

Principes  
des  
Systèmes  
Informatiques

INF2  
le bit comme unité d'information

# INF2

# Principes des Systèmes Informatiques

Olivier Ridoux

# Objectif

- Initiation à la **théorie de l'information**

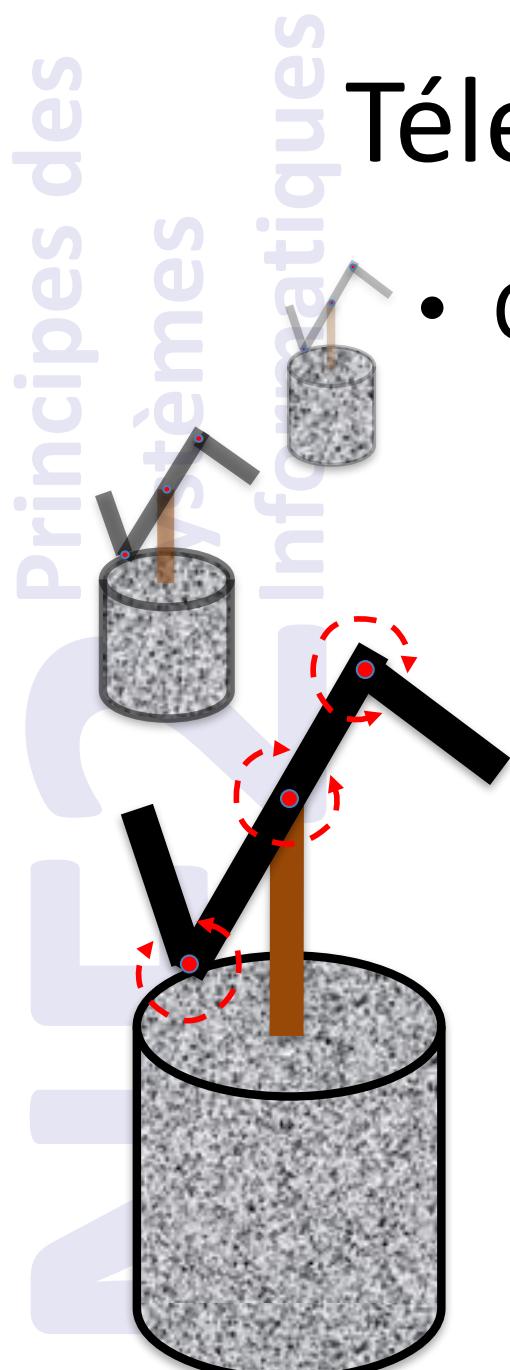


- Applications
  - **systèmes communicants / mémorisants**
  - **compression / décompression**
  - **détection / correction d'erreur**

# Plan

- Motivations
  - codages historiques
  - détection / correction d'erreur
  - compression de donnée
- Théorie de l'information

# Codages historiques



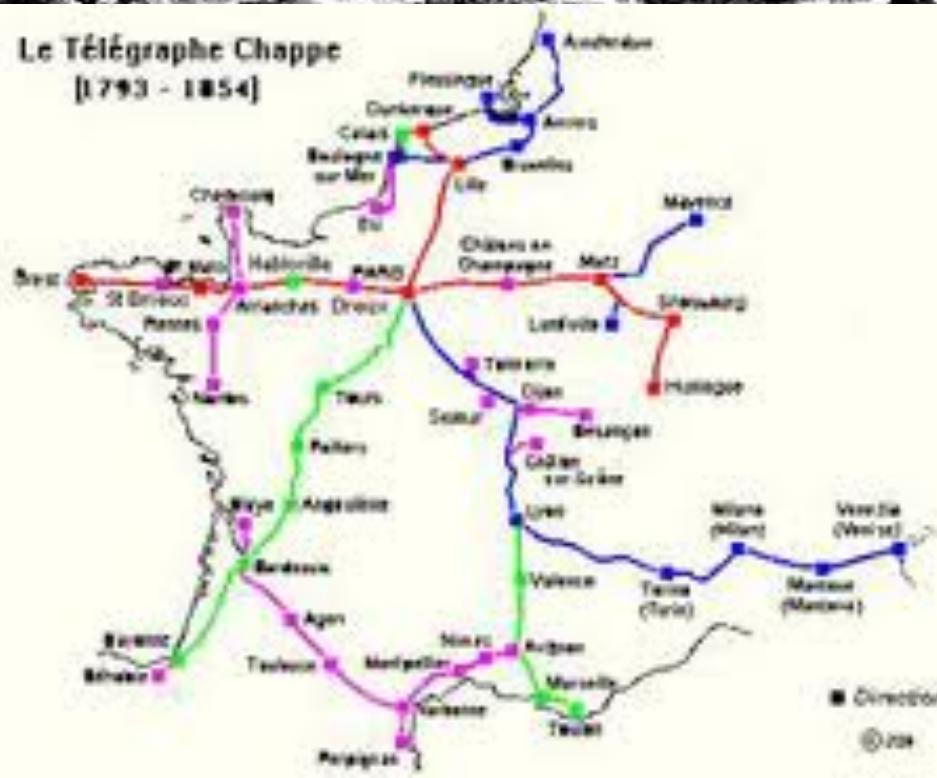
# Télégraphe de Chappe (1)

- Claude Chappe, France
  - 1794, 3 tours
  - 1844, 534 tours
    - connexions avec Amsterdam, Venise...
    - réseaux mobiles militaires en Crimée...
    - réseaux coloniaux en Tunisie...
  - **1845, télégraphe électrique !**
  - 1855, abandon
  - **2012, une tour à Saint-Marcan et une maquette à l'espace Ferrié**

# Télégraphe de Chappe (2)



## Le Télégraphe Chappe [1793 - 1854]



INF2 - bit d'information

	1	2	3	4	5	6	7	8	9	0
<i>Signaux primitifs du télégraphe Chappe.</i>										
1	↑		24			47			70	
2		1	25	1		48			71	
3	↓		26			49			72	
4		↑	27	1		50			73	
5	↑		28			51			74	
6		↑	29	1		52			75	
7	↑		30			53			76	
8		↑	31	1		54			77	
9	↑		32			55			78	
10		↑	33	1		56			79	
11	↑		34			57			80	
12		↑	35	1		58			81	
13	↑		36			59			82	
14		↑	37	1		60			83	
15	↑		38			61			84	
16		↑	39	1		62			85	
17	↑		40			63			86	
18		↑	41	1		64			87	
19	↑		42			65			88	
20		↑	43	1		66			89	
21	↑		44			67			90	
22		↑	45	1		68			91	

Le comité notera la situation financière de leurs clients à Lisbonne.



Archiv für

vient de  
x-numé-  
rit d'ail-  
rance à  
daient à  
tait très  
-  
nts dont  
illisèrent  
lique »).  
est sans

# Télégraphe de Chappe (3)

- Principe

- $2 \times 7 \times 7$  positions possibles

- $= 98$  positions

- (dont 6 de service)

- 1 message =  $n \times 2$  positions successives

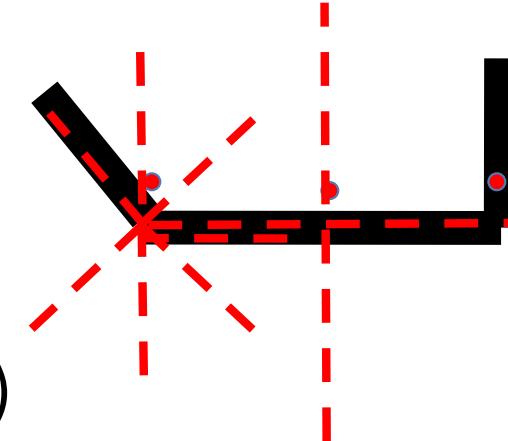
- un livre de code unique

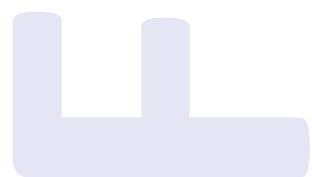
- paire de positions  $\leftrightarrow$  signification

- signification = mot ou **phrase**

- 98 positions = 98 symboles  $\Rightarrow$  7 bits

- **compression** d'une phrase en 7 bits !





