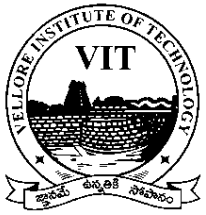


VIT-AP UNIVERSITY

Natural Language Processing

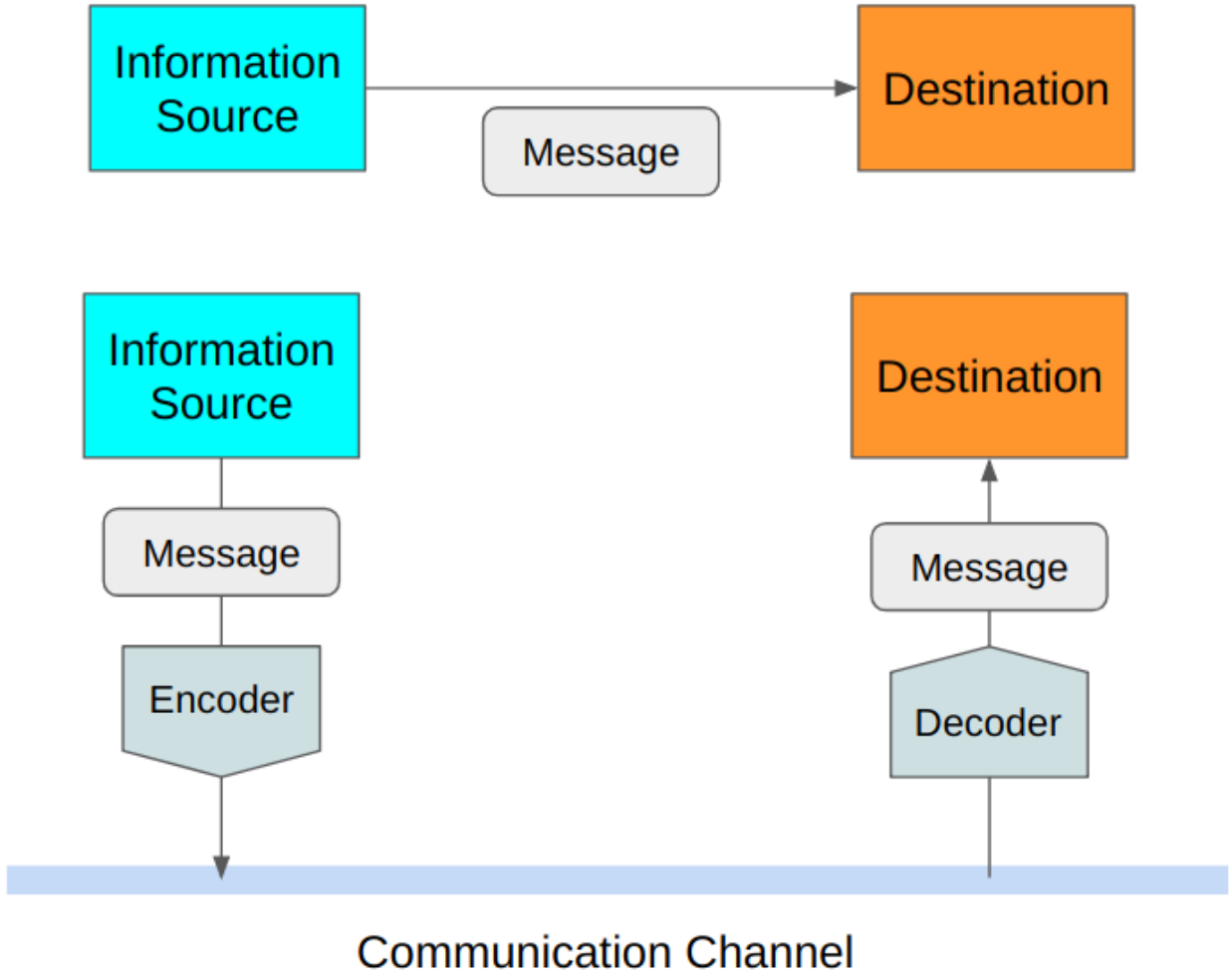
(Course Code: CSE 3015)

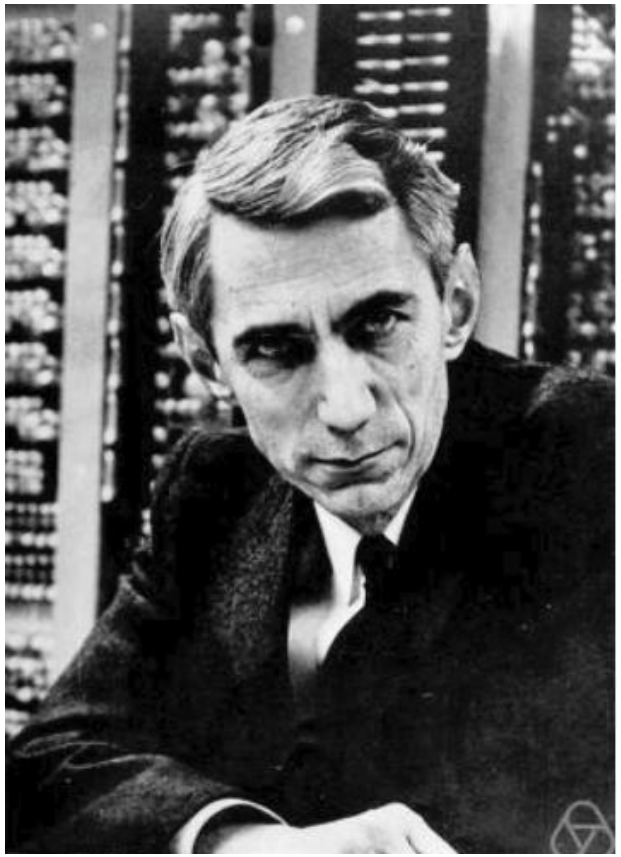
Module-1:Lecture-5: Entropy and Cross Entropy

