

Information

Experience shows that the **information value** of certain *news* depends on their probability.

$$I_{E_i} = f(P_i)$$

in which I_{E_i} means the information value. In this aspect the more unexpected or unlikely (rumour) a *news* is the bigger its *news* value.

So the f function was selected according to Shannon's suggestion:

$$I_E = -\log_2 \frac{1}{p_E} = -\log_2(p_E) \text{ [bit]}$$

Shannon used the logarithm to measure information because only the logarithmic function makes the information of independent events additive. If two independent events A and B occur, their joint probability is: $p(A, B) = p(A) \cdot p(B)$.

We expect that the total information should add up:

$$I(A, B) = I(A) + I(B)$$

The logarithm satisfies this property:

$$I(p) = -\log p \quad \rightarrow \quad I(A, B) = -\log(p(A)p(B)) = I(A) + I(B)$$

If we used $I(p) = 1/p$, the values would multiply rather than add.

Example: three coin tosses

Consider three independent tosses of a fair coin. The probability of heads on each toss is $(1/2)$. The probability of getting heads on all three tosses is:

$$p(H, H, H) = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

If we used $I(p) = 1/p$, learning that one toss resulted in heads would give an information value of:

$$I(H) = \frac{1}{1/2} = 2$$

Adding the information values of the three individual outcomes would give $(2+2+2=6)$. However, applying the same formula to the combined outcome gives:

$$I(H, H, H) = \frac{1}{1/8} = 8$$

Therefore, this formula does not make information additive: $(8 \neq 2+2+2)$. Instead, the values multiply: $(8 = 2 \cdot 2 \cdot 2)$.

With the logarithmic formula, learning the result of each coin toss provides **1 bit** of information:

$$I(H) = -\log_2(1/2) = 1 \text{ bit}$$

Learning that all three tosses resulted in heads provides:

$$I(H,H,H) = -\log_2(1/8) = 3 \text{ bits}$$

This is exactly the sum of the information from the three individual outcomes: $1 + 1 + 1 = 3 \text{ bits}$. We obtain the same total information whether we learn the results one by one or all at once.

The properties of the logarithm function play an important role in modeling the quantitative properties of information.

If an event space consist of two equal-probability event ($p(E_1) = p(E_2) = 0.5$) then,

$$I_{E_1} = I_{E_2} = \log_2 \frac{1}{0.5} = -\log_2 0.5 = 1 \text{ [bit]}$$

So the unit of the information means the news value which is connected to the simple, less likely, same probability choice.

If the event system consist of n number of events and all these events have the same probability then the probability of any event is the following:

$$p_E = \frac{1}{n}$$

In these cases, the news value which is connected to these events will be the following:

$$I_E = \log_2 \frac{1}{p_E} = \log_2 (n) \text{ [bit]}$$

Entropy

If the events in the event space are not equally likely, then the set of messages can be well characterized by the average information content of the messages.

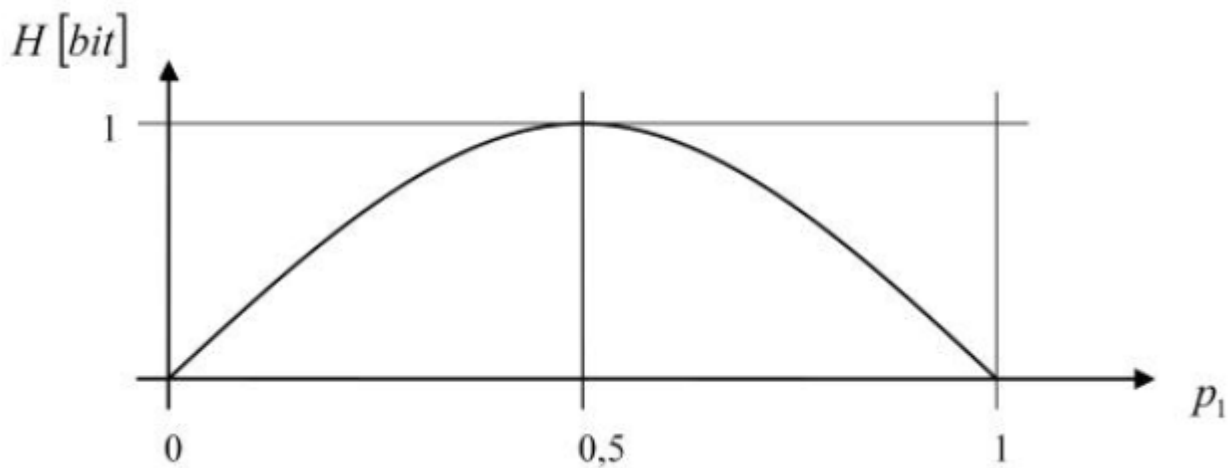
The average information content of the set of messages is called the *entropy* of the message set.

