For example, the input was entered in the machine via a punched paper tape, seven-eighths of an inch wide, which had up to six holes per row. Each row was read as a unit. If the sensing relays expected a two-out-of-six code, they would prevent further computation if more or less than two holes appeared in a given row. Similar checks were used in nearly every step of a computation.

Bell Labs

Quoted from *From Error-Correcting Codes through Sphere Packings to Simple Groups*, pp.16-17.

When such a check failed, two results were possible depending upon whether the machine was set for "daytime" use with operating personnel present or for unattended "nighttime" or "weekend" operation. In the first mode, a check failure stopped the machine and sounded an alarm. In the second, a check failure switched the machine immediately to other work. In the first case, the check failure was located by operating personnel, whose efforts were facilitated by an elaborate check light panel. However, in the second case the problem simply had to be rerun.

Bell Labs

Quoted from *From Error-Correcting Codes through Sphere Packings to Simple Groups*, pp.16-17.

Hamming did not have priority use of the Model V. In fact, he had only weekend access, and that only when another department wasn't using it. “When they didn't need it, I got it.” Furthermore, for weekend use the machine was placed in the unattended mode ... Of this unfortunate circumstance Hamming said:

Two weekends in a row I came in and found that all my stuff had been dumped and nothing was done. I was really aroused and annoyed because I wanted those answers and two weekends had been lost. And so I said, 'Damn it, if the machine can detect an error, why can't it locate the position of the error and correct it?'

A Single Error Correcting Code

This is the simplest code that Hamming devised in his 1947 paper, *Self-Correcting Codes – Case 20878, Memorandum 1130-RWH-MFW*, Bell Telephone Laboratories. [The first paper on error correcting codes]

The information that is to be encoded is a collection of binary strings of length t^2. The code words will be binary strings of length $(t+1)^2$, so $2t+1$ check digits will be added to the information data. To compute the check digits, arrange the data into a $t \times t$ array, add a mod 2 check sum digit to each row and column (including one for the row and column of check sums). The code word is the string obtained by writing out the rows of the extended array.

A Single Error Correcting Code

As a small example, consider the t = 3 case. We would code up the data 011010111 by

$$\begin{matrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{matrix}$$

Calculate the check digits,

and produce the codeword 0110010111111100.

When a word is received, it is arranged in a 4×4 array, the row and column check sums are recalculated and compared with the entries in the last column and last row. Differences indicate the row and column of any single error, which can then be corrected!

A Single Error Correcting Code

Suppose that the codeword 0110010111111100 is sent, but the word 0110011111111100 is received. We decode this:

0 1 1 0 0
0 1 1 1 0 ←
1 1 1 1 1
1 1 0 0 0
1 1 1 0
↑

By recalculating the check sums, we locate the row and column of the error. Clearly, any single error in a data digit will change only its row and column checks and so will be located. If an error is spotted only in a row but not a column (or vice versa) the error has occurred in a check digit and so, can again be corrected.

Redundancy

F C T S S T R N G R T H N F C T N

Fact is stranger than fiction

0^{th} Order Markov Process equal probabilities

XFOML RXKHRJFFJUJ ZLPWCFWKCYJ
FFJEYVKCQSGHYD
QPAAMKBZAACIBZLHJQD
ZEWRTZYNSADXESYJRQY WGECIJJ
OBVKRBQPOZBYMBUAWVLBTQCNIKFMP
MKVUUGB M DM QASCJDGFOZYNX
ZSDZLXIKUD

0^{th} Order Markov Process
English Frequencies

OCRO HLI RGWR NMIELWIS EU LL
NBNESEBYATH EEI ALHENHTTPA
OOBTTVA NAH BRL OR L RW NILI E
NNSBATEI AI NGAE ITF NNR ASAEV
OIE BAINTHA HYROO POER
SETRYGAIETRWCO EHDUARU EU C FT
NSREM DIY EESE F O SRIS R
UNNASHOR

1st Order Markov Process digram frequencies

ON IE ANTSOUTINYS ARE T INCTORE
ST BE S DEAMY ACHIN D ILONSIVE
TUCOOWE AT TEASONARE FUSO
TIZIN ANDY TOBE SEACE CTISBE

2nd Order Markov Process trigram frequencies

IN NO IST LAT WHEY CRATICT
FROURE BIRS GROCID PONDENOME
OF DEMONSTURES OF THE
RETAGIN IS REGOACTIONA OF CRE

0^{th} Order Markov Process word frequencies

REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE HAD MESSAGES BE THESE

1st Order Markov Process word pair frequencies

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHOEVER TOLD THE PROBLEM FOR AN UNEXPECTED

3rd Order Monte Carlo Approximation

The best film on
best film on television
film on television tonight
on television tonight is
television tonight is there
tonight is there no-one
is there no-one here

THE BEST FILM ON TELEVISION
TONIGHT IS THERE NO-ONE HERE WHO
HAD A
LITTLE BIT OF FLUFF

Measuring Entropy

1.Depends only on the probabilities of the output events.
If k possible events have probabilities $p_1, p_2, \ldots, p_k$, then we are trying to define a function: $H(p_1, p_2, \ldots, p_k)$.

2. H should be continuous in each of its variables.

3. In the case of equiprobable events (each probability = 1/k) then H should be a monotonically increasing function of k.

4. The entropy of a compound event should be the weighted sum of the entropies of its constituent simple events.

Theorem (Shannon): The only such functions are

$$\mathbf{H(p_1,p_2,\ldots,p_k) = -\lambda \Sigma\, p_i \log p_i}$$

Entropy and Redundancy

Define:

$$\mathbf{H(p_1,p_2,...,p_k) = - \Sigma\, p_i \log_2 p_i}$$

its units are called ***bits***.

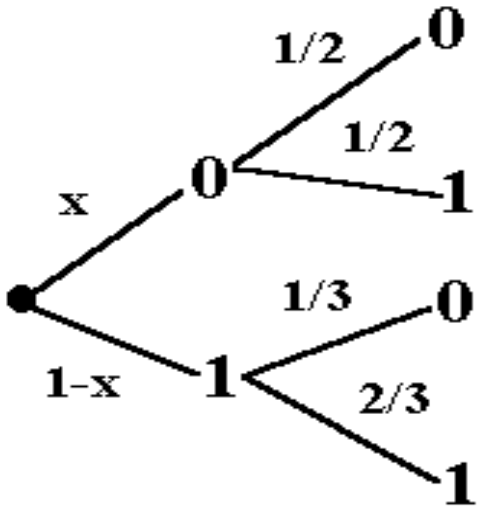
Define:

Redundancy = 1 - ($H/\log_2 k$)

it has a value between 0 and 1.

An Entropy Calculation

1st order Markov process on a binary alphabet where the probability of a 0 following a 0 is 1/2, but following a 1 is 1/3.



Probability of a 0 at Step i = Probability of a 0 at Step i+1

$$x = (1/2)x + (1-x)(1/3)$$

From which it follows that x = 2/5.

An Entropy Calculation

The entropy given a 0, is

$H_0 = -(\frac{1}{2} \log \frac{1}{2} + \frac{1}{2} \log \frac{1}{2}) = 1$

while given a 1 is,

$H_1 = -(1/3 \log 1/3 + 2/3 \log 2/3) = .912$

so for the 1st order process we get:

$H = 2/5\ H_0 + 3/5\ H_1 = .4(1) + .6(.912) = .9472$ bits/letter.

For the 0th order process we would obtain:

$H = -(2/5 \log 2/5 + 3/5 \log 3/5) = .97095$ bits/letter

Redundancy Again

The redundancy of a 0^{th} order Markov process on two symbols with equal probability is 0 (since H = 1).

Various methods for approximating the entropy of English give the value of about 1 bit/letter. With $\log 27 \approx 4$, we have the redundancy = .75 (75%)