Gundimeda Venugopal, Professor of Practice, SCOPE

Mathematical Theory of Communication

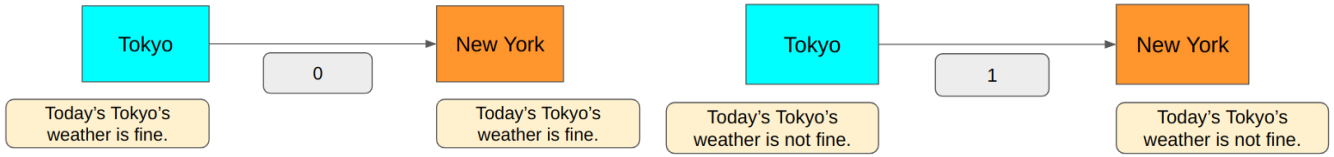
- In 1948, Claude Shannon introduced the concept of information entropy in his paper "A Mathematical Theory of Communication".
- Shannon defined entropy as the smallest possible average size of lossless encoding of the messages sent from the source to the destination. He showed how to calculate entropy, which is valuable for using the communication channel efficiently.

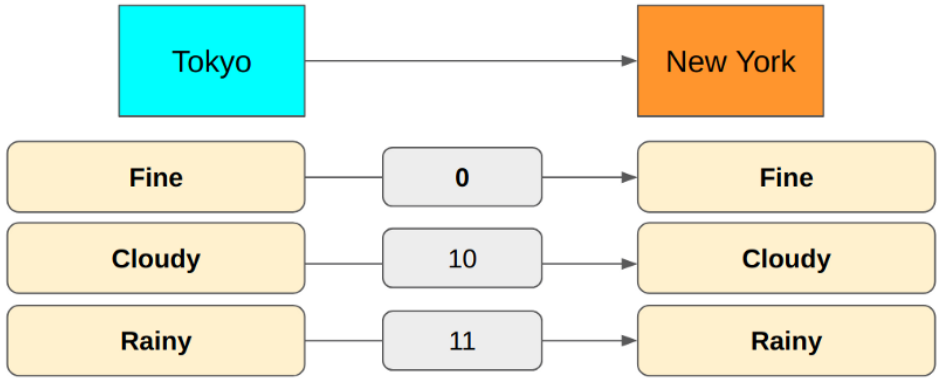




How to Make Efficient and Lossless Encoding?

- City Weather Information Encoding using text is inefficient
- Encode using Binary [‘0’ for “Fine”] or [‘1’ for “Not Fine”]
- Is it Lossless? Yes. Decode without losing any info using the msg received (‘0’ or ‘1’)

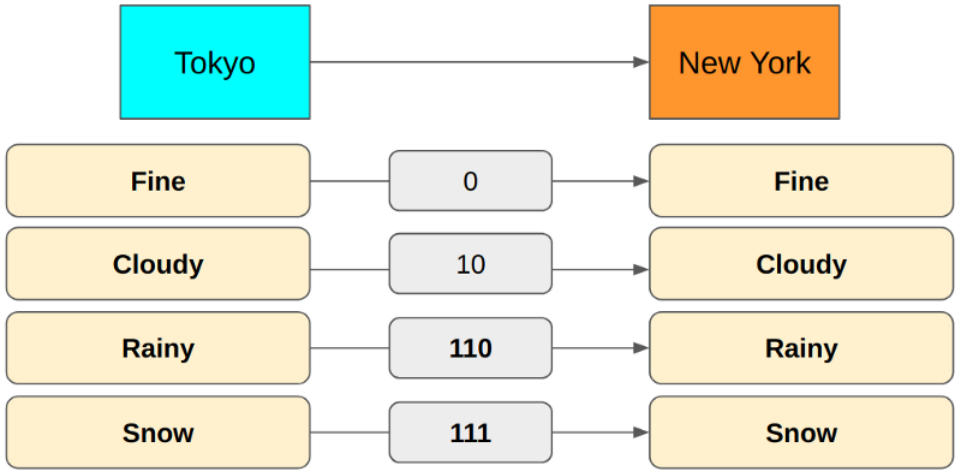




The first bit indicates Fine or not in the above encoding. The second bit indicates Cloudy or Rainy.

But there is a problem. It can snow in Tokyo, and we cannot communicate with such weather.

