Introduction to Information

by Erol Seke

For the course “`Data Compression`”

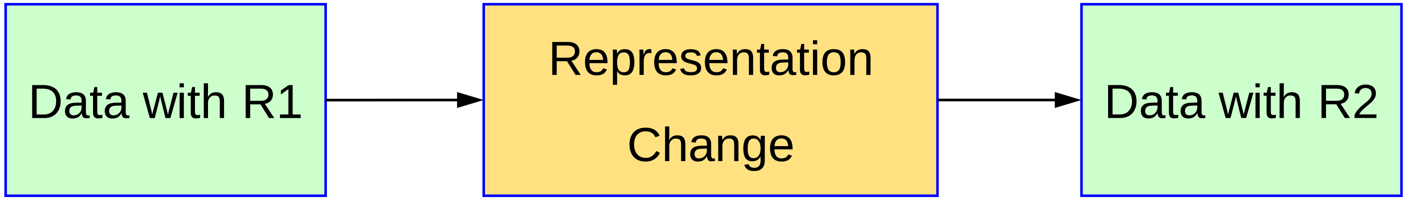
ESKİŞEHİR OSMANGAZİ UNIVERSITY

The Goal

Represent information from using lesser amount of data that the original

It is actually a representation change

Representation Changes



Examples

idea → words → speech/voice

states → bits → electrical signals

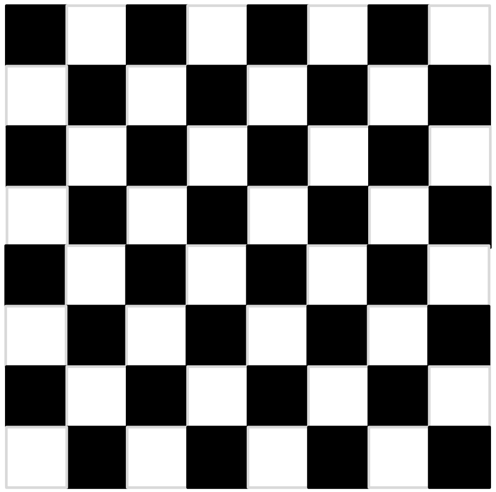
voice → electrical signals → binary data → binary data

several representation changes may occur before obtaining output data

here is an example from Communications course

idea → words → speech/voice → electrical signals → bits → electrical signals

Chessboard



Question: how would you describe a chessboard?

method-1

1. Describe a black square
    1. Tell what a square is
    2. Give size
    3. Describe what black is
    4. Describe what filling with black is
    5. …
2. Describe a white square
3. Describe an array of black & white squares
4. Describe a filling pattern

method-2

1. Describe what a pixel is
    1. Describe white pixel
    2. Describe black pixel
2. Describe creating a square array of pixels
3. Describe an array of black & white squares
4. …

method-3

1. …
2. …
3. …

method-n

1. Tell the word "chessboard" because everybody knows what it is

Question: how about this one?

Question: how much detail would you require to give?

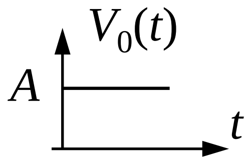
Question: Will you be giving all the information about cracks, scratches, view, ?

Question: What is information ?

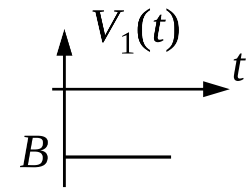
Representation Example

States

True ⟶ represented by 0 ⟶ represented by



False ⟶ represented by 1 ⟶ represented by



Fact 1 : if it is always ‘True’, then nobody needs to share this information, that is there is no information to share

Fact 2 : Information user must know what the representation mean
(speak same language/symbols/signals etc)

Simple Example

A sentence : "The sun will rise tomorrow"

meaning : The star that the earth rounds around will continue to exist and earth will continue to spin and no catastrophic event will occur to prevent that and our side of the earth will complete a considerable part of its rotation cycle, facing towards the sun. (probability=1)

The opposite of the above event has the probability of 0.