$$H_E = \sum_{i=1}^n p_i \cdot I_{E_i} = \sum_{i=1}^n p_i \cdot \log_2 \frac{1}{p_i} = -\sum_{i=1}^n p_i \cdot \log_2 p_i \text{ [bit]}$$

Example: Given an event space consisting of two events: ($E = \{E_1, E_2\}$), and further ($p = \{p_1, p_2\}$) with ($p_2 = 1 - p_1$), then the average information content is:

$$H_E = -[p_1 \cdot \log p_1 + (1 - p_1) \cdot \log (1 - p_1)]$$

This function can be represented as follows:



For two possible outcomes, entropy is highest when both have probability 0.5. In this case, we are most uncertain about which outcome will occur.

Entropy decreases as one outcome becomes more likely and the other becomes less likely. If one outcome is certain, entropy is zero: observing the result provides no new information.

Unlikely outcomes do not necessarily mean entropy is low. For example, if there are 256 equally likely outcomes, each has a probability of only $1/256$, but the entropy is 8 bits. What matters is the whole probability distribution, not just the probability of one outcome.

Example:

- If a source always sends the same letter ("AAAAA..."), the entropy = 0 bits → there is no new information.
- If every letter occurs with equal probability (e.g., random characters), the entropy is maximal → the source is rich in information.

This concept is crucial in various fields, including *data compression*, *cryptography*, and *machine learning*, where understanding and managing entropy can lead to more efficient algorithms and systems. For example, lossless data compression represents the same information using fewer bits by exploiting redundancy in its representation. It does not reduce the information that must be preserved: the original data can be reconstructed exactly.

For an intuitive example, you can describe a sequence of 1000 identical letters as "1000 copies of A" instead of writing out every letter. The representation becomes shorter, but no information is lost.

Redundancy

The average information content of a message set describing an equally probable, completely random set of events is the highest. In contrast, the average information content of a message set describing a completely ordered, i.e., fully known event set, is the lowest.

A probability distribution that deviates from the maximum possible entropy leads to a message set that is redundant.

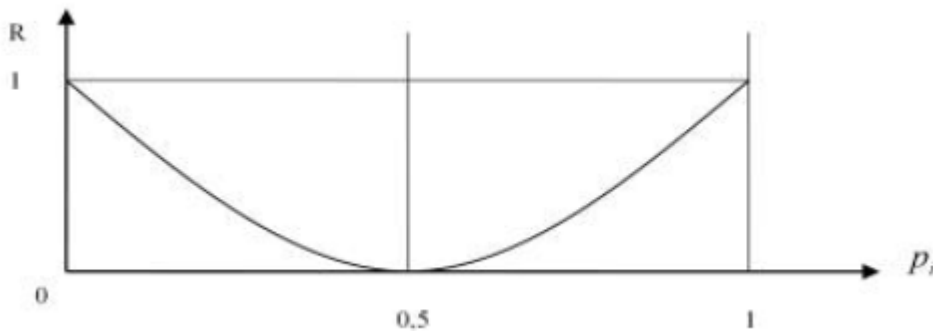
The measure of redundancy is:

$$R = \frac{H_{\max} - H}{H_{\max}} = 1 - \frac{H}{H_{\max}}$$

If the event space consists of n equally probable events:

$$H_{\max} = \log_2 n \quad ; \quad R = 1 - \frac{H(p_1, \dots, p_n)}{\log_2 n}$$

Redundancy plays a significant role in information theory. Redundancy enables the secure communication of messages over a noisy channel. The redundancy of human verbal communication is typically more than 30%. The changes in redundancy for a two-event message set are illustrated in the figure below:



Thus, redundancy is minimal when the probabilities of the events are equal.