What do we do? We could add a third bit for Rainy and Snow cases.



We changed Rainy from 11 to 110 and added Snow as 111 because 110 distinctly differs from 111, but 11 is ambiguous when multiple encodings are transmitting one after another.

For example, if we use 11 for Rainy and 111 for Snow, a 5-bit value of 11111 could mean either Rainy, Snow or Snow, Rainy.

Because of this, we use 110 for Rainy so that we can use 110111 for Rainy, Snow and 111110 for Snow, Rainy. There is no ambiguity.

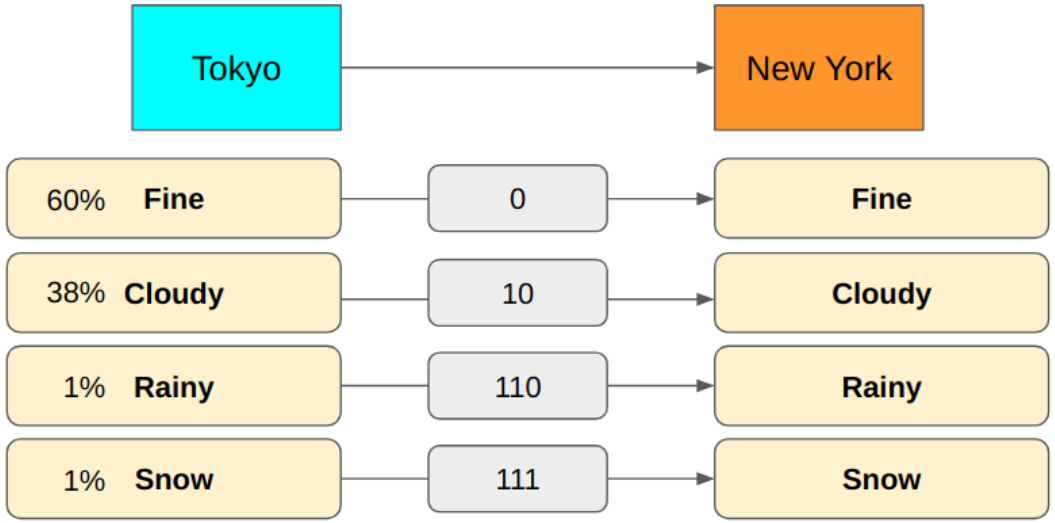
So, the encoding now uses the first bit for Fine or not, the second for Cloudy or others, and the third for Rain or snow.

For example, **010110111 means Fine, Cloudy, Rainy, Snow**.
110010111 means Rainy, Fine, Cloudy, Snow.
Once again, there is no ambiguity. It is lossless, and it looks pretty efficient.

How to Calculate Average Encoding Size?

❖Example 1

Suppose Tokyo sends weather messages to New York often (say, every hour) using the lossless encoding

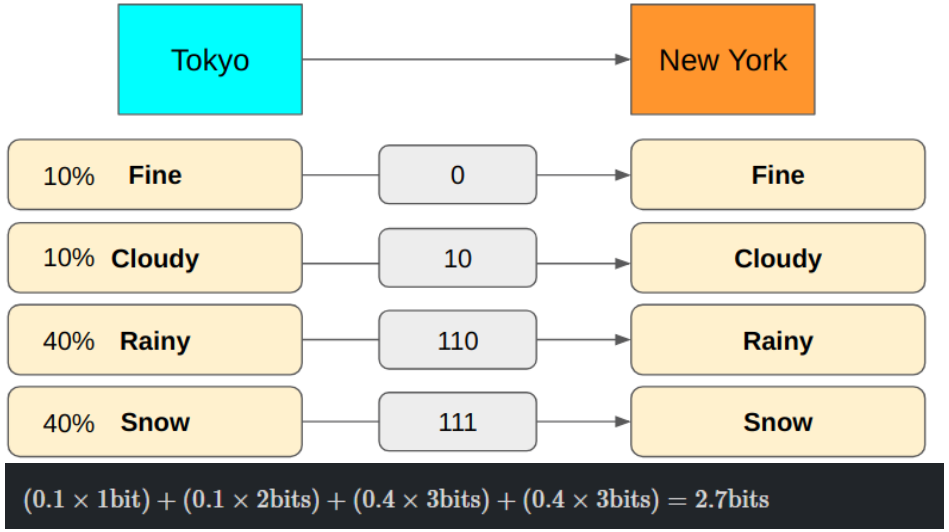


Then, we can calculate the average number of bits used to send messages from Tokyo to New York.

$$(0.6 \times 1\text{bit}) + (0.38 \times 2\text{bits}) + (0.01 \times 3\text{bits}) + (0.01 \times 3\text{bits}) = 1.42\text{bits}$$