It turns out that there is no point of sharing this sentence as it does not contain any information unless the sentence has some epic meaning. For other meanings, of course, both sides must *speak the same language*.

So, what is *information*?

Information and Data

Fact : In order for an event to be counted as informative event, its probability must be between (0,1) excluding both ends

So: To have a probability within (0,1) a complementing probability (opposite of the event) must exist

* So that the occurring event might change in the future

* So that the representative data might change in the future

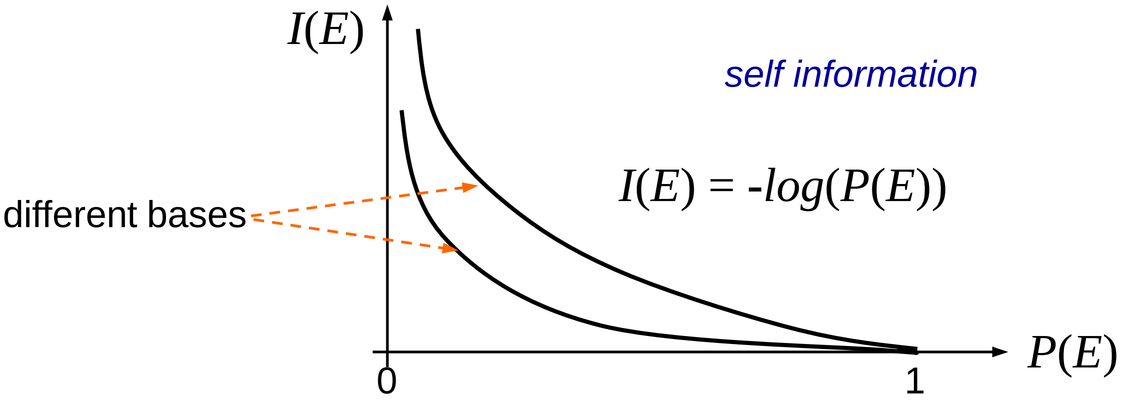
hate him when he ignores punctuations

Information

$P(E)$ $\qquad$ $I(E)$

"Stocks will drop 0.5% tomorrow" $\quad$ low information (happens everyday)

"Stocks will drop 25% tomorrow" $\quad$ high information (rarely happens)



information is [unit]less quantity.
But in order to compare quantities we use the base of the logarithm as if it is a unit

$I(E) = -log_2(P(E))$ [bits] (information value in bits)

Example

Expected grades in Communications Course (approximately)

AA : 5%
BA : 10%
BB : 15%
CB : 20%
CC : 20%
DC : 15%
DD : 5%
FF : 10%

$$I_{\mathrm{AA}} = -\log_2(P_{AA}) \quad = -\log_2(0.05) \sim 4.32 \text{ bits}$$

$$I_{\mathrm{CC}} = -\log_2(P_{CC}) \quad = -\log_2(0.2) \sim 2.32 \text{ bits}$$

.. so on

Meaning : When someone said "I got an AA", he/she actually transferred 4.32 bits worth of information to us.

Question: How much information does he/she transfer by telling all the grades?

Answer: SumOf(All_Info) = Info_Student1 + Info_Student2 + ...
= Number_of_students X Average_Info_Per_Grade?

Question: What is the Average_Info_Per_Grade?

Average Information Per Source Output

Information Generator $\longrightarrow$ info in symbols (like AA,BA, etc.)