Example: The occurrence probabilities of a system consisting of four events are as follows:

$$E = \{E_1, E_2, E_3, E_4\},$$

and the probabilities of the individual events are as follows:

$$p = \{0.5, 0.25, 0.2, 0.05\}.$$

The individual information content for the states of the system are:

$$I_{E_1} = -\log_2 0.5 = 1 \text{ [bit]}, \quad I_{E_2} = -\log_2 0.25 = 2 \text{ [bit]}, \quad I_{E_3} = -\log_2 0.2 = 2.32 \text{ [bit]}, \quad I_{E_4} = -\log_2 0.05 = 4.32 \text{ [bit]}$$

What is the entropy of the message set?

$$H_E = \sum_{i=1}^4 p_i \cdot I_{E_i} = 0.5 \cdot 1 + 0.25 \cdot 2 + 0.2 \cdot 2.32 + 0.05 \cdot 4.32 = 1.68 \text{ [bit]}$$

What is the redundancy of the message set?

Let's calculate the maximum entropy:

$$H_{\max} = \log_2 n = \log_2 4 = 2 \text{ [bit]}$$

Then, substituting into the formula for redundancy:

$$R = 1 - \frac{H}{H_{\max}} = 1 - \frac{1.68}{2} = 0.16,$$

which means the redundancy of the event system is approximately 16%.

Example of Entropy calculation

Why do not store raw password strings in compiled code?

```
#include <stdio.h>
#include <math.h>
#include <string.h>

float calculateEntropy(unsigned char counts[], int length);

int main(void) {
    const char sample[] =
        "Some poetry types are unique to particular cultures and genres "
        "and respond to yQ%v?FY}ZT=/5cJ.m~A{9^8L the characteristics of the
language in which the "
        "poet writes. Readers accustomed to identifying poetry with Dante, "
        "Goethe, Mickiewicz, or Rumi may think of it as written in lines "
        "based on rhyme and regular meter. There are, however, traditions, "
        "such as Biblical poetry and alliterative verse, that use other "
        "means to create rhythm and euphony. Much modern poetry reflects "
        "a critique of poetic tradition, testing the principle of euphony "
        "itself or altogether forgoing rhyme or set rhythm.";

    const int windowWidth = 20;
    unsigned char counts[256];

    int sampleLength = strlen(sample);

    for (int start = 0; start <= sampleLength - windowWidth; start++) {
        memset(counts, 0, sizeof(counts));

        // Count characters in current window
        for (int j = 0; j < windowWidth; j++) {
            unsigned char c = (unsigned char)sample[start + j];
            counts[c]++;
        }

        float entropy = calculateEntropy(counts, windowWidth);
        printf("%d - %.3f\n", start, entropy);
    }

    return 0;
}

float calculateEntropy(unsigned char counts[], int length) {
    float entropy = 0.0f;
    for (int i = 0; i < 256; i++) {
        if (counts[i] > 0) {
            float freq = (float)counts[i] / length;
```

Last update:

2026/10/05 17:15 tanszek:oktatas:techcomm:information <https://edu.iit.uni-miskolc.hu/tanszek:oktatas:techcomm:information?rev=1791220545>

```
        entropy -= freq * log2f(freq);  
    }  
}  
return entropy;  
}
```

From:

<https://edu.iit.uni-miskolc.hu/> - Institute of Information Science - University of Miskolc

Permanent link:

<https://edu.iit.uni-miskolc.hu/tanszek:oktatas:techcomm:information?rev=1791220545>

Last update: **2026/10/05 17:15**