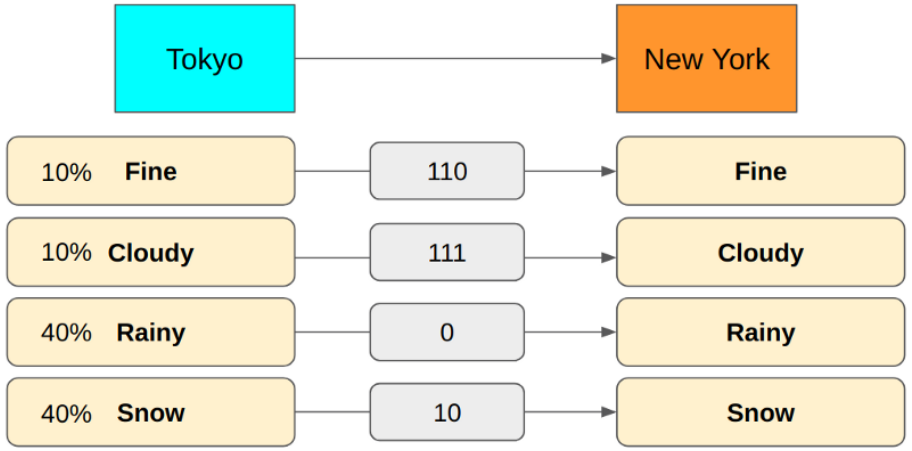
So, with the above encoding, we use 1.42 bits on average per message. As you can see, the average number of bits depends on the probability distribution and the message encoding (how many bits we use for each message type).

❖Examples 2 and 3



$$(0.1 \times 1\text{bit}) + (0.1 \times 2\text{bits}) + (0.4 \times 3\text{bits}) + (0.4 \times 3\text{bits}) = 2.7\text{bits}$$

We would use 2.7 bits to encode messages on average.
We could reduce the average encoding size by changing the encoding to use fewer bits for Rainy and Snow.



Now, We would use 1.8 bits to encode messages on average.

Entropy Calculation Examples

Example 1

Suppose we have eight message types, each happening with equal probability (18=12.5).
What is the minimum number of bits we need to encode them without any ambiguity?

12.5%	A	000
12.5%	B	001
12.5%	C	010
12.5%	D	011
12.5%	E	100
12.5%	F	101
12.5%	G	110
12.5%	H	111

We can not reduce any bit from any of the message types. If we do, it will cause ambiguity. Also, we don't need more than 3 bits to encode eight different values.

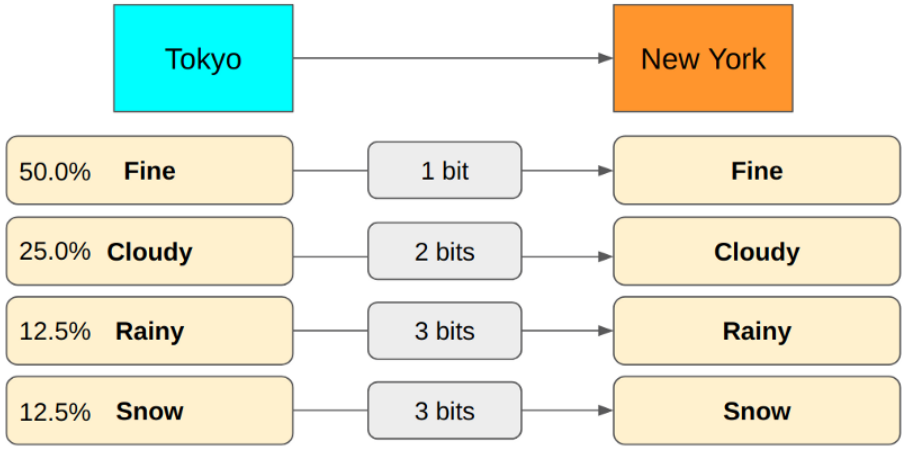
$$\log_2 8 = \log_2 2^3 = 3 \text{ bits}$$

In general, when we need N different values expressed in bits, we need

$$\log_2 N = -\log_2 \frac{1}{N} = -\log_2 P$$

$\log_2 N$ In other words, if a message type happens 1 out of N times, the above formula gives the minimum size required. As $P = \frac{1}{N}$ is the probability of the message, the same thing can be expressed as:

Example 2



$$P(\text{Fine}) = 0.5$$
$$-\log_2 P(\text{Fine}) = -\log_2 0.5 = -\log_2 1/2 = \log_2 2 = 1 \text{ bit}$$

$$P(\text{Cloudy}) = 0.25$$
$$-\log_2 P(\text{Cloudy}) = -\log_2 0.25 = -\log_2 1/4 = \log_2 4 = 2 \text{ bits}$$

$$P(\text{Rainy}) = 0.125$$
$$-\log_2 P(\text{Rainy}) = -\log_2 0.125 = -\log_2 1/8 = \log_2 8 = 3 \text{ bits}$$

$$P(\text{Snow}) = 0.125$$
$$-\log_2 P(\text{Snow}) = -\log_2 0.125 = -\log_2 1/8 = \log_2 8 = 3 \text{ bits}$$