Since we know the probabilities, we can calculate

$$I_{avg} = \sum_{n=1}^{N_{sym}} p_n I_n$$ (weighted average)

N_{sym} : the number of possible grades (8 in our example)

$$I_{avg} = -\sum_{n=1}^{N_{sym}} p_n \log_2(p_n)$$

we give it a special name : *entropy of the source*

which depends only on the symbol probabilities

and denote it as $H(z)$ where $z = \{p_n, n = 1, \ldots, N_{sym}\}$

(in our example z={0.05, 0.1, 0.15, 0.2, 0.2, 0.15, 0.05, 0.1})

Examples

We have 2 possible events : H, T with equal probabilities (like a coin drop)

$$I_H = -\log_2(0.5) = 1 \quad \text{and} \quad I_T = -\log_2(0.5) = 1 \quad \text{bit}$$

$$H(z) = I_{avg} = \sum_{n=1}^{2} p_n I_n = 0.5\times 1 + 0.5\times 1 = 1 \quad \text{bit per symbol}$$

H can be represented by binary 0 T can be represented by binary 1

Coin Drop $\rightarrow$ tell the truth in binary symbols

0 : Heads

1 : Tails

Question: What if the coin is not a fair one (probs are not equal)?

example : $z=\{0.25, 0.75\}$

$$I_H = -\log_2(0.25) = 2 \qquad I_T = -\log_2(0.75) \cong 0.415 \quad \text{bits}$$

oops, how do we use 0.415 bits ?

Examples

We have 8 possible symbols with equal probabilities of 0.125 each

$$I_s = -\log_2(0.125) = 3$$ bits for each symbol (logical)

These can well be { 000, 001, 010, 011, 100, 101, 110, 111 }

or { 0, 1, 2, 3, 4, 5, 6, 7 } or { a, b, c, d, e, f, g, h } or ...

The point is : the symbols do not need to be represented in binary
(although their info can be measured in bits)

However : we prefer binary since we use it all the time (in all digital systems).
But that does not prevent us to create symbols like "01011" which
might conveniently be represented by 01011 bit sequence.

Question: What if the symbol information values are not integers?

Answer: No problem. That all depends on what we want to do with
them or how we represent them.

Extensions

Extensions are constructed by having symbols together (side by side or in a bag)

example $A = \{0,1\}$

then $B = \{000, 001, 010, 011, 100, 101, 110, 111\}$ (fixed length)

is a 3rd extension of binary alphabet A

Why ? : To have more symbols to have more efficient representations

Probabilities of newly created symbols are

$$u = \{ p_{000}, p_{001}, p_{010}, p_{011}, p_{100}, p_{101}, p_{110}, p_{111} \}$$

$$p_{abc} = p_a p_b p_c$$

example $z = \{0.25, 0.75\}$

$$p_{011} = p_o p_1 p_1 = 0.25 \times 0.75 \times 0.75 \cong 0.14$$

Extensions

Neither extensions nor original alphabet needs to have fixed length codes

example alphabet constructed of fixed length extensions of binary alphabet symbols

$$B = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

example alphabets constructed from variable length extensions of binary alphabet symbols

$$C = \{00, 01, 011, 1011, 101, 11001, 110, 111\}$$

$$D = \{0, 1, 10, 11, 100, 101, 110, 111\}$$

We can have infinite number of alphabets representing the same source symbol-set

Question : So, What are their differences, advantages, disadvantages etc?

Coding : Representations with Other Symbol Sets

Symbol	Code-1	Code-2	Code-3	Code-4	Code -5	Code -6	
s_1	000	0	1	1	0	00	
s_2	001	1	01	10	01	01	
s_3	010	10	001	100	011	10	...
s_4	011	11	0001	1000	0111	110	
s_5	100	100	00001	10000	01111	111	

fixed length

variable length codes

Coding Representing symbols (or a sequence of symbols) from a symbol set with symbols (or a sequence of symbols) from another set

example

abc... $\longrightarrow$ 123...

it is also good to have

123... $\longrightarrow$ abc...

Question: Why are we doing it?

Answer: For efficient representation

Average Code Length

p_i	Symbol	Code-1	Code-2	Code-3	Code-4	Code -5	Code -6	
0.36	s_1	000	0	1	1	0	00	
0.18	s_2	001	1	01	10	01	01	
0.17	s_3	010	10	001	100	011	10	...
0.16	s_4	011	11	0001	1000	0111	110	
0.13	s_5	100	100	00001	10000	01111	111	

$$L_{avg} = \sum_{n=1}^{N_{sym}} p_n l_n = 3$$ bits for Code-1

$$L_{avg} = \sum_{n=1}^{N_{sym}} p_n l_n = 2.29$$ bits for Code-6

so, using Code-6 is better

Why not use Code-2 then? It looks like it will result a shorter average code length

Because Code-2 is not uniquely decodable
when transferred consecutively

123... $\not\longrightarrow$ abc...