# Télégraphe de Chappe (4)

- Invention du mot **télégraphe**
- Toute une administration
  - le **télégraphiste** (2 par tour)  
**répète le message**  
sans le décoder, mais en le notant
  - **l'inspecteur** (1 par 12 tours)  
**vérifie le dispositif**
  - le **directeur** (1 par extrémité)  
**a le livre de code**  
pour encoder/décoder le message

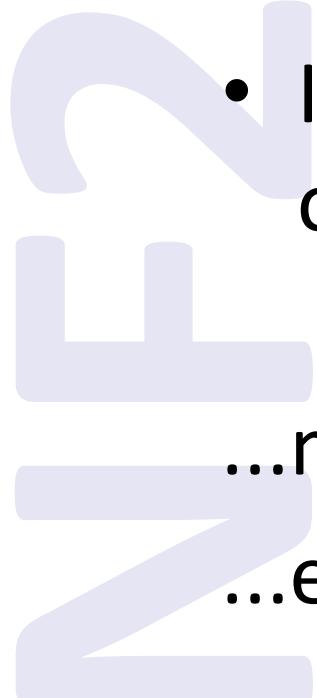


# Télégraphe de Chappe (5)

- Avantage
  - vitesse
    - seulement quelques heures pour traverser la France
    - **à comparer à celle d'un homme à cheval !**
- Inconvénients
  - nécessite un temps clair
  - le livre de code unique
    - distribution et mise à jour

# Tambour parlant d'Afrique (1)

- Des populations s'échangent des messages à l'aide de tambours
- Question des observateurs étrangers  
**quel est le code ?**



# Tambour parlant d'Afrique (2)

- Idée 1 : le tambour **code des lettres**...  
style code Morse ou ASCII  
...mais langues non écrites
- Idée 2 : le tambour **code des mots**  
ou des phrases  
style Chappe  
...mais où est le livre de code ?  
...et trop grande variété de ce qui est transmis

# Tambour parlant d'Afrique (3)

- Aparté linguistique
  - Les langues des populations concernées sont des **langues à tons**
  - **phonème** = plus petite différence sonore élémentaire qui change le sens d'un mot
    - voyelle ou consonne **ou ton**

table ≠ sable ≠ sabre ≠ sobre

laid = laie = lai = lait

**Il fait beau ≠ Il fait beau ?**

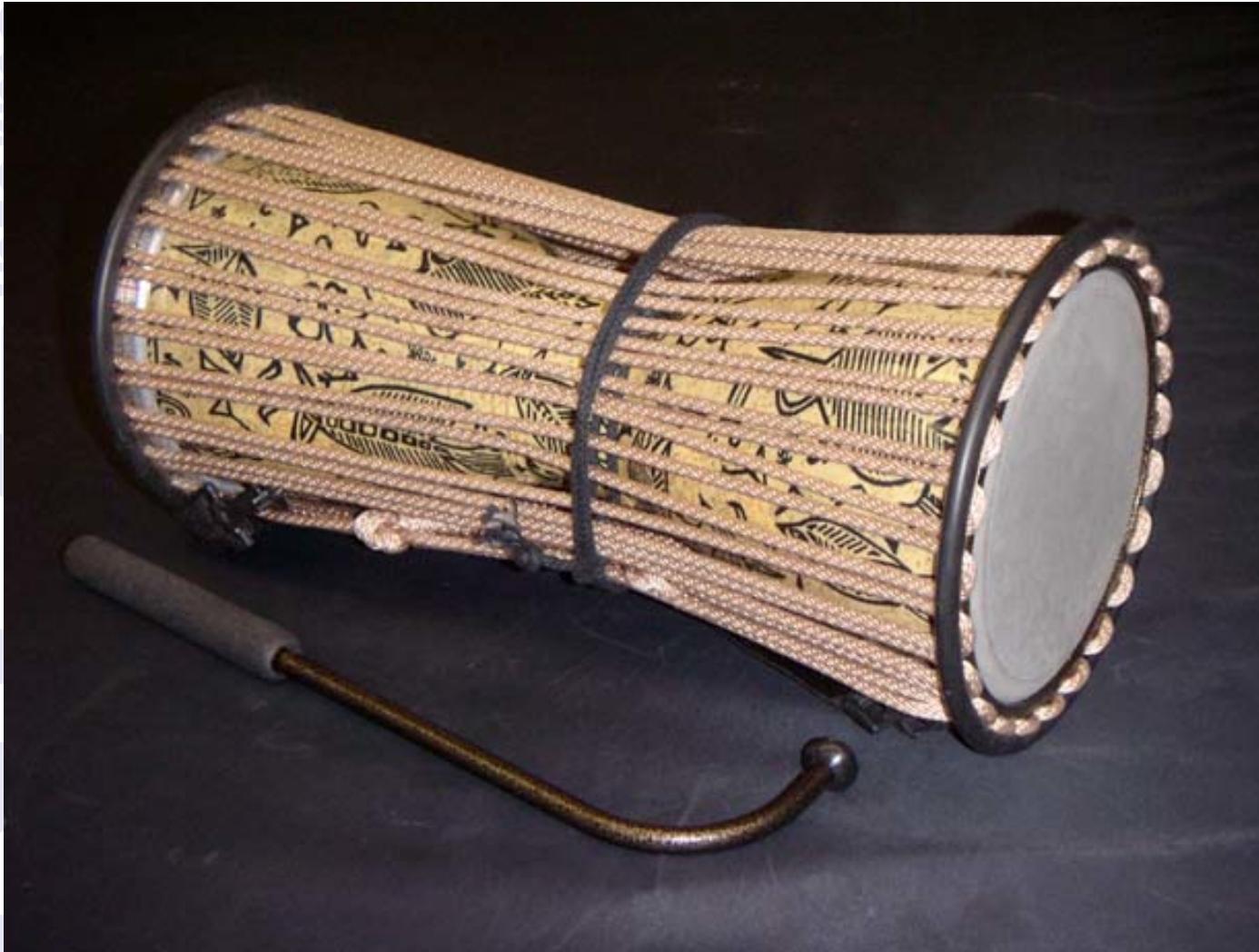
## → introduction **d'ambiguïtés**

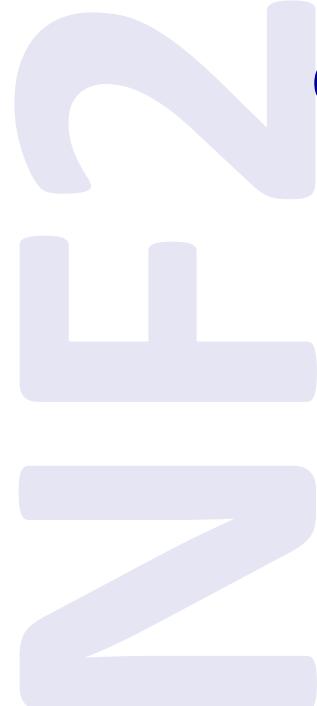
- imaginons ne prononcer que les consonnes  
*table ≠ sable ≠ sabre = sobre*

- Nouvelle question

**comment lever l'ambiguïté ?**

# Tambour parlant d'Afrique (5)





# Tambour parlant d'Afrique (6)

- Répéter ne sert à rien

**sbr, sbr, sbr, ...**

- Réponse

**compléter les mots devenus ambigus**

**pour les désambiguer**

**sbrtrch = sabretranchant**

**# sbrcmdrmdr =**

**sobrecommeundromadaire**

# Autres codages historiques (1)

- Télégraphie électrique
  - Morse
    - version 1 devait vraiment **écrire** des . et des –
    - version 2 se contente de **bip** et de **biiip**
  - Baudot
    - transmettre des caractères en télégraphie électrique
- Ce sont des codes **discrets**

## Autres codages historiques (2)

- Téléphonie
  - 1830, Sudre

- langue musicale universelle, **Solrésol**

*Il ne faut jamais commettre d'imprudences pour ne pas être malade*

***lasol falado fasol dosifado mido dofafadoré***

- 1876, Gray et Graham Bell

- téléphone analogique
    - **200Hz-20000Hz audible contre 300Hz-3400Hz transmis**

# Autres codages historiques (3)

- Télévision
  - 1920 environ
- Vidéo
  - 1950 environ
- Fax (+ scanner)
  - 1881

**Tous analogiques au début**

**Tous devenus numériques vers 2000**

# Conclusion

- Des messages **codés**
  - transmis dans un système technique qui ne voit que le code
  - du codage **discret** ou analogique
  - des éléments du code qui ne sont bien pas transmis

le **bruit**

- de la **redondance** pour y remédier

formalisé par

**la théorie de l'information**

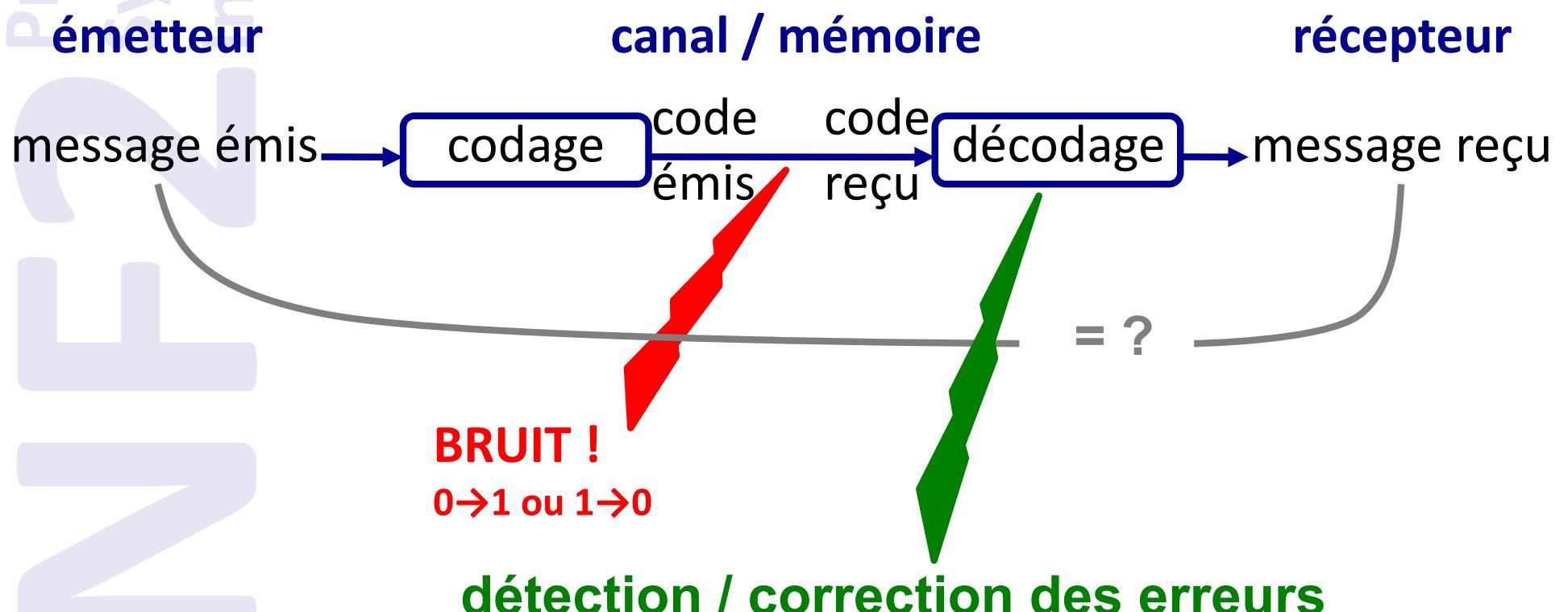
# Détection / correction d'erreurs

# Exemple motivant

## Altération d'un bit de stockage

- Bascule spontanée d'un bit de stockage ou de communication
  - 0 devient 1 ou 1 devient 0
- Analogie **mémorisation / communication**
  - message dans le **temps**
  - message dans **l'espace**

# Canal de communication / mémorisation



# Détection des erreurs

- Décider si **message émis = message reçu** en ne connaissant que la sortie du canal (le **code reçu**)
  - code **détecteur d'erreur**

# Hypothèses (1)

- Erreurs **rares**
  - probabilité d'altération spontanée faible
- Erreurs **indépendantes**
  - pas de rafales d'erreurs
- Pour fixer les idées
  - probabilité d'erreur  $p = 1/10000$
  - longueur du message = 10 bits

## Hypothèses (2)

- Donc
  - probabilité de 0 erreur = 0,998
  - erreur **simple**
    - probabilité de exactement une erreur dans le message =  $1,7 \times 10^{-3}$
  - erreur **double**
    - probabilité de deux erreurs et plus =  $1,4 \times 10^{-6}$
- S'occuper des erreurs **simples**

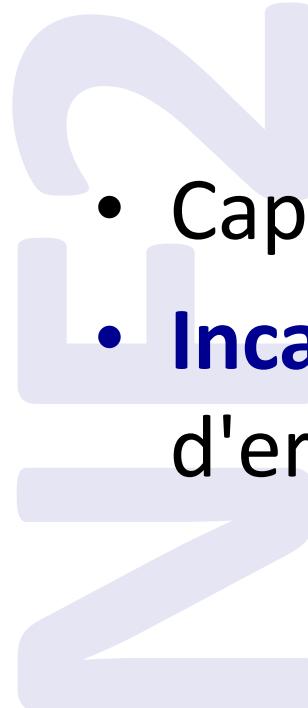
# Duplication des bits du message

## Idée du codage

- Coder 1 en 11 et 0 en 00

## Exemple

- Message émis = 1101011001
- **Code émis = 11110011001111000011**



# Idée du décodage

- Lire les bits deux par deux

$11 \rightarrow 1$

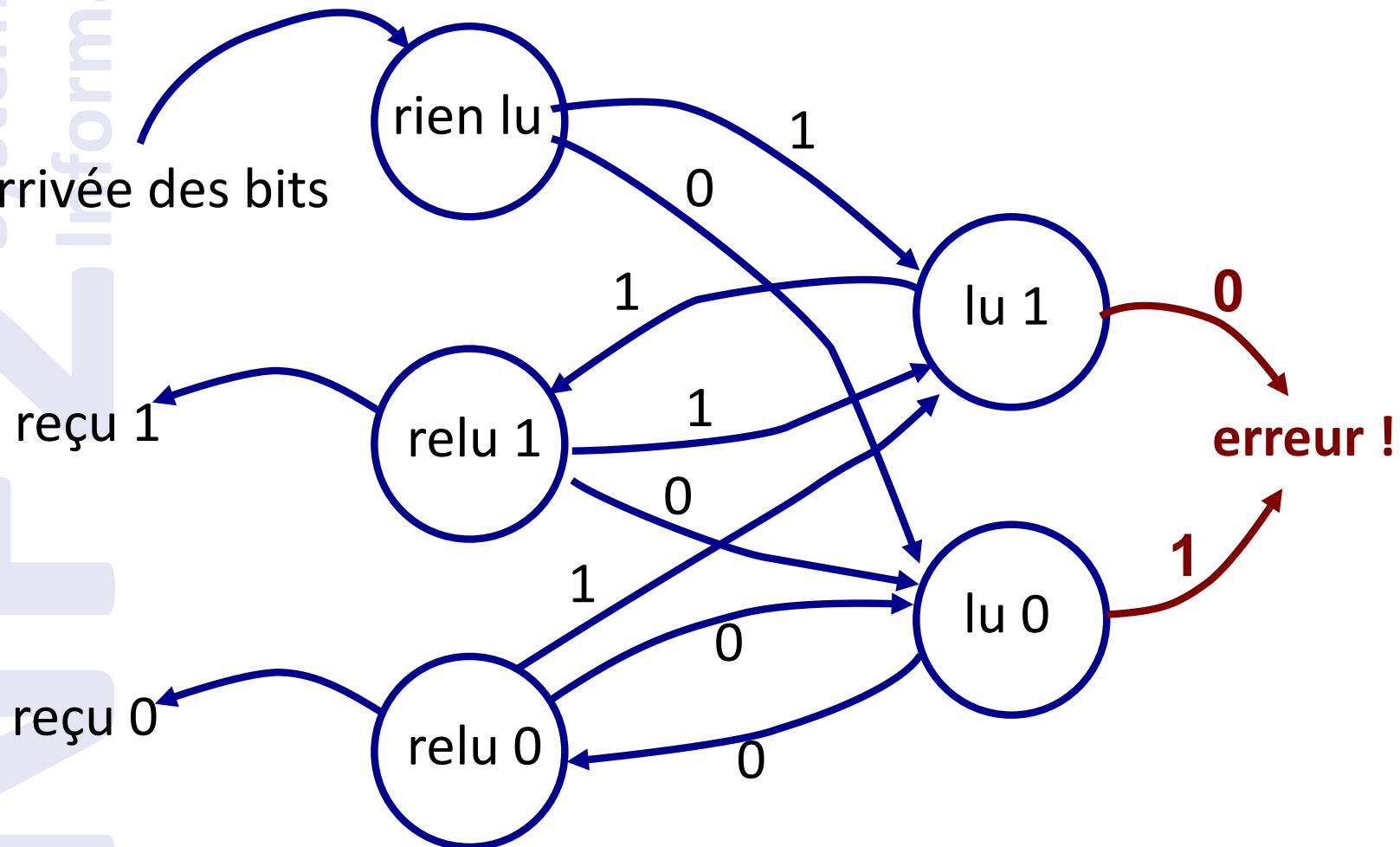
$00 \rightarrow 0$

si 01 ou 10 alors **erreur**

- Capable de **déetecter** une erreur par bit émis
- **Incapable** de détecter les nombres **pairs**  
d'erreurs sur le même bit (proba  $\leq 1/1000000$ )

$11 \rightarrow 10 \rightarrow 00 \rightarrow 01 \rightarrow 00$

# Automate de décodage



# Bilan

- Détection des erreurs simples  
    1 par bit émis
- Détection des erreurs impaires
- Localisation de l'erreur

Mais

- correction impossible  
 $00 \rightarrow \textcolor{blue}{10} \leftarrow 11 ?$
- 100 % de surcoût

# Tripllication des bits du message

## Idée du codage

- Coder 1 en 111 et 0 en 000

# Exemple

- Message émis = 1101011001
- **Code**  
**émis = 111110001110001111110000**  
**00111**

si 011, 101 ou 110 alors **erreur corrigible** → 1

si 100, 010 ou 001 alors **erreur corrigible** → 0

- Capable de **corriger** toutes les erreurs **simples**
- **Se trompe** systématiquement pour toutes les erreurs **doubles**

$1 \rightarrow 111 \rightarrow 101 \rightarrow 100 \rightarrow 0$

# Bilan

- Détection des erreurs simples
  - 1 par bit émis
- Correction des erreurs simples
- Localisation de l'erreur

Mais

- erreur systématique quand erreur double
- 200 % de surcoût



# Contrôle de parité

# Idée du codage

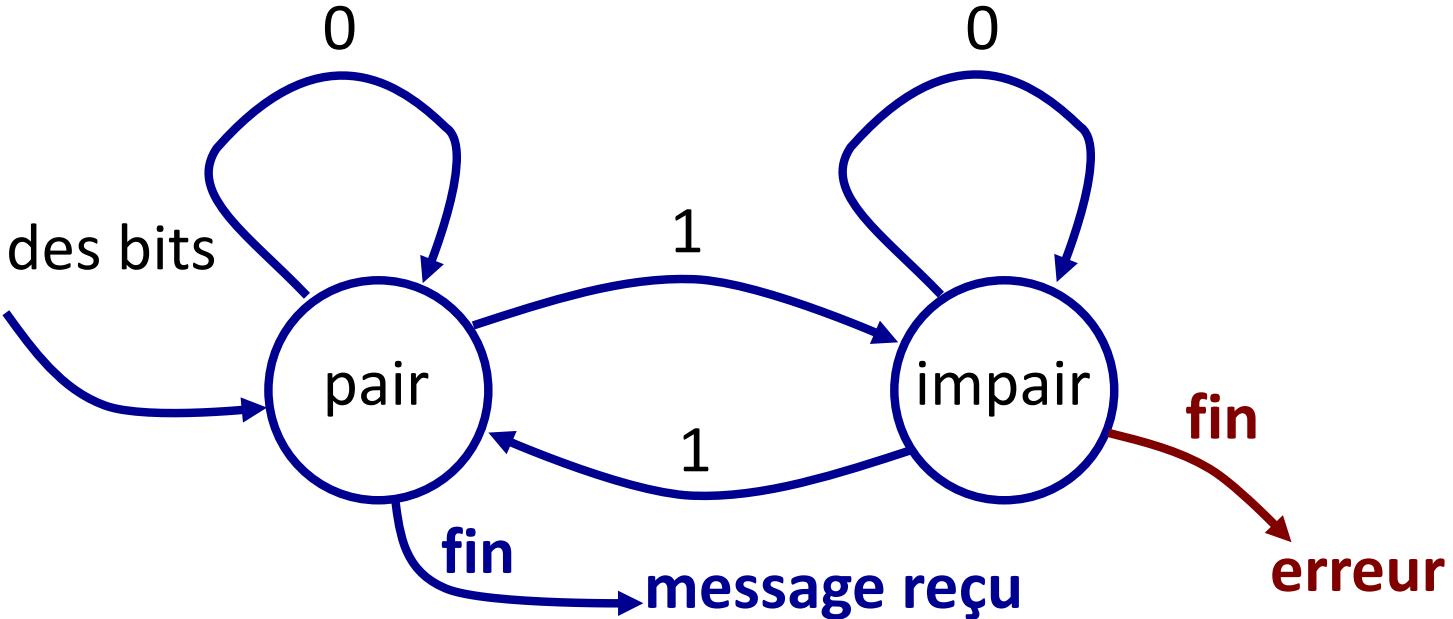
- Ajouter un 11<sup>ème</sup> bit
  - **message émis** sur 10 bits
  - **code émis** sur 11 bits
- **1 si nombre de bits à 1 impair, 0 sinon**
  - parité = somme des chiffres modulo 2
  - code émis = message émis • parité
    - opération de **concaténation**
- **Au total le nombre de bits à 1 du code émis est pair**

- Message émis = 1101011001
  - 6 bits à 1, donc **parité = 0**
- **Code émis = 1101011001• 0**  
**= 11010110010**

# Idée du décodage

- Compter les bits à 1 dans le code reçu
  - si **pair** alors **message reçu = (code reçu)<sub>1 à 10</sub>**
  - si **impair** alors **erreur**
- Capable de **déetecter** tout nombre **impair** d'erreurs
- **Incapable** de détecter les nombres **pairs** d'erreurs  
(probabilité  $\leq 1/1000000$ )

# Automate de décodage



- Pas besoin de savoir compter !
  - comme pour détecter les pages récemment utilisées

# Bilan

- Détection des erreurs simples
- Détection des erreurs impaires

Mais

- pas de localisation de l'erreur
- 10 % de surcoût
- l'erreur peut être n'importe où dans le code reçu,  
**même dans le bit de parité**



# Contrôle de parité multiple

# Idée du codage

- Représenter le message dans un tableau  $n \times m$

Ex : 1101011001 →

1	1	0	1	0
1	1	0	0	1

- **Ajouter un bit de parité à chaque ligne et à chaque colonne**

## Exemple

- Message émis = 1101011001
- Vision tableau

1	1	0	1	0	<b>1</b>
1	1	0	0	1	<b>1</b>
<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>

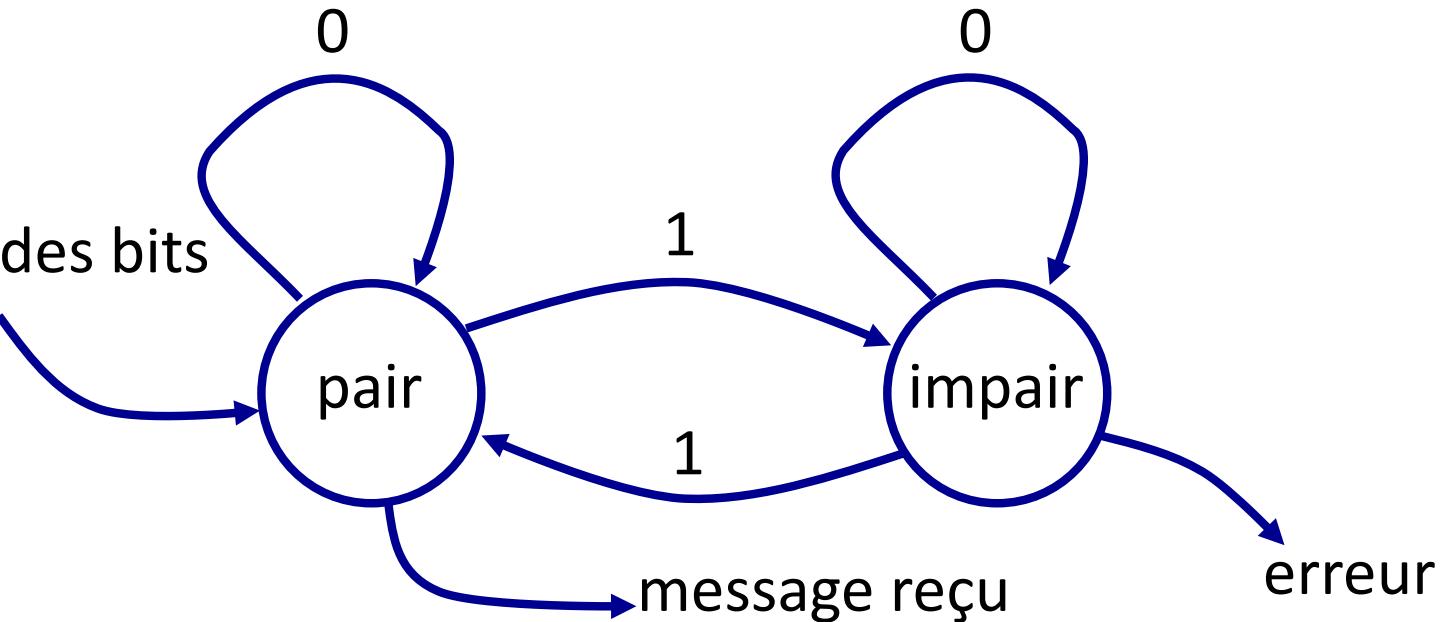
- **Code émis = 11010• 1• 11001• 1• 000110**

## Idée du décodage

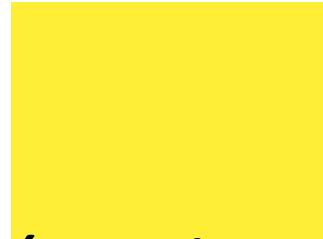
- Compter les 1 dans les lignes du code reçu
  - si **pair** alors **message reçu = (code reçu)<sub>1 à 10</sub>**
  - si **impair** alors **erreur** dans ligne **I**
  - compter les bits à 1 dans les colonnes
  - si **impair** dans colonne **c**  
alors erreur dans colonne **(I,c)**
- Capable de **déetecter** et **corriger** toutes les erreurs simples

XOR	0	1
0	0	1
1	1	0

# Automate de décodage



- Parité<sub>attendue</sub> =  
Parité<sub>attendue</sub> **XOR** Ligne<sub>reçue</sub>



## Bilan (1)

- Détection et correction des erreurs simples
- Détection des erreurs doubles

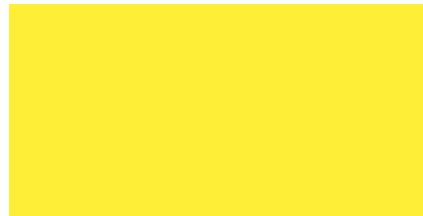
Mais

- $[(n+1) \times (m+1) - n \times m]/n \times m \%$  de surcoût  
 $= (n+m+1)/n \times m \%$
- $5 \times 2 \rightarrow 80 \%$  de surcoût
- optimal pour messages carrés,  $n = m$

## Bilan (2)

<b>n</b>	<b><math>n \times n</math></b>	<b>surcoût</b>	<b>proba 0 erreur</b>	<b>proba 1 erreur</b>	<b>proba <math>\geq 2</math> erreur</b>
10	100	21 %	0,988	0,012	0,00007
100	10000	2 %	0,36	0,37	0,27
1000	1000000	0,2 %	0,000...	0,000...	0,999...

# Codes de Hamming



# Objectif du codage

- Corriger les erreurs simples en les localisant
- Si  $m$  bits de messages et  $n$  bits de surcoût, localiser parmi  $m+n$  bits
- $n$  bits permettent de localiser  $m+n$  positions
- Donc  $2^n > (n+m) \rightarrow m < 2^n - n$

# Redondance

$n =$ <b>surcoût</b>	3	4	5	6	7	8	9	10	20	30
$2^n$	8	16	32	64	128	256	512	1024	$\sim 10^6$	$\sim 10^9$
$2^n - n - 1 =$ <b>longueur message utile</b>	4	11	26	57	120	247	502	1013	$\sim 10^6$	$\sim 10^9$
<b>% surcoût</b>	75	36	19	11	6	3	1,8	1	$\varepsilon$	$\varepsilon$

$\% \text{ surcoût} = \text{surcoût} / \text{longueur message utile}$

# Idée du codage

- Considérer les bits de contrôle comme des bits d'adresses dans le code
- Noter dans chaque bit de contrôle  $c_i$  la parité des positions dont le  $i$ -ème bit d'adresse contient un 1

pos	$c_3$	$c_2$	$c_1$	$c_0$
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

# Exemple (1)

- 11 bits de données  
+ 4 bits de contrôle
- contrôle =  $c_3c_2c_1c_0$ 
  - $c_0$  code les positions  
1, 3, 5, 7, 9, 11, 13, 15
  - $c_1$  code les positions  
2, 3, 6, 7, 10, 11, 14, 15
  - $c_2$  code les positions 4 à 7, 12 à 15
  - $c_3$  code les positions 8 à 15

pos	$c_3$	$c_2$	$c_1$	$c_0$	val
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	1
4	0	1	0	0	
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

INF2 - bit d'information

## Exemple (2)

- 11 bits de données + 4 bits de contrôle
- contrôle =  $c_3c_2c_1c_0$ 
  - $c_i$  sera en position  $2^i$
- 10110001101 sera codé pp1p011p0001101 soit **001101110001101**
  - détecte et corrige toute erreur simple

pos	$c_3$	$c_2$	$c_1$	$c_0$	val
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	1
4	0	1	0	0	
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

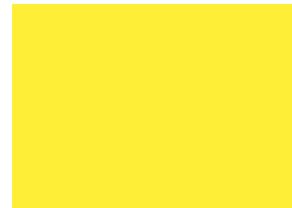
## Exemple (3)

- Code de Hamming étendu
  - pour avoir  $2^k$  bits
- 10110001101 sera codé  
**0011011100011010**
  - détecte et corrige toute erreur simple, et détecte toute erreur double

# Conclusion sur correction d'erreur

- Des bits redondants rendent le codage plus robuste
- Les meilleures solution codent plusieurs bits à la fois
- **Peut-on améliorer arbitrairement la détection/correction d'erreur ?**

# Compression / décompression



# Objectif

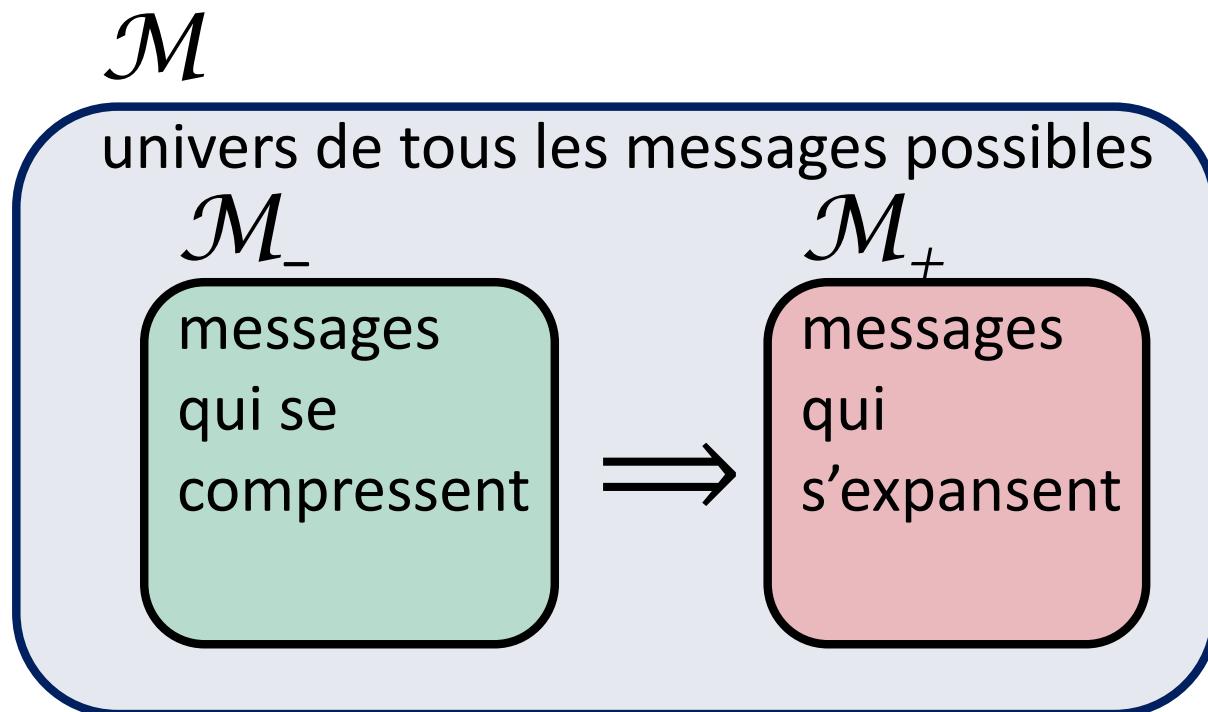
- Remplacer une représentation par une plus petite de **même sémantique**
- Compression **sans perte**
  - **nécessaire** pour textes, programmes, données numériques, etc
- Compression **avec perte** sinon
  - tolérable (et recommandé) pour image et son

# *Pigeon hole principles*

- Si il a plus de pigeons que de trous,  
il y aura au moins un trou  
avec plusieurs pigeons

## No free lunch theorem (1)

Une méthode de **compression sans perte** qui compresse **strictement** certaines sources ne peut que en **expanser** d'autres



## No free lunch theorem (2)

- Application du Pigeon Hole Principle

$n$  bits codent  $2^n$  messages,

donc  $n' < n$  bits codent  $2^{n'} < 2^n$  messages,

donc **perte de messages**

si compression stricte  
d'une taille  $n$  à une taille  $n' < n$

# No free lunch theorem (3)

- Soit  $M \in \{\text{messages de taille } t\}$   
alors  $|\{m \text{ de taille } < t\} \cup \{M\}| = |\{m \text{ de taille } < t\}| + 1$
- Si  $K$  compresse strictement  $M$   
alors  $K(M) \in \{m \text{ de taille } < t\}$

et  $K(\{m \text{ de taille } < t\} \cup \{M\}) \subseteq \{m \text{ de taille } < t\}$

d'où  $|K(\{m \text{ de taille } < t\} \cup \{M\})| \leq |\{m \text{ de taille } < t\}|$

# No free lunch theorem (5)

- Si  $K$  sans expansion

alors  $|K(\{m \text{ de taille} < t\} \cup \{M\})|$

$$\leq |\{m \text{ de taille} < t\}|$$

*absurde*

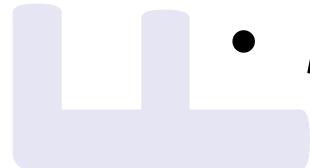
- Si  $K$  sans perte

alors  $|K(\{m \text{ de taille} < t\} \cup \{M\})|$

$$= |\{m \text{ de taille} < t\} \cup \{M\}|$$

$$= |\{m \text{ de taille} < t\}| + 1$$

$$> |\{m \text{ de taille} < t\}|$$



# Codage par plage

- Remplacer une suite de symboles identiques (une **plage**) par le symbole et la **longueur** de la plage

**AAAABBBACCCCCC**AAAABB

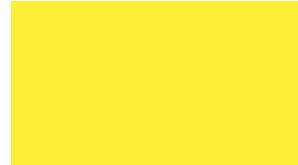
→ **A4B3A1C7A4B2**

- *No free lunch!*

**ABC** → **A1B1C1**

- Ex. codage des fax : beaucoup d'à-plats de noir et de blanc

# Codage de Huffman (1)



- Utiliser **moins de bits** pour représenter les symboles **fréquents** que pour les symboles rares

# Codage de Huffman (2)

message source	probabilité du message	codage	longueur du code	longueur pondérée
AA	9/16	0	1	9/16
AB	3/16	10	2	6/16
BA	3/16	110	3	9/16
BB	1/16	111	3	3/16
longueur moyenne du code				27/16 = 1,69 au lieu de 2 sans compression

Code préfixe!

AAAAAABAAAABAAABBAAAA → 001000110011100

au lieu de 00000100001000110000

- *No free lunch!*

BB → 111 au lieu de 11 (50 % expansion)

# Conclusion sur compression

- *No free lunch theorem*
- Meilleures solutions codent plusieurs bits à la fois
- **Peut-on améliorer arbitrairement la compression ?**

# Théorie de l'information

Claude Shannon, MIT et Bell Labs (AT&T)

*A Mathematical Theory of Communication,*  
1948

- Notion de quantité d'information

# Définitions préliminaires (1)

- **Vocabulaire**  $V = \{ s_1, s_2, \dots, s_n \}$   
Ex. lettres de l'alphabet
- Message sur un vocabulaire  $V$   
séquence finie de symboles de  $V$
- Ensemble de tous les messages possibles =  $V^*$

## Définitions préliminaires (1)

$V^*$

univers de tous les messages possibles

# Définitions préliminaires (2)

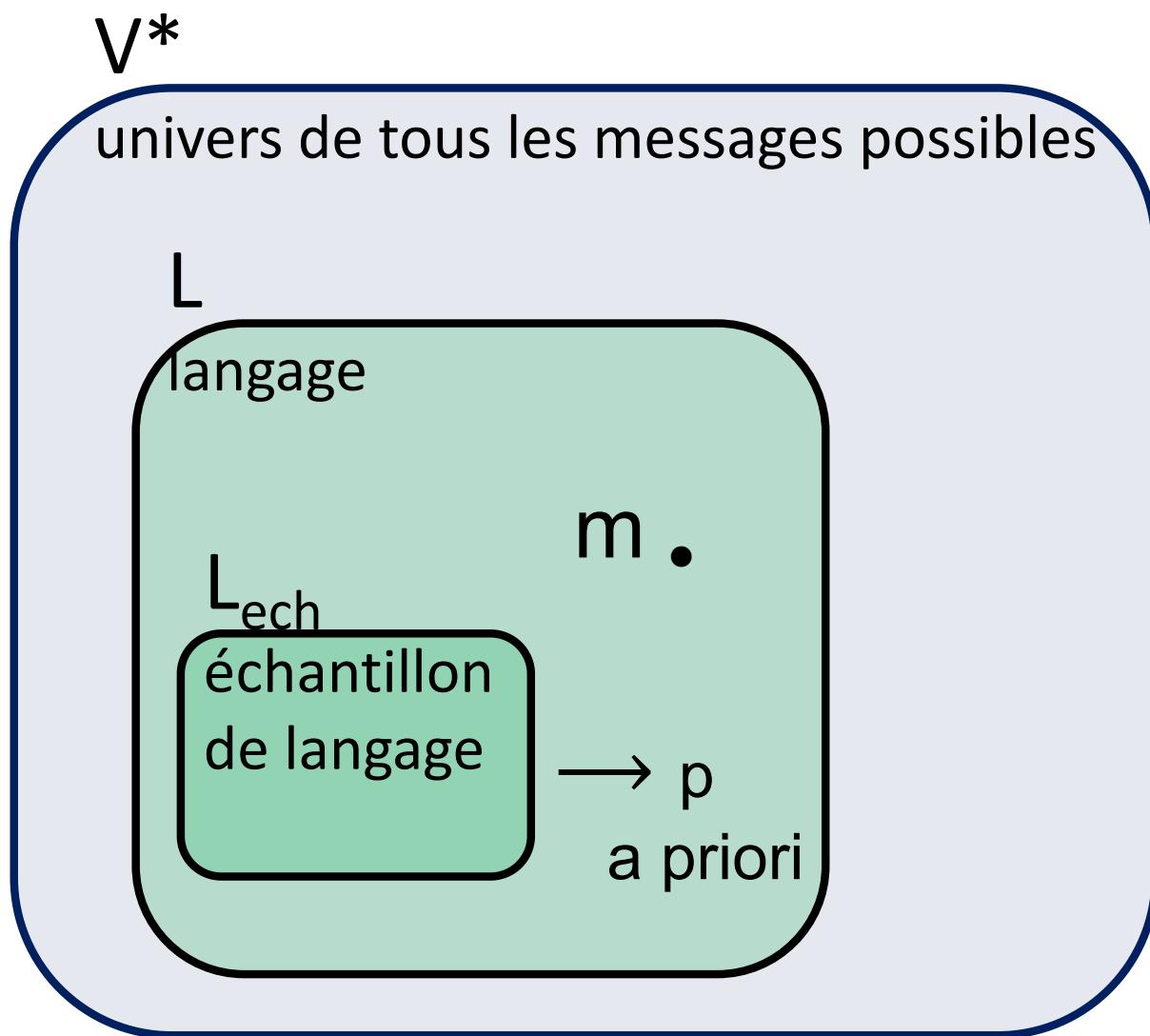
- **Langage**  $L \subset V^*$

**Probabilité a priori** d'occurrence dans  $L$   
de chaque symbole

$$p(s), \sum_{s \in V} p(s) = 1$$

Ex. en français,  $p(e) \gg p(z)$

## Définitions préliminaires (2)



# Définitions préliminaires (3)

- Cas du français écrit
  - **Vocabulaire**  $V = \{ a, b, c, d, \dots, x, y, z \}$
  - **Probabilité** d'occurrence de chaque symbole (estimée sur un échantillon)
- Un message écrit en français
  - « Zazie zézaie à Zanzibar »

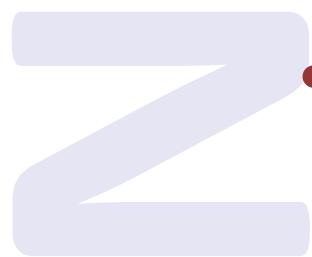
$$f(z) > f(a) > f(e) \quad \text{même si} \quad p(e) > p(a) > p(z)$$

e	0,1210
a	0,0711
...	...
t	0,0592
...	...
w	0,0017
z	0,0015

# Le bit d'information (1)



- Soit un récepteur qui connaît la probabilité a priori des différents symboles
  - il découvre un nouveau message, symbole après symbole
  - chaque symbole fait diminuer l'incertitude sur ce que contient le message
    - **un symbole rare fait beaucoup diminuer l'incertitude**
    - **un symbole fréquent la fait peu diminuer**



## Le bit d'information (2)

- Définition (Shannon) : quantité d'information portée par un symbole

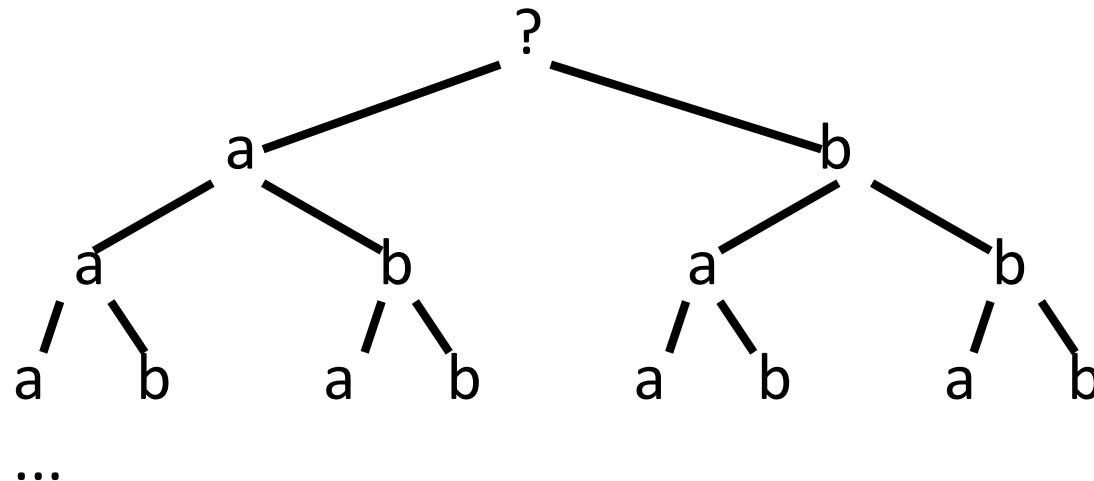
$$h(s) = \log_2 1/p(s) = -\log_2 p(s)$$

–  $h$  parfois noté  $I$  (grand i)

- **si probabilité croît,  
information décroît**

# Exemple 1 – quantité d'information

- $V = \{ a, b \}$ ,  $p(a) = p(b) = \frac{1}{2}$   
 $\mathbf{h(a) = h(b) = \log_2 1/(\frac{1}{2}) = \log_2 2 = 1 \text{ bit}}$
- Chaque symbole, **a** ou **b**, divise exactement par deux l'ensemble des messages probables



## Exemple 2 – quantité d'information

- $V = \{ a, b \}$ ,  $p(a) = \frac{3}{4}$ ,  $p(b) = \frac{1}{4}$   
$$h(a) = \log_2 1/(3/4) = \log_2 4/3 = \log_2 4 - \log_2 3$$
$$= 2 \log_2 2 - \log_2 3 = 0,415 \text{ bit}$$
  
$$h(b) = \log_2 1/(1/4) = \log_2 4 = 2 \log_2 2 = 2 \text{ bits}$$
- **b** est **3** fois plus rare que **a**,  
divise l'ensemble des messages probables  
par **4**
- **a** étant **fréquent** apporte **peu** d'information

## Le bit d'information (3)

- Définition (Shannon) : quantité d'information portée par un message

$$h(m) = \sum_{s \in m} h(s)$$

- si  $m$  contient  $m'$ ,  $h(m) > h(m')$

# Entropie

- Définition (Shannon) : quantité moyenne d'information portée par un vocabulaire  $V$

$$H(V) = \sum_{s \in V} (p(s) \times h(s))$$

$$= \sum_{s \in V} (p(s) \times \log_2 1/p(s))$$

$$= - \sum_{s \in V} (p(s) \times \log_2 p(s))$$

## Exemple 3 – entropie

- $V = \{ a, b \}$ ,  $p(a) = p(b) = \frac{1}{2}$

$$h(a) = h(b) = \log_2 1/\left(\frac{1}{2}\right) = \log_2 2 = 1 \text{ bit}$$

$$H(V) = p(a) \times h(a) + p(b) \times h(b) = \frac{1}{2} + \frac{1}{2} = 1$$

## Exemple 4 – entropie

- $V = \{ a, b \}$ ,  $p(a) = \frac{3}{4}$ ,  $p(b) = \frac{1}{4}$

$$\begin{aligned} h(a) &= \log_2 1/\left(\frac{3}{4}\right) = \log_2 4/3 = \log_2 4 - \log_2 3 \\ &= 2 \log_2 2 - \log_2 3 = \mathbf{0,415 \text{ bit}} \end{aligned}$$

$$h(b) = \log_2 1/\left(\frac{1}{4}\right) = \log_2 4 = \log_2 2 = \mathbf{2 \text{ bits}}$$

$$\begin{aligned} H(V) &= p(a) \times h(a) + p(b) \times h(b) \\ &= \frac{3}{4} \times \log_2 4/3 + \frac{1}{4} \times \log_2 4 = \mathbf{0,812 \text{ bits}} \end{aligned}$$

- Si même vocabulaire,  
l'entropie est maximale  
pour distribution uniforme

## Exemple 4 – entropie

- $V = \{ a, b \}$ ,  $p(a) = \frac{3}{4}$ ,  $p(b) = \frac{1}{4}$

$$h(a) = 0,415 \text{ bit}$$

$$h(b) = 2 \text{ bits}$$

$$H(V) = 0,812 \text{ bits}$$

- Codage naïf

**100** bits pour message de longueur **100**

- Compression

peut-on approcher de **82** bits ?

## Exemple 5 – entropie

- $V = \{ a, b, c, d \}$ ,  $p(a) = p(b) = p(c) = p(d) = \frac{1}{4}$

$$h(a) = h(b) = h(c) = h(d) = \log_2 1/\left(\frac{1}{4}\right) = \log_2 4 = 2 \text{ bits}$$

$$H(V) = 4 \times [ \frac{1}{4} \times 2 ] = 2 \text{ bits}$$

- Si distribution uniforme,  
l'entropie croit avec la taille du vocabulaire



prime aux codages multi-symboles

# Entropie maximale

Théorème : Pour un nombre de symboles donné,  
les vocabulaires où les **probabilités** d'occurrence  
des symboles sont **égales**  
ont **l'entropie la plus grande**  
(= quantité d'information moyenne la plus grande)

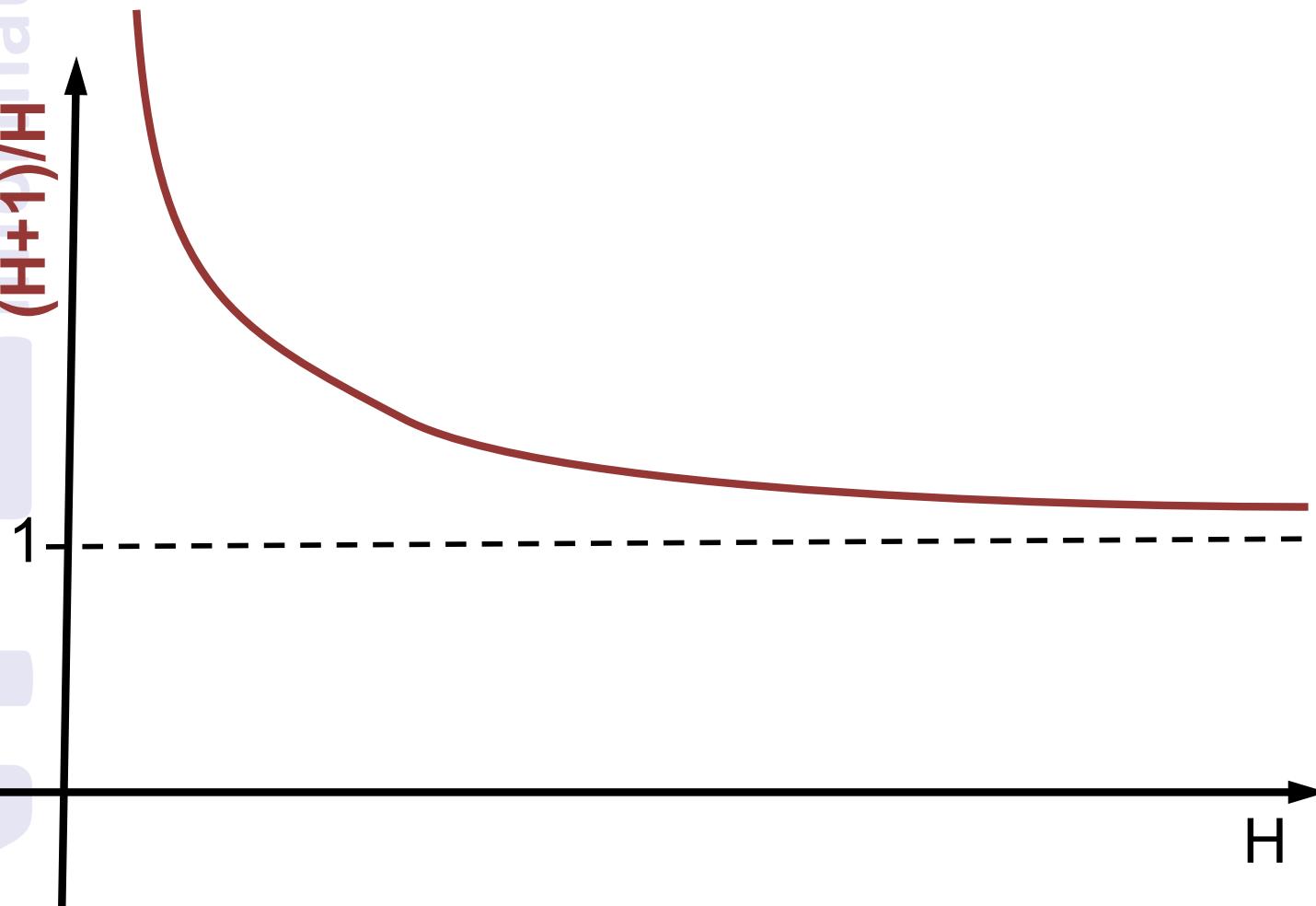
# Théorème - Compression

- Soit une source  $L$ , il existe un codage dont la longueur moyenne est **longueur** avec

$$H(L) \leq \text{longueur} \leq H(L) + 1$$

- Compression limitée par **entropie de la source**
- Avantage aux codages multi-symbole

# Théorème - Compression



# Théorème - Correction d'erreur

Version simplifiée

- Soit un **m** message de **M** bits,  
transmis sur un canal **bruité**

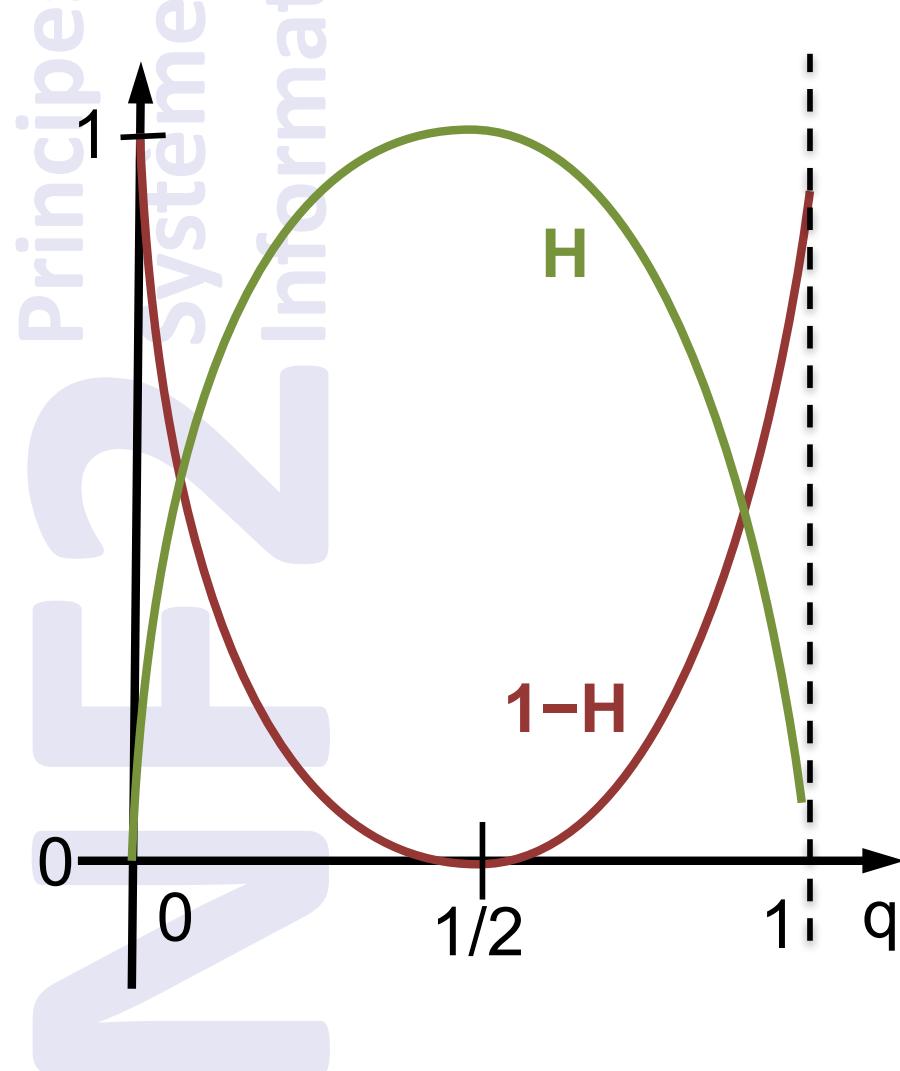
**q** la probabilité d'erreur par bit

**$M_c$**  la taille du message encodé ( $M_c \geq M$ )

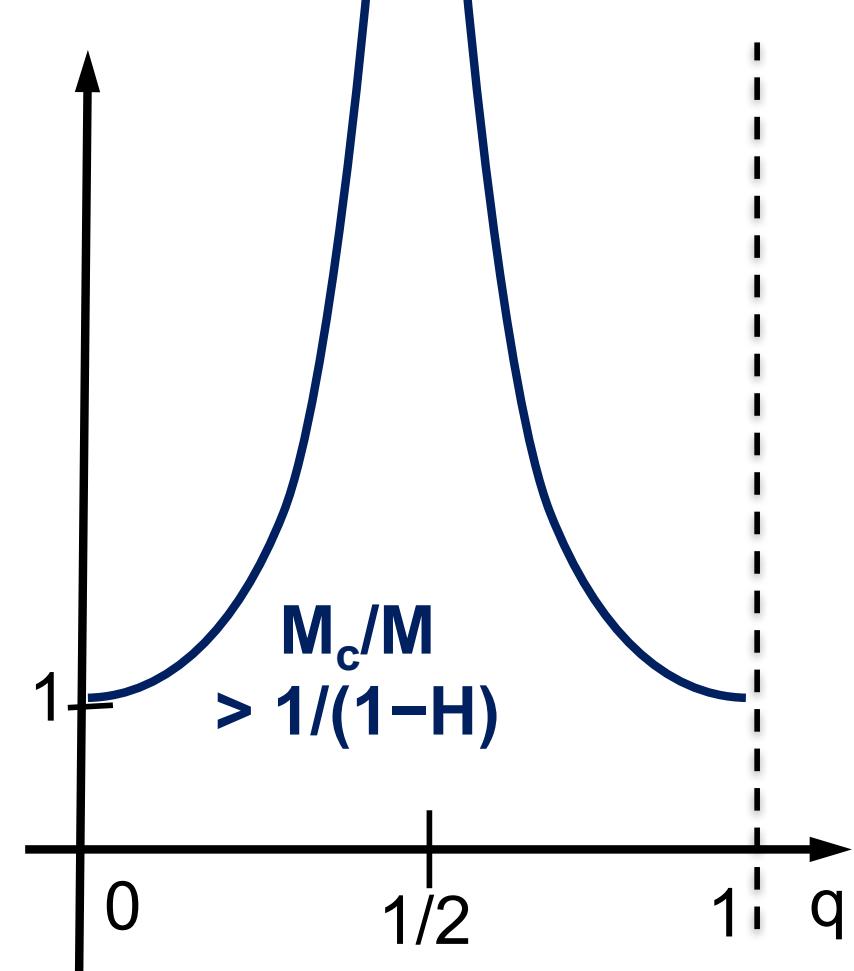
Il est possible de transmettre **m** sans erreur si

$$\frac{M}{M_c} < 1 - [-q \times \log_2 q - (1-q) \times \log_2 (1-q)]$$

# Théorème - Correction d'erreur



INF2 - bit d'information



91

# Conclusion (1)

- Besoin universel de **détection / correction** d'erreur dans les systèmes informatique
  - **fragilité** du stockage
  - fragilité des communications
- Besoin universel de **compression**
  - **optimisation** du stockage
  - optimisation des communications

## Conclusion (2)

- Distinguer bit de stockage / communication de bit d'information
  - information =  **$-\log_2 \text{probabilité d'occurrence}$**
  - entropie = **information moyenne pondérée**
- Théorèmes limites
  - compression limitée par **entropie de la source**
  - débit limitée par **entropie du canal**

## Conclusion (3)

- Le sujet de votre séquence de TP
  - un vocabulaire
    - les lettres de l'alphabet [+ ponctuation]
  - une distribution de probabilité  $p$ 
    - mesurée sur des exemples
  - calculer information de chaque symbole et entropie du vocabulaire
  - engendrer des textes aléatoires selon cette distribution de probabilité