Introduction aux Systèmes de Communication

Luciano Sbaiz, Patrick Thiran, Rüdiger Urbanke

8 octobre 2007

Table des matières

Chapitre 2

Codage de Source

2.1 Eléments de théorie de l'information

2.1.1 Mesure de l'information

On considère une source discrète d'information S [1], délivrant des messages ou symboles s_1, s_2, ..., s_M avec les probabilités $P(S = s_1) = p(s_1) = p_1$, $P(S = s_2) = p(s_2) = p_2$, ..., $P(S = s_M) = p(s_M) = p_M$. L'ensemble des M symboles est appelé alphabet $[S]$. Lorsqu'on reçoit un de ces symboles, quelle quantité d'information apporte-t-il ? Si $p_1 = 1$ (et donc $p_i = 0$ pour tout $i \neq 1$), il n'y a aucune surprise à recevoir le symbole s_1, celui-ci n'apporte aucune information. Par contre, si $p_1 = 0.0001$, la "surprise" de recevoir s_1 parmi les M symboles que la source peut délivrer est beaucoup plus grande, ainsi que la quantité d'information apportée. L'information représente une mesure de l'incertitude, sa valeur réside dans l'effet de surprise qu'elle génère et croît donc avec l'inverse de la probabilité.

On cherche à construire une fonction $I(S)$ de cette source d'information S qui mesure l'information apportée par chaque symbole, ce qu'on peut encore noter $I(p)$ où p désigne la probabilité d'apparition d'un des symboles.

Pour construire cette fonction $I(S)$, on doit donc faire les hypothèses suivantes :

H1. $I(S) = I(p) \geq 0$: l'information est une mesure non négative,

H2. $I(p = 1) = 0$: un évènement certain n'apporte aucune information,

H3. $I(p)$ est une fonction décroissante et continue de p,

H4. $I(p_1 p_2) = I(p_1) + I(p_2)$ pour deux évènements indépendants (condition d'additivité).

La seule fonction qui satisfait à ces conditions est

$$I(S) = I(p) = -C \log_b p$$

où la constante positive C et la base b peuvent être choisies arbitrairement. On prendra $C = 1$ et $b = 2$, d'où

$$I(p) = -\log_2 p. \tag{2.1}$$

L'unité rigoureuse de mesure d'information est le *shannon*, l'unité fréquemment utilisée est le *bit*.

2.1.2 Entropie

Pour connaître la *quantité d'information moyenne par symbole* émis par la source S, qu'on désigne par $H(S)$, on calcule :

$$H(S) = \sum_{i=1}^{M} p_i \, I(p_i) = -\sum_{i=1}^{M} p_i \, \log_2 p_i. \tag{2.2}$$

[1] plus précisemment, cette source est modélisée par une variable aléatoire discrète S

Cette quantité sera appelée *entropie* de l'alphabet $[S]$, ou encore entropie de source, par analogie avec la formule d'entropie d'un gaz en thermodynamique. Son unité est le shannon par symbole de source ou le *bit par symbole de source*.

Propriétés

P1. Si pour un certain i, $p_i = 1$ et $p_j = 0$ pour $j \neq i$, alors $H(S) = 0$.

P2. Si tous les M symboles de l'alphabet sont équiprobables, alors l'entropie est maximale et vaut $H(S) = \log_2 M$.

Pour montrer rigoureusement cette propriété, on se base sur l'*inégalité fondamentale de Gibbs* qui s'applique à M nombres positifs a_i et b_i tels que

$$\begin{aligned} \sum_{i=1}^{M} a_i &= 1 \\ \sum_{i=1}^{M} b_i &\leq 1 \end{aligned}$$

et qui énonce que

$$\sum_{i=1}^{M} a_i \log_2(b_i/a_i) \leq 0 \tag{2.3}$$

et dont la démonstration fait l'objet de l'exercice 11.

Il suffit alors de prendre $a_i = p_i$ et $b_i = 1/M$ dans cette inégalité pour obtenir

$$0 \geq \sum_{i=1}^{M} p_i \log_2(1/Mp_i) = \sum_{i=1}^{M} p_i \log_2(1/p_i) - \sum_{i=1}^{M} p_i \log_2 M = H(S) - \log_2 M.$$

Comme $H(S) = \log_2 M$ lorsque $p_i = 1/M$ pour $1 \leq i \leq M$, le résultat est démontré.