So, the entropy is:

$$\text{Entropy} = -\sum_i P(i) \log_2 P(i)$$

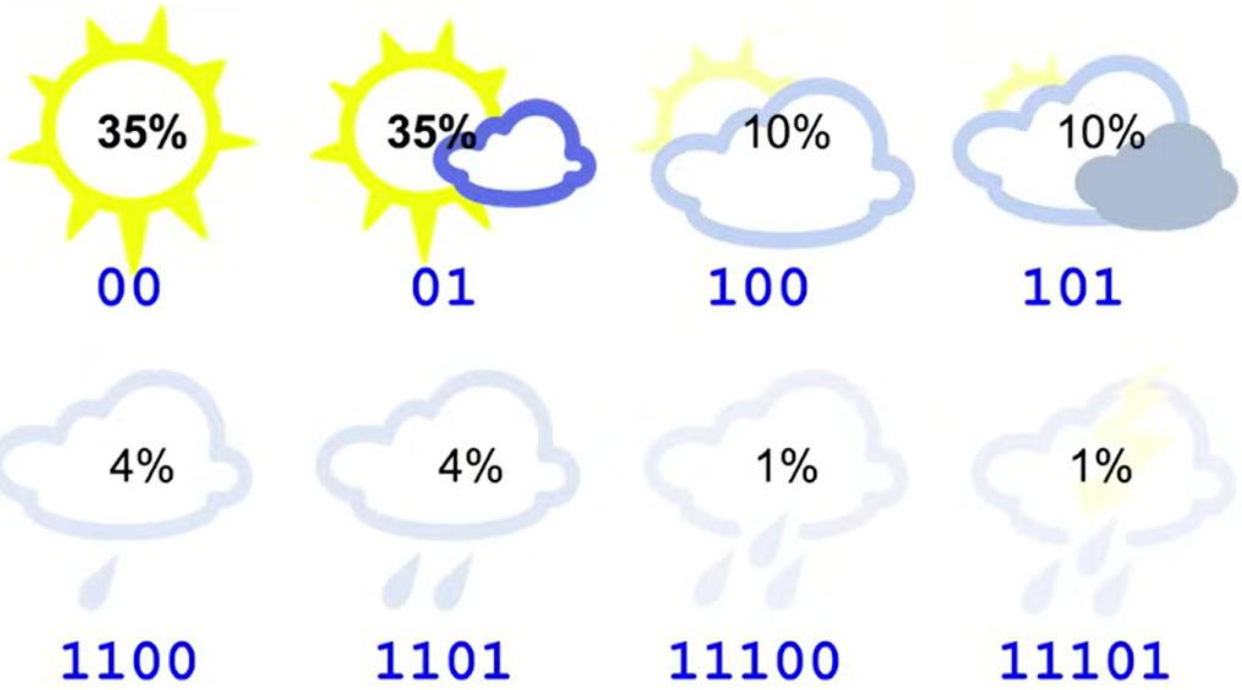
minimum average encoding size (in bits)

$$(0.5 \times 1\text{bit}) + (0.25 \times 2\text{bits}) + (0.125 \times 3\text{bits}) + (0.125 \times 3\text{bits}) = 1.75\text{bits}$$

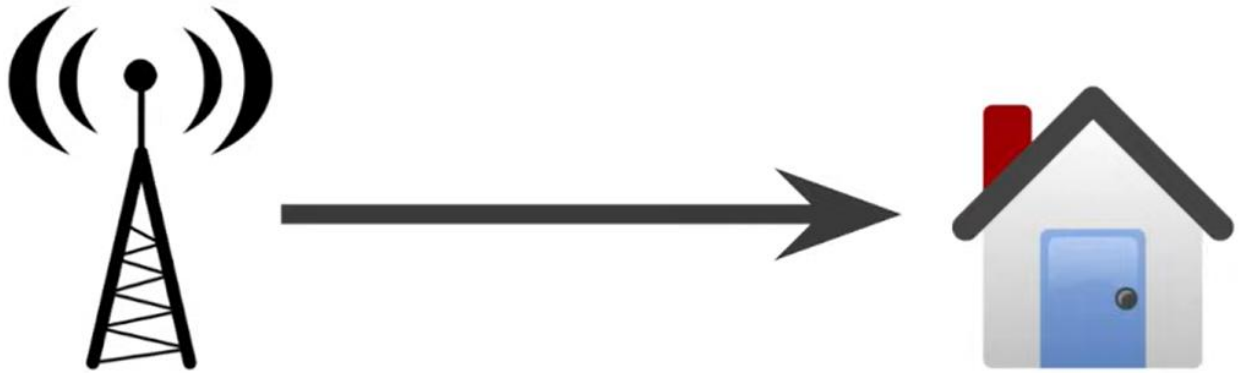
Cross Entropy

Entropy for a True Distribution

Assume we designed a coding scheme for a particular city say Mumbai

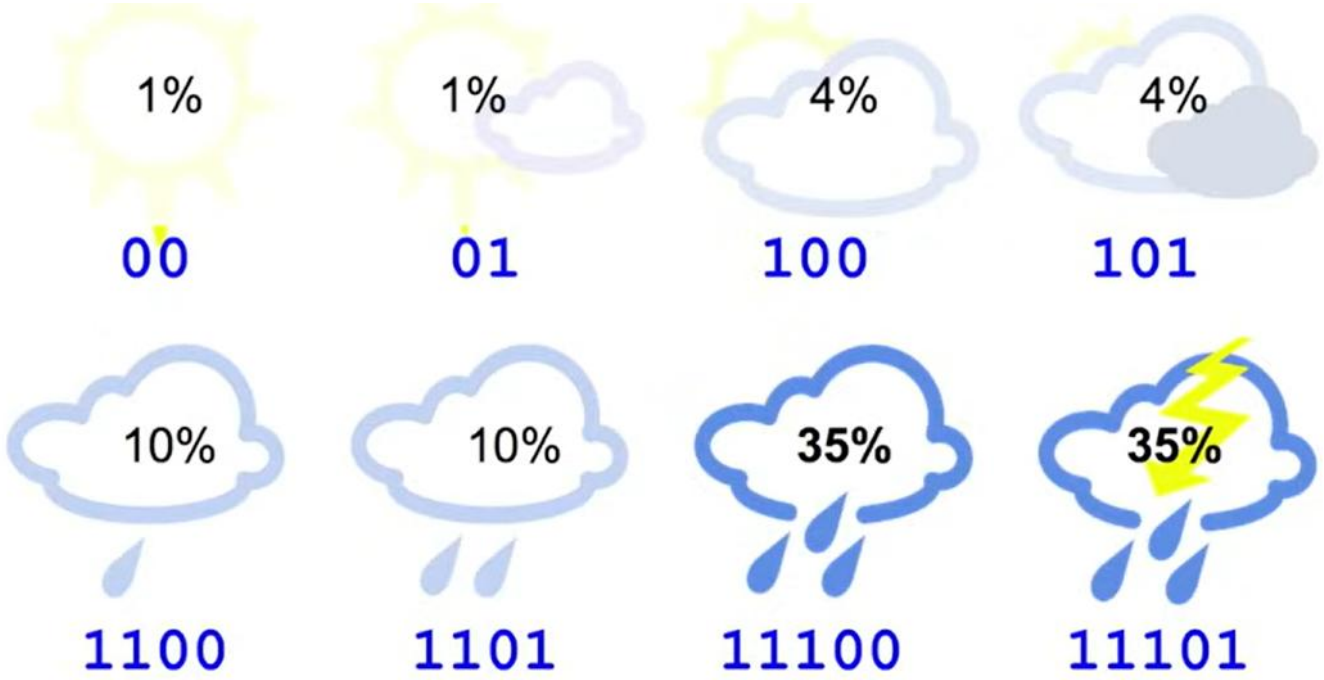


$$35\%\times2 + 35\%\times2 + 10\%\times3 + 10\%\times3 + 4\%\times4 + 4\%\times4 + 1\%\times5 + 1\%\times5 = \mathbf{2.42\ bits}$$

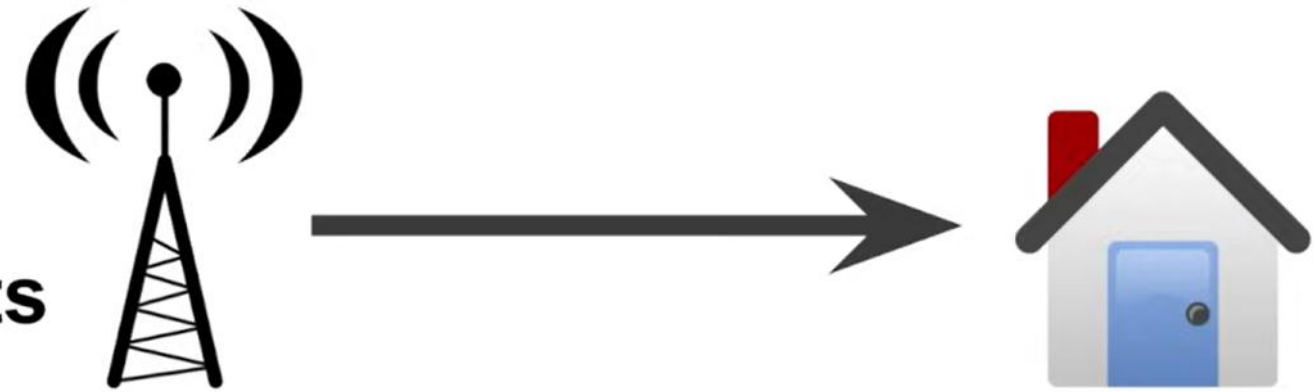


Entropy for a Predicted Distribution

Assume we use the same coding scheme for a particular city say Chirrapunji



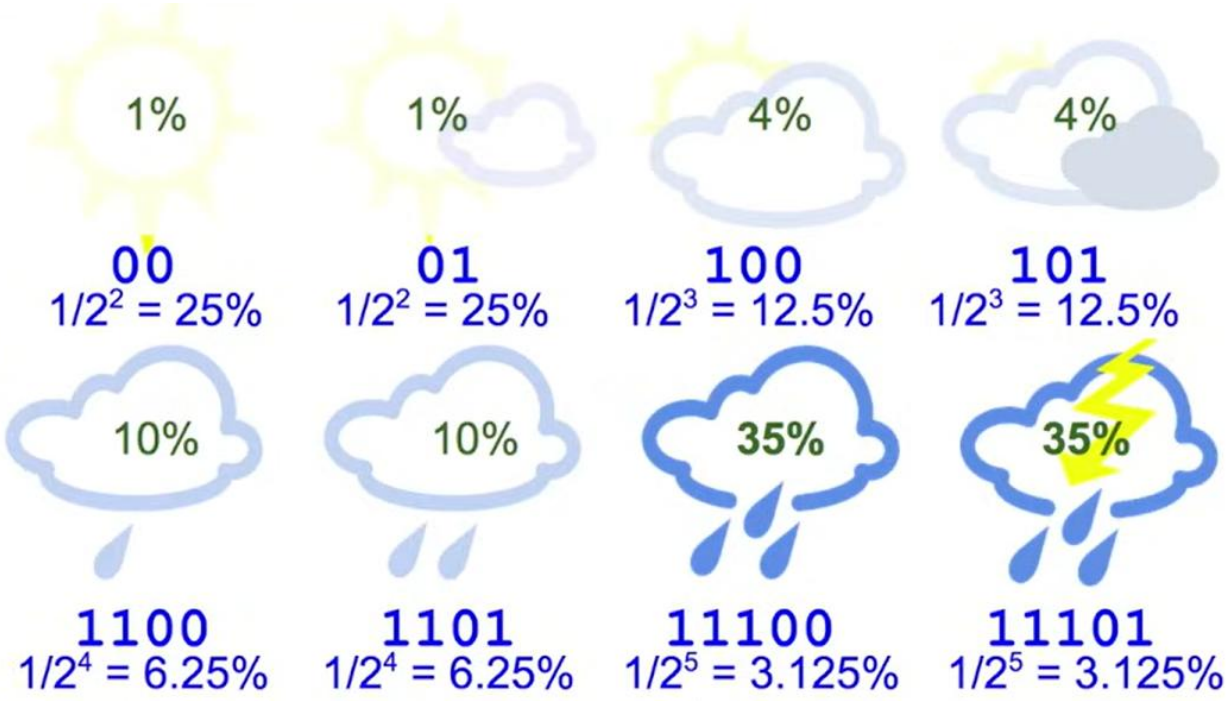
$1\%\times2 + 1\%\times2 + 4\%\times3 + 4\%\times3 + 10\%\times4 + 10\%\times4 + 35\%\times5 + 35\%\times5 = \mathbf{4.58\ bits}$



Cross Entropy

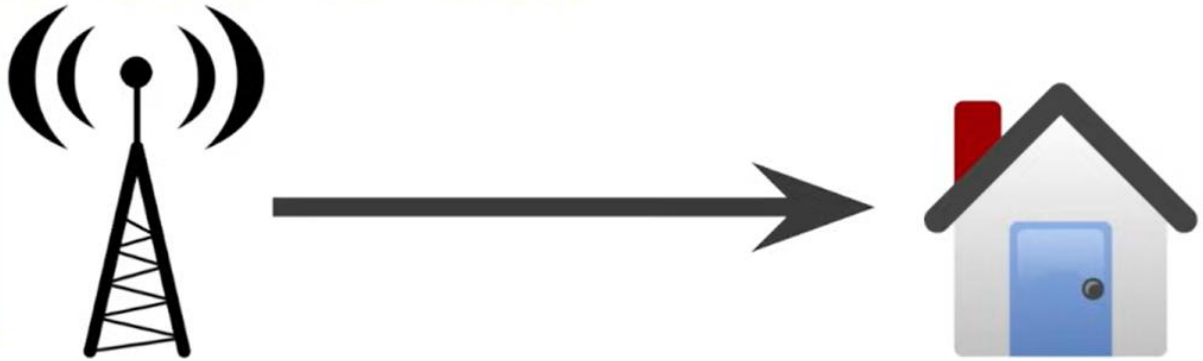
p = true distribution

q = predicted distribution



Cross-Entropy:

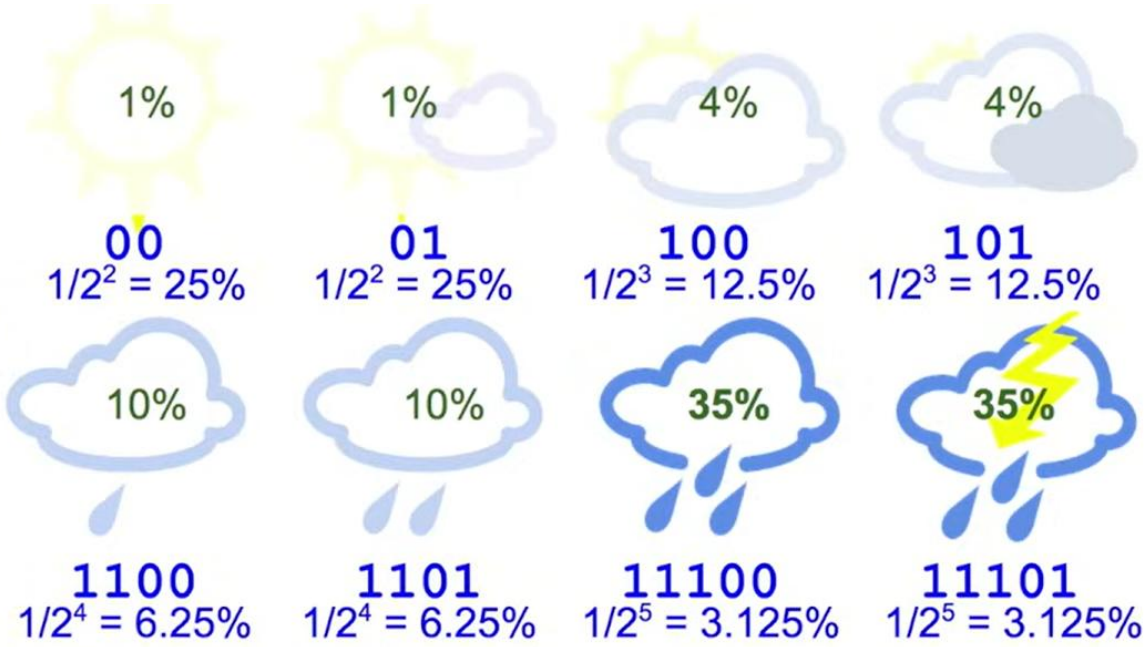
$$H(\mathbf{p}, \mathbf{q}) = -\Sigma_i \, p_i \log_2(q_i)$$



Kullback-Leibler Divergence (K L Divergence)

p = true distribution

q = predicted distribution

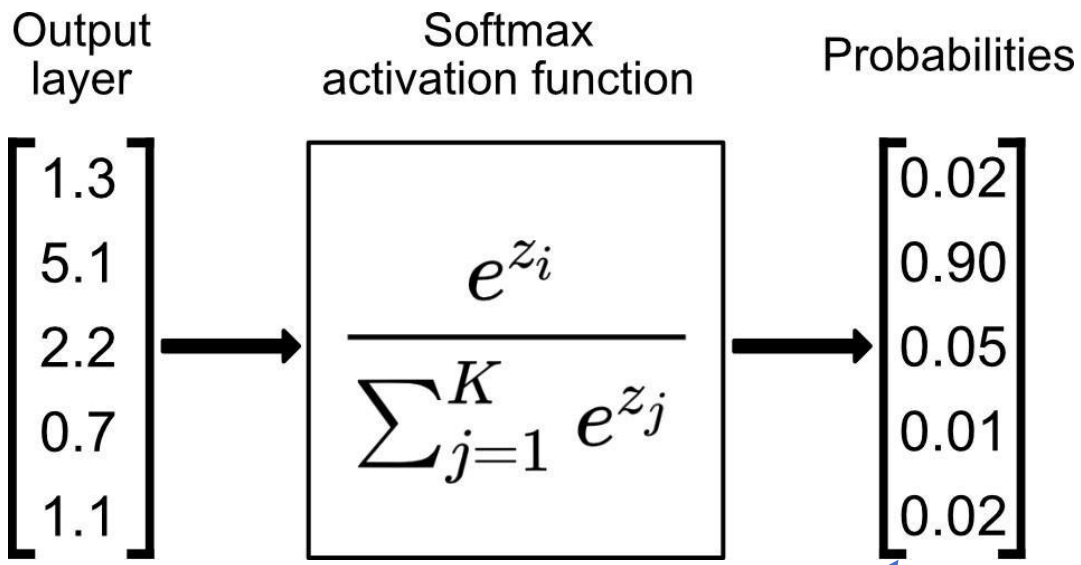


Cross-Entropy = Entropy + KL Divergence

KL Divergence:

$D_{KL}(\mathbf{p} \,||\, \mathbf{q}) = 4.58 - 2.23$

Softmax function Layer



A probability distribution (sum of all probabilities = 1)

```
output_layer = np.array([1.3, 5.1, 2.2, 0.7, 1.1])
output_layer>>> array([1.3, 5.1, 2.2, 0.7, 1.1])

exponentiated = np.exp(output_layer)
exponentiated>>> array([ 3.66929667, 164.0219073 ,
9.0250135 , 2.01375271, 3.00416602])

probabilities = exponentiated / np.sum (exponentiated)
Probabilities>>> array([0.02019046, 0.90253769, 0.04966053,
0.01108076, 0.01653055])

probabilities.sum()
>>> 1.0
```

Source: towardsdatascience.com

Cross Entropy as a Loss/Cost Function

	Cat	Dog	Fox	Cow	Red Panda	Bear	Dolphin
True distribution:	0%	0%	0%	0%	100%	0%	0%
Predicted distribution:	2%	30%	45%	0%	25%	5%	0%

Log Loss

Cross-Entropy Loss:

$$H(\mathbf{p}, \mathbf{q}) = -\Sigma_i\, p_i \log(q_i)$$
$$= -\log(0.25) = 1.386$$

Classifier

$$\log_2(x) = \log(x) / \log(2)$$

References

- A Short Introduction to Entropy, Cross-Entropy and KL-Divergence

- https://kikaben.com/entropy-demystified/
- https://kikaben.com/cross-entropy-demystified/