Unique Decodability

Let us have an information source generating symbols from the alphabet

$$A = \{s_1, s_2, s_3, s_4, s_5\}$$

with the probabilities of u={ 0.36, 0.16, 0.17, 0.16. 0.13 }

Assume that the source has generated the sequence of $s_1 s_2 s_3 s_1 s_1 s_5 s_4$

Coding the symbols with Code-2, we would have : 0, 1, 10, 0, 0, 100, 11

or a binary sequence of : 01100010011

We would like to decode the sequence 01100010011 back to $s_1 s_2 s_3 s_1 s_1 s_5 s_4$

remembering that we do not have symbol separators, we see that it is impossible to decode it back to original

So, the Code-2 is not uniquely decodable (that means it is nearly useless)

Unique Decodability

How about using Code-6 on the same source

Sequence is $s_1 s_2 s_3 s_1 s_1 s_5 s_4$

Code-6 coder output : 00, 01, 10, 00, 00, 111, 110

binary sequence without separators: 0001100000111110

On the receiver side we would like to decode the sequence 0001100000111110 back

0001100000111110

0: not in table, take another bit from the stream. Remaining : 01100000111110
00: in table, so output S_1
0: not in table, take another bit from the stream. Remaining : 100000111110
01: in table, so output S_2
1: not in table, take another bit from the stream. Remaining : 0000111110
10: in table, so output S_3

...so on and so forth, up to the end of the stream

Therefore, Code-6 is uniquely decodable although the symbols are variable length

Corollary

We need to have uniquely decodable codes with lower (than original) average code-lengths

Let us examine the previous code table again

p_i	Symbol	Code-1	Code-2	Code-3	Code-4	Code-5	Code-6	
0.36	s_1	000	0	1	1	0	00	
0.18	s_2	001	1	01	10	01	01	
0.17	s_3	010	10	001	100	011	10	...
0.16	s_4	011	11	0001	1000	0111	110	
0.13	s_5	100	100	00001	10000	01111	111	

Code-1 : uniquely decodable, Lavg = 3, fixed-length

Code-2 : **not** uniquely decodable, Lavg = ?, variable-length, not instantaneous*

Code-3 : uniquely decodable, Lavg = x, variable-length

Code-4 : uniquely decodable, Lavg = x, variable-length, not instantaneous*

Code-5 : uniquely decodable, Lavg = x, variable-length, not instantaneous*

Code-6 : uniquely decodable, Lavg = 2.29, variable-length

The code is considered instantaneous if the symbols can be determined when their last bits are received

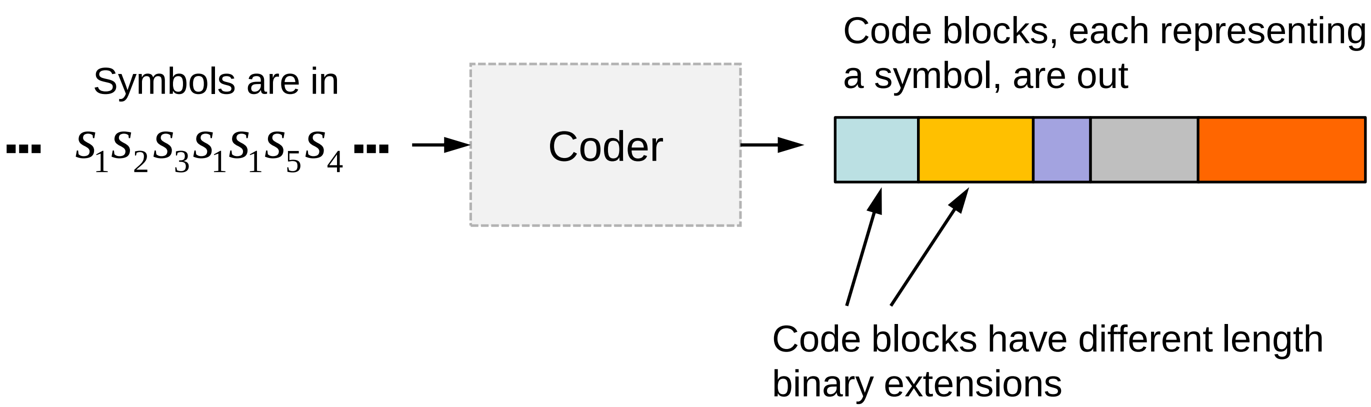
Minimum Average Code Length

We see that we can have infinite number of Codes that are uniquely decodable.
We also need to have efficient representation (smaller average code length)

Question : Is there a way to find a code with minimum average code length?

Answer: Yes for block codes

block code : symbol-to-binarycode representation
(implying that there are other (non-block) codes as well)



Example

Symbols	Prob.s	New symbols /code
00	0.49	?
01	0.21	?
10	0.21	?
11	0.09	?

We would like to determine a code for each symbol, which, for the given probabilities, best represents the self-information of the symbol.

A method : divide the pre-ordered set of probabilities into two so that sum of probabilities on both sides are as close as possible. Continue doing that until there is only one in each division

p_i	prefixes		
0.49	0		
0.21	1	0	
0.21	1	1	0
0.09	1	1	1

prefix sides with 0 or 1

and do it again

Generated code table

00	0
01	10
10	110
11	111

Now we have code for each input symbol to replace with

Code is variable length and uniquely decodable. The method is called *Shannon-Fano*

Example

p_i	code table	
0.49	00	0
0.21	01	10
0.21	10	110
0.09	11	111

$$L_{avg} = \sum_{i=1}^{4} P(s_i)L(s_i) \qquad 0.49\times 1 + 0.21\times 2 + 0.21\times 3 + 0.09\times 3 = 1.81$$

a single input bit is now represented by
1.81 / 2 = 0.905 bits

Notice that this distribution is actually 2nd extension of the ensemble (A, z) where $z = \{0.7, 0.3\}$

Shannon states that "wider the extension, better the representation"

Let us now test this argument with the 2nd extension of the 2nd extension (or the 4th extension of the original binary alphabet)

Example

Probs	code table	
0.2401	0000	00
0.1029	0001	010
0.1029	0010	0110
0.1029	0011	0111
0.1029	0100	1…
0.0441	.	1…
0.0441	.	1…
0.0441	.	1…
0.0441	.	1…
0.0441	.	1…
0.0441	.	1…
0.0189	.	1…
0.0189	.	1…
0.0189	.	1…
0.0189	.	1…
0.0081	1111	1…

hmw: complete the table

L_{avg} =3.5948 bits/symbol

a single input bit is now represented by 3.5948 / 4 = 0.8987 bits

we see that it is getting **better**

The entropy is $H(u)$ = 3.5252

so, we still have room for improvement

It is guaranteed that extensions of n>4 will have better representations

But cannot be lower than H

Huffman

It is proven that Huffman's code generates smallest ACL among dictionary-based statistical block codes

Example

	Probs
s_1	0.49
s_2	0.21
s_3	0.21
s_4	0.09

Let these symbols with smallest probabilities be a single symbol s_5

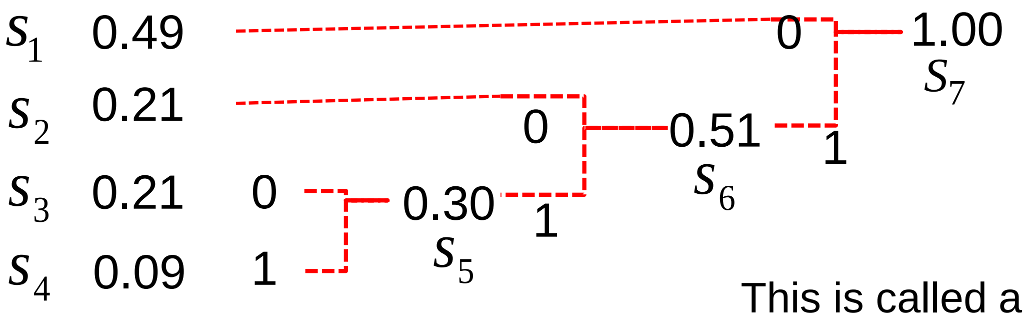
Its probability would be 0.30

But they are actually two symbols and when its code is seen at the decoder we need to have a bit to differentiate them

		additional bit
s_3	0.21	0
s_4	0.09	1

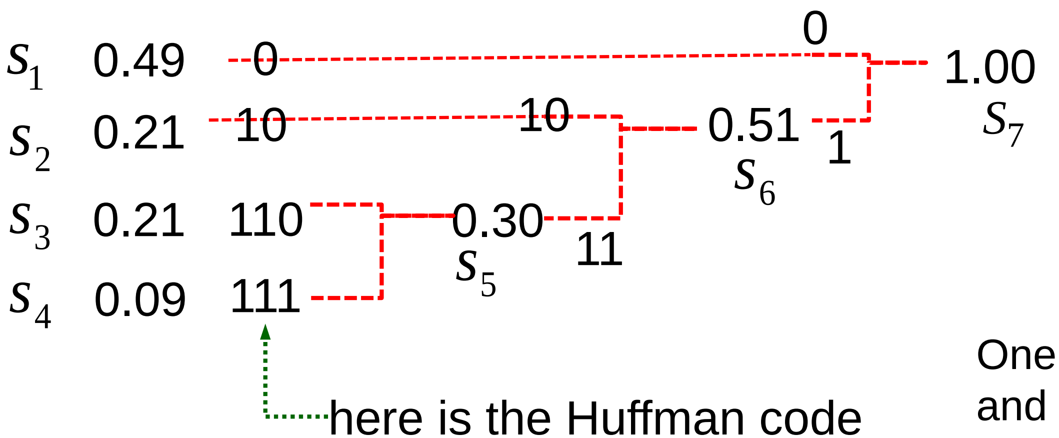
s_5 0.30

Now we have three symbols. Continue combining symbols with smallest probs and prepending differentiating bit marks.



This is called a Huffman tree

Now, from right to left, follow the paths for each symbol and find assigned bits to them by appending each bit to the right (LSB)



here is the Huffman code

One can create the tree first and assign bits later

We see that the ACL is same as the code found using Shannon-Fano. It is guaranteed that Huffman method generates shorter or same length codes

$$L_{avg} = \sum_{i=1}^{4} P(s_i)L(s_i) \qquad 0.49\times1+0.21\times2+0.21\times3+0.09\times3=1.81$$

Here is an Example where two methods generate different code lengths

		SF	Huf
s_1	0.36	00	0
s_2	0.18	01	100
s_3	0.17	10	101
s_4	0.16	110	110
s_5	0.13	111	111

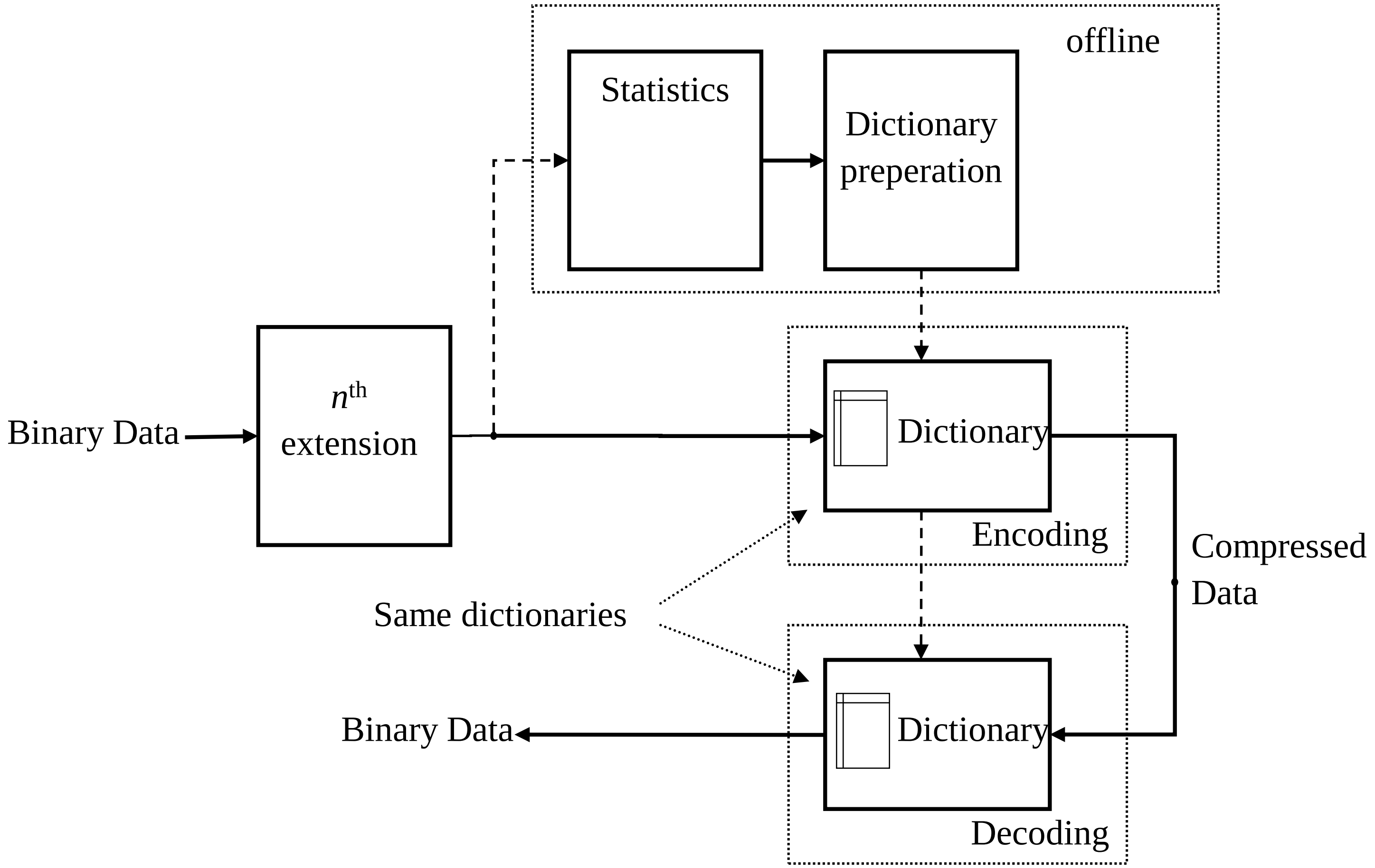
$$H(z) \leq L_{avgHuf} \leq L_{avgSF}$$

$H(z)$ =2.216 $\qquad L_{avgHuf}$=2.28

L_{avgSF}=2.29

Flow of Dictionary Coding

(Compression/Decompression)



Summary

What we have seen/learned

1. Information is the entity we need to preserve/communicate
2. We measure the information using probabilities of events/symbols
3. Average information of a source is the weighted (by their probabilities) self-information of events/symbols
4. Average Info Per source output is called Entropy of the source
5. Unique decodability is a requirement but instantaneous decodability is not.
6. Block codes are the simplest but not the only coding method.
7. Efficiency of block coding can be improved by extensions.
8. ACL $\geq$ Entropy

Related

$\sum\Delta$-Coding

END