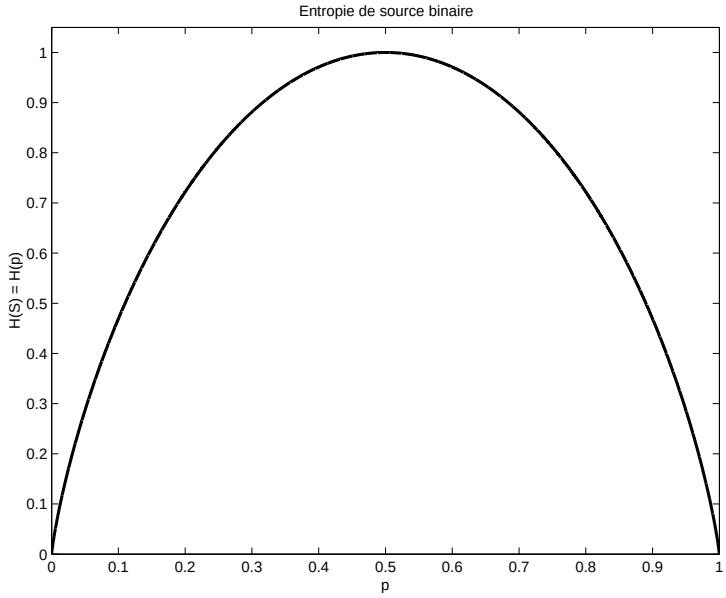

FIG. 2.1 – Entropie d'une source binaire.

Exemple 1 : entropie d'une source binaire.

L'entropie d'une source binaire émettant deux symboles, par exemple 0 et 1, avec des probabilités p et $(1-p)$, est représentée à la figure 1.

Exemple 2 : entropie d'un alphabet naturel.

On peut également calculer l'entropie d'un alphabet naturel, à partir des fréquences de caractères. Le tableau suivant donne ces fréquences (exprimées en pourcentages) pour l'anglais.

A	7.87	N	7.05
B	1.56	O	7.76
C	2.68	P	1.88
D	3.89	Q	0.10
E	12.66	R	5.95
F	2.55	S	6.31
G	1.87	T	9.78
H	5.74	U	2.80
I	7.07	V	1.02
J	0.10	W	2.15
K	0.60	X	0.16
L	3.94	Y	2.01
M	2.43	Z	0.07

L'entropie de l'anglais calculée à partir de ce modèle est d'environ 4.1 bits par lettre.

2.1.3 Entropie d'une source étendue

L'*extension d'ordre* n d'une source S sans mémoire est la source, notée S^n, dont l'alphabet est formé de toutes les concaténations possibles de n symboles de l'alphabet $[S]$. Par exemple, si $[S] = [0, 1]$, $[S^2] = [00, 01, 10, 11]$. Si l'alphabet $[S]$ comporte M caractères, l'alphabet $[S^n]$ en comportera M^n. L'entropie d'une telle source peut être aisément calculée grâce à la propriété suivante :

P3. $H(S^n) = nH(S)$.

2.1.4 Débit d'information

Le *débit d'information* d'une source, que nous noterons $\dot{H}$, est défini comme le produit de l'entropie de la source par le nombre moyen de symboles émis par seconde et sera exprimé en shannons/seconde ou en bits/seconde.

2.1.5 Redondance

La *redondance* R est la différence entre la valeur maximale possible de l'entropie (atteinte lorsque toutes les probabilités sont égales) et sa valeur réelle. Si l'alphabet comporte M symboles, on a donc

$$R(S) = H_{\max} - H(S) = \log_2 M - H(S). \tag{2.4}$$

Par exemple, dans le cas de l'anglais, on trouve $R = \log_2 26 - 4.1 = 0.6$ bit par lettre, en calculant $H(S)$ pour chaque lettre prise séparément, comme nous l'avons fait plus haut. En fait, certaines combinaisons de lettres sont beaucoup plus fréquentes que d'autres (par exemple les lettres t et h apparaissent souvent ensemble en anglais, et la paire th est beaucoup plus fréquente que la paire zq), ce qui nous oblige à considérer les fréquences des paires de lettres, puis des triplets, etc... On arrive alors, pour l'anglais, à $H(S) \approx 1.5$ bit par lettre, et donc $R(S) = 3.2$ bit par lettre. Une langue naturelle est toujours fortement redondante.

2.2 Codage entropique de source

Le codage est une opération de transformation du signal entre la source d'information et le canal de transmission. Les opérations inverses de décodage se font dans le récepteur. Il y a différentes sortes de codage :

- le *codage de source* vise à augmenter la quantité d'information transmise à travers le canal, et donc à réduire au maximum la longueur et la redondance des messages (on parle de *compression*).

- le *codage détecteur et correcteur d'erreurs*, au contraire du précédent, augmente la longueur et la redondance des messages pour permettre la détection et éventuellement la correction à la réception des erreurs provoquées par un canal perturbé par le bruit.

- le *codage d'émission* ou *de ligne* enfin vise une adaptation technique des signaux à celles du canal (bande passante, distorsion linéaire, etc.).

Nous allons étudier quelques techniques particulières de base de codage de source. L'alphabet de la source est

$$[S] = [s_1, s_2, \ldots, s_M]$$

tandis que celui du code est

$$[X] = [x_1, x_2, \ldots, x_D].$$

Par exemple, pour un code binaire ($D = 2$), $[X] = [0, 1]$. Les *mots* du code sont des suites finies de l'alphabet $[X]$:

$$[C] = [c_1, c_2, \ldots, c_M],$$

le codage de source établissant une correspondance biunivoque entre les symboles s_i et les mots c_i.

2.2.1 Codes à longueur constante

Le codage le plus simple consiste à donner à tous les mots du code la même longueur. Ce type de code est largement utilisé, un exemple en est le code ASCII, utilisant des mots de 7 bits.

2.2.2 Codes à longueur variable

Si on dispose des probabilités d'émission des différents symboles de l'alphabet $[S]$, il est plus judicieux d'utiliser des codes à longueur variable : le mot-code d'un symbole aura une longueur décroissant avec la probabilité d'émission du symbole. Un exemple d'un tel code est le Morse, où la lettre E, fréquemment utilisée, est codée par un seul symbole ($\cdot$), tandis que la lettre Q, plus rare, est codée par quatre symboles ($- - \cdot -$).

La première propriété qu'un code doit posséder est celle de **décodage unique**. En effet, à chaque succession de mots-code ne peut correspondre qu'une seule succession de symboles de la source. A titre d'exemple, considérons une source dont l'alphabet comporte quatre symboles, et

les trois codes suivants :

	Code A	Code B	Code C
s_1	0	0	0
s_2	01	10	01
s_3	10	110	011
s_4	11	1110	0111

Supposons que le message 0110 soit reçu. Avec le code A, il peut représenter la séquence $s_1 s_4 s_1$ tout comme la séquence $s_2 s_3$. Le code A n'est donc pas à décodage unique. Par contre, le code B est à décodage unique parce qu'il a un symbole (0) qui indique la fin de chaque mot. Il en est de même pour le code C est parce qu'il a un symbole (0) indiquant cette fois le début de chaque mot.

La seconde propriété qu'un code devrait posséder est d'être à **décodage instantané**, à savoir qu'à mesure que les séquences de symboles x_k de l'alphabet du code sont reçus, les mots c_i du code peuvent être déterminés sans s'inquiéter des symboles suivants. Par exemple, le code B défini ci-dessus est à décodage instantané, au contraire du code C. Supposons que la séquence reçue soit 00110. Avec le code B, dès la réception du premier symbole 0, le récepteur sait qu'il s'agit de s_1. Avec le code C par contre, aucune conclusion ne peut être tirée dès la réception du premier symbole. Il faut attendre le deuxième 0 pour décider que le premier représentait s_1.

Par conséquent, un code instantané doit être tel qu'aucun mot du code ne soit un *préfixe* d'un autre mot du code. (Si $c_i = x_1 x_2 \dots x_{l_i}$ est un mot du code, la suite de symboles $x_1 x_2 \dots x_k$, avec $k < l_i$, est un préfixe du mot c_i). On parle de code préfixe ou instantané.

Le décodage d'un code instantané revient à construire un arbre "D-aire". Un tel arbre débute par une racine, donnant naissance à D branches. Chaque branche se termine en un noeud, et est étiquettée avec un des symboles du code $x_1, \dots, x_D$. Chaque noeud de cette première génération est soit un noeud terminal (encore appelé feuille de l'arbre), ne donnant naissance à aucune branche, soit le point de départ de D nouvelles branches, étiquettées de nouveau avec un des symboles du code $x_1, \dots, x_D$, et ainsi de suite jusqu'à ce que tous les noeuds descendants soient des feuilles. Le chemin allant de la racine à chacune des feuilles de l'arbre définit une suite de symboles du code, qui sont les étiquettes successives de chacune des branches, et correspond donc à un des mots-codes. A cause de la construction de cet arbre, aucun mot-code n'est le préfixe d'un autre mot-code, et le code est donc instantané. Réciproquement, tout code préfixe peut être représenté par un arbre. La longueur de cet arbre (le nombre de couches) est la longueur maximale des mots du code.

2.2.3 Longueur moyenne des mots

Les codes à longueur variable ont pour but la minimisation de la *longueur moyenne* L des mots-code c_i,

$$L = \sum_{i=1}^{M} p_i l_i \tag{2.5}$$

où p_i sont les probabilités des symboles s_i représentés par les mots c_i de longueur l_i.

2.2.4 Inégalité de Kraft

Un code instantané de M mots-codes ayant les longueurs l_i existe si et seulement si

$$\sum_{i=1}^{M} D^{-l_i} \leq 1, \tag{2.6}$$

ce qui implique qu'on ne peut pas prendre trop de mots-code ayant une très faible longueur.

Démonstration de l'inégalité de Kraft

Nous faisons la démonstration dans le cas binaire ($D = 2$), l'extension au cas $D > 2$ est laissée au lecteur. Nous supposons également que M est fini, l'inégalité restant vraie pour M infini.

($\Leftarrow$) Soit C un code instantané. Montrons que les longueurs des mots-codes vérifient l'inégalité (2.6). Rappelons-nous qu'un code instantané binaire peut se mettre sous la forme d'un arbre binaire. Nous procédons par récurrence sur la longueur de l'arbre, qui est égale à la longueur maximale des mots du code $l_{\max}$. Nous vérifions tout d'abord que la propriété est vraie pour un arbre de longueur 1, c'est l'étape d'initialisation de la preuve. Nous supposons ensuite que le propriété a été vérifiée pour les arbres de longueurs $1, 2, \ldots, l_{\max}$ et nous montrons qu'elle l'est encore pour des arbres de longueur $(l_{\max} + 1)$, c'est l'étape de récurrence ou d'induction.

Etape d'initialisation : si la longueur de l'arbre est 1, cet arbre est fait de 2 branches qui se terminent par deux feuilles, et par conséquent le code est formé de $M = 2$ mots-codes, à savoir 0 et 1, chacun de longueur $l_1 = l_2 = 1$. La relation (2.6) est donc bien vérifiée (avec une égalité au lieu d'une inégalité).

Etape de récurrence : nous supposons que tout arbre de longueur $1, \ldots, l_{\max}$ représente des mots-codes dont les longueurs vérifient l'inégalité de Kraft. Soit un code instantané de longueur $(l_{\max} + 1)$. Les deux noeuds de la première couche donnent chacun naissance à un sous-arbre de longueur au plus égale à $l_{\max}$. Parmi les M mots-codes, soit M_1 le nombre de mots dont le premier bit est 0 et soit $M_2 = M - M_1$ le nombre de mots dont le premier bit est 1. Prenons un mot code quelconque $c_i = x_1 x_2 \ldots x_{l_i}$, de longueur $l_i \leq l_{\max} + 1$. Si on élimine le premier bit x_1 du mot-code c_i, on obtient un nouveau mot-code $c'_i = x_2 \ldots x_{l_i}$, de longueur $l'_i = l_i - 1 \leq l_{\max}$. Si $x_1 = 0$, le chemin correspondant à c'_i appartient au premier sous-arbre binaire, sinon le chemin correspondant à c'_i appartient au second sous-arbre binaire. Ceci permet de construire chaque mot-code c_i de longueur l_i à partir d'un mot-code c'_i de longueur $l_i - 1$, que l'on fait précéder d'un bit 0 ou 1. Comme les mots codes c'_i appartiennent à deux sous arbres qui ont chacun une longueur au plus égale à $l_{\max}$, l'hypothèse d'induction nous permet d'écrire que

$$\sum_{i=1}^{M_1} 2^{-(l_i - 1)} \leq 1$$

pour les M_1 mots-codes c'_i qui se situent dans le premier sous-arbre binaire, et de même

$$\sum_{i=M_1+1}^{M_1+M_2} 2^{-(l_i - 1)} \leq 1$$

pour les M_2 autres-codes c'_i qui se situent dans le second sous-arbre binaire. En additionant les membres respectifs de ces deux inégalités, on trouve

$$\sum_{i=1}^{M_1} 2^{-(l_i-1)} + \sum_{i=M_1+1}^{M_1+M_2} 2^{-(l_i-1)} = \sum_{i=1}^{M_1+M_2} 2^{-(l_i-1)} \leq 2,$$

qui est bien l'inégalité (2.6) après simplifcation par le facteur 2 des deux membres de l'inégalité. Ceci établit l'étape de récurrence.

($\Rightarrow$) Supposons à présent que $l_1, \ldots, l_M$ vérifient (2.6), et construisons un code instantané dont les mots-codes ont ces longueurs. Sans perte de généralité, on peut étiquetter ces longueurs dans l'ordre croissant $l_1 \leq l_2 \leq \ldots \leq l_M$. Nous construisons un arbre binaire de longueur l_M, qui comporte donc 2^{l_M} feuilles. Prenons le premier noeud de la couche se situant à une longueur l_1 dans l'arbre. Nous faisons correspondre le chemin de la racine de l'arbre jusqu'à ce noeud avec le premier mot-code c_1, et nous éliminons tous les $2^{l_M-l_1}$ descendants de ce noeud, de manière en faire une feuille de l'arbre. Nous prenons ensuite le premier noeud appartenant à la couche se situant à une longueur l_2 dans l'arbre restant après l'élagage des descendants du premier noeud. Nous faisons correspondre le chemin de la racine de l'arbre jusqu'à ce noeud avec le deuxième mot-code c_2, et nous éliminons tous les $2^{l_M-l_2}$ descendants de ce noeud, de manière en faire une nouvelle feuille de l'arbre. En procédant de la sorte, nous construisons un code préfixe ayant les longueurs spécifiées $l_1, \ldots, l_M$, CQFD.

On peut se demander si la contrainte (2.6) imposée sur les longueurs des mots peut être rendue moins stricte, si on impose simplement que le code soit à décodage unique, mais pas nécessairement instantané. On peut montrer qu'il n'en est rien (Théorème de McMillan), et que donc on ne perd rien à imposer que le code à décodage unique soit également instantané.

2.2.5 Efficacité d'un code

Comme l'inégalité de Kraft doit être satisfaite, on peut utiliser l'inégalité de Gibbs (2.3) avec $b_i = D^{-l_i}$ et $a_i = p_i$ pour obtenir

$$H(S) = -\sum_{i=1}^{M} p_i \log_2 p_i \leq -\sum_{i=1}^{M} p_i \log_2 D^{-l_i} = \sum_{i=1}^{M} p_i l_i \log_2 D = L \log_2 D$$

d'où

$$L \geq \frac{H(S)}{\log_2 D} = L_{\min}.$$

L'*efficacité* du code, définie par

$$\eta = \frac{L_{\min}}{L} = \frac{H(S)}{L \log_2 D} \tag{2.7}$$

est donc toujours inférieure ou égale à 1. On appelle *redondance du code* la quantité $\rho = 1 - \eta$.

2.2.6 Codes optimaux

Existe-t-il un algorithme de codage qui maximise l'efficacité du code ? Nous allons voir que la réponse est affirmative. Avant de le décrire, nous établissons d'abord trois propriétés que doit avoir un code instantané binaire optimal.

P4. Les mots les plus probables sont les plus courts : si $p_k > p_j$ alors $l_k \leq l_j$.

P5. Les deux mots les plus longs ont la même longueur.

P6. Les deux mots les plus longs ne diffèrent que par le dernier symbole binaire.

Démontrons ces propriétés.

P4 : Supposons que $p_k > p_j$ mais que $l_k > l_j$, et désignons par L la longueur moyenne des mots de ce code. Echangeons les mots-codes c_k et c_j. Dans ce cas la différence entre les longueurs moyennes L' et L des codes respecitvement après et avant cet échange est

$$\begin{aligned} L' - L &= p_k l_j + p_j l_k - p_k l_k - p_j l_j \\ &= (p_k - p_j)(l_j - l_k) \\ &< 0, \end{aligned}$$

ce qui montre que le nouveau code obtenu après échange des mots-codes c_k et c_j est plus efficace.

P5 : Supposons que les deux mots-codes les plus longs (et donc, en vertu de P4, les moins probables), aient une longueur différente. Comme le code est instantané, aucun mot-code ne peut être le préfixe d'un mot code plus long. Par conséquent, nous pouvons supprimer le dernier bit du mot le plus long, sans que le mot ainsi raccourci ne prête à confusion avec un autre mot du code. Ceci réduit la longueur moyenne du code, et donc le code avec les deux mots les moins probables ayant des longueurs différentes ne pouvait être optimal. Pour que le code soit optimal, il faut donc que les deux mots les moins probables aient la même longueur.

P6 : Supposons que tous les mots-codes les plus longs (et donc, en vertu de P5, de même longueur), diffèrent par un bit autre que le dernier. A nouveau, comme aucun mot-code ne peut être le préfixe d'un mot-code plus long, nous pourrions alors éliminer le dernier bit de deux de ces mots-codes, sans que les mots ainsi raccourcis ne puissent être confondus avec un autre mot du code. Ceci impliquerait à nouveau que le code original n'était pas optimal.

2.2.7 Codage de Huffman (cas binaire)

L'algorithme de codage de Huffman comporte une première partie qui est un processus de réduction de l'alphabet de la source :

Etape 1 : Ordonner l'ensemble des M symboles de l'alphabet $[S]$ dans l'ordre des probabilités décroissantes. A titre d'exemple, prenons $[S] = [s_1, s_2, s_3, s_4, s_5]$ avec $p_1 = 0.4$, $p_2 = p_3 = 0.2$, $p_4 = p_5 = 0.1$.

Etape 2 : Combiner les deux symboles les moins probables en un seul symbole. On se retrouve

alors avec un nouvel alphabet de source $[S']$ de $M-1$ symboles. Pour notre exemple, on a $[S'] = [s_1, s_2, s_3, s'_4]$ avec $p_1 = 0.4$, $p_2 = p_3 = p'_4 = 0.2$.

Etape 3 : retour à l'étape 1, jusqu'à ce que l'alphabet de source soit réduit à 2 symboles.

Pour notre exemple, on a donc (complétez l'arbre binaire) :

s_i	p_i			
s_1	0.4	0.4	0.4	0.6
s_2	0.2	0.2	0.4	0.4
s_3	0.2	0.2	0.2	
s_4	0.1	0.2		
s_5	0.1			

A la fin de ce processus, on n'a plus que deux symboles qu'il est très facile d'encoder : le premier sera 0 et le second 1 (ou vice-versa). Ensuite, on remonte l'arbre construit par le processus de réduction, l'un des deux symboles précédents est séparé en deux autres symboles, en ajoutant un second symbole 0 pour l'un d'eux, et un 1 pour l'autre, et ainsi de suite. Ce processus d'éclatement se déroule donc comme suit :

s_i		p_i						
s_1	**1**	0.4	**1**	0.4	**1**	0.4	**0**	0.6
s_2	**01**	0.2	**01**	0.2	**00**	0.4	**1**	0.4
s_3	**000**	0.2	**000**	0.2	**01**	0.2		
s_4	**0010**	0.1	**001**	0.2				
s_5	**0011**	0.1						

La manière dont les symboles 0 et 1 sont assignés empêche un mot d'être le préfixe d'un autre mot, et le code est instantané.

On a donc les mots codes suivants :

s_i	$\longrightarrow$	c_i
s_1	$\longrightarrow$	1
s_2	$\longrightarrow$	01
s_3	$\longrightarrow$	000
s_4	$\longrightarrow$	0010
s_5	$\longrightarrow$	0011

La longueur moyenne d'un mot-code étant $L^H = 2.2$ bits par mot-code, et l'entropie de la source valant $H(S) = 2.122$ bits par mot-code, l'efficacité du code est $\eta^H = 0.9645$.

Le codage de Huffman n'est pas toujours unique. Pour notre exemple, on aurait tout aussi bien pu prendre les processus de réduction et d'éclatement suivants :

s_i	p_i			
s_1	0.4	0.4	0.4	0.6
s_2	0.2	0.2	0.4	0.4
s_3	0.2	0.2	0.2	
s_4	0.1	0.2		
s_5	0.1			

et

s_i		p_i						
s_1	**00**	0.4	**00**	0.4	**1**	0.4	**0**	0.6
s_2	**10**	0.2	**01**	0.2	**00**	0.4	**1**	0.4
s_3	**11**	0.2	**10**	0.2	**01**	0.2		
s_4	**010**	0.1	**11**	0.2				
s_5	**011**	0.1						

La longueur moyenne, et donc l'efficacité, des deux codes est la même, mais les mots-code ne le sont pas.

2.2.8 Optimalité du code binaire de Huffman

Le code de Huffman est un code instantané optimal. Nous le montrons par récurrence, dans le cas $D = 2$, sur le nombre de mots-codes M. Désignons par $C^H(M)$ le code de Huffman binaire d'une source S de M symboles.

Etape d'initialisation : si $M = 2$, il est clair que le code $C^H(2)$ dont les mots-code sont $[C^H(2)] = [0, 1]$ est optimal.

Etape de récurrence : Supposons que le code $C^H(M-1)$ soit optimal pour toute source de $(M-1)$ symboles.

Considérons une source S de M symboles ($[S] = [s_1, s_2, \ldots, s_{M-2}, s_{M-1}, s_M]$, ordonnés dans l'ordre décroissant des probabilités de leurs apparitions (en d'autres termes, $p_1 \geq p_2 \geq \ldots p_{M-2} \geq p_{M-1} \geq p_M$).

Nous cherchons un code instantané $C(M)$ de cette source qui soit optimal. La propriété P4 impose d'avoir c_{M-1} et c_M comme mots-codes les plus longs. La propriété P5 impose d'avoir $l_{M-1} = l_M$. Enfin, la propriété P6 nous permet d'écrire ces deux mots-code comme $c_{M-1} = c'_{M-1}0$ et $c_M = c'_{M-1}1$, où c'_{M-1} est un mot formé des $l_{M-1} - 1$ premiers bits communs à c_{M-1} et c_M. Considérons ensuite la source réduite S' obtenue à partir de S, en combinant les deux symboles s_{M-1} et s_M en un seul s'_{M-1}, qui apparait avec une probabilité $p'_{M-1} = p_{M-1} + p_M$, et en gardant tous les autres symboles identiques : $s'_i = s_i$ pour $1 \leq i \leq M-2$. Observons que le processus de réduction de l'algorithme de Huffman transforme précisemment la source S en source S' de cette manière. Soit $C'(M-1)$ le code de cette source S', défini par l'alphabet $[C'(M-1)] = [c_1, \ldots, c_{M-2}, c'_{M-1}]$. A cause des observations faites ci-dessus, $C'(M-1)$ est un code instantané, avec la longueur de c'_{M-1} valant $l'_{M-1} = l_{M-1} - 1$.

Les longueurs moyennes des codes $C(M)$ et $C'(M-1)$ sont liées par la relation suivante :

$$\begin{aligned} L(C(M)) &= \sum_{i=1}^{M} p_i l_i \\ &= \sum_{i=1}^{M-2} p_i l_i + p_{M-1} l_{M-1} + p_M l_M \\ &= \sum_{i=1}^{M-2} p_i l_i + (p_{M-1} + p_M) l_{M-1} \\ &= \sum_{i=1}^{M-2} p_i l'_i + (p_{M-1} + p_M)(l'_{M-1} + 1) \\ &= \sum_{i=1}^{M-1} p'_i l'_i + p_{M-1} + p_M \\ &= L(C'(M-1)) + p_{M-1} + p_M \end{aligned}$$

et ne diffèrent donc que d'une constante $(p_{M-1} + p_M)$ indépendante de l'algorithme de codage. Par conséquent, minimiser la longueur moyenne du code $C(M)$, à savoir $L(C(M))$, est équivalent à minimiser la longueur moyenne du code $C'(M-1)$, à savoir $L(C'(M-1))$. L'hypothèse d'induction nous permet d'affirmer que le code optimal de la source S' de $(M-1)$ symboles est le code de Huffman $C^H(M-1)$, et donc le code $C'(M-1)$ est le code de Huffman $C^H(M-1)$. Le processus d'assignement des bits 0 et 1 de l'algorithme de Huffman produit exactement les mots-codes $c_1, \ldots, c_{M-2}$, $c_{M-1} = c'_{M-1}0$ et $c_{M-1} = c'_{M-1}1$, ce qui montre que le code optimal $C(M)$ est lui aussi le code de Huffman $C^H(M)$.

2.2.9 Codage de Shannon-Fano

Le codage de Huffman donne les longueurs de chaque mot-code en fonction de sa probabilité d'émission, mais cette méthode fait dépendre chaque longueur de l'ensemble complet des probabilités. Le codage de Shannon-Fano permet au contraire de passer directement de chaque probabilité individuelle p_i à la longueur l_i du mot-code c_i.

On part de l'observation que pour tout p_i il y a un entier l_i tel que

$$\log_D(1/p_i) \leq l_i < \log_D(1/p_i) + 1. \tag{2.8}$$

Cette équation définit la longueur de chaque mot-code. Dans le cas de l'exemple du paragraphe précédent, $D = 2$ et le codage de Shannon-Fano donne donc

$$l_1 = 2, \quad l_2 = l_3 = 3, \quad l_4 = l_5 = 4.$$

Il faut encore vérifier qu'un tel code instantané existe. Dans ce but, on élimine les logarithmes de la relation (2.8), ce qui donne

$$1/p_i \leq D^{l_i} < D/p_i,$$

ou encore

$$p_i \geq D^{-l_i} > p_i/D.$$

Comme $\sum_{i=1}^{M} p_i = 1$, en sommant ces inégalités pour $1 \leq i \leq M$, on a que

$$1 \geq \sum_{i=1}^{M} D^{-l_i} > 1/D$$

ce qui donne l'inégalité de Kraft et montre qu'il existe un code instantané possédant les longueurs de Shannon-Fano.

Une fois les longueurs calculées par (2.8), les mots-code sont simplement assignés dans l'ordre, de la manière suivante :

$$\begin{array}{ccc} s_i & \longrightarrow & c_i \\ s_1 & \longrightarrow & 00 \\ s_2 & \longrightarrow & 010 \\ s_3 & \longrightarrow & 011 \\ s_4 & \longrightarrow & 1000 \\ s_5 & \longrightarrow & 1001 \end{array}$$

La longueur moyenne des mots est maintenant $L^{SF} = 2.8$ bits par mot-code, et l'efficacité de ce code n'est plus que $\eta^{SF} = 0.758$. Le code de Shannon-Fano est plus simple que le code de Huffman, mais ne peut évidemment être plus efficace, puisque ce dernier est optimal.

Il est facile de donner une borne supérieure de la longueur moyenne des mots à partir de (2.8). En effet, en multipliant cette inégalité par p_i et en sommant, on a

$$\sum_{i=1}^{M} p_i \log_D(1/p_i) \leq \sum_{i=1}^{M} p_i l_i < \sum_{i=1}^{M} p_i \log_D(1/p_i) + 1$$

ou encore comme

$$\log_D(1/p_i) = \frac{\log_2(1/p_i)}{\log_2 D}$$

et avec la définition de $H(S)$,

$$\frac{H(S)}{\log_2 D} \leq L^{SF} < \frac{H(S)}{\log_2 D} + 1. \tag{2.9}$$

2.2.10 Extension d'un code

Les sections précédentes nous ont donc permis de montrer que

$$\frac{H(S)}{\log_2 D} \leq L^{H} \leq L^{SF} < \frac{H(S)}{\log_2 D} + 1. \tag{2.10}$$

Pour une source étendue S^n, et le code étendu C^n qui lui correspond (que ce soit en utilisant l'algorithme de Huffman ou de Shannon-Fano), la longueur moyenne d'un mot-code de C^n est

désignée par L_n et le nombre moyen de symboles de l'alphabet de code par symbole de la source S est donc L_n/n. L'inégalité (2.10) implique alors, grâce à la propriété P3,

$$n\frac{H(S)}{\log_2 D} \leq L_n < n\frac{H(S)}{\log_2 D} + 1$$

ou encore, si on divise les membres de cette inégalité par n,

$$\frac{H(S)}{\log_2 D} \leq \frac{L_n}{n} < \frac{H(S)}{\log_2 D} + \frac{1}{n}.$$

En prenant la limite pour n très grand, on a donc

$$\lim_{n\to+\infty} \frac{L_n}{n} = \frac{H(S)}{\log_2 D} = L_{\min}$$

ce qui montre que par un codage approprié (codage par groupes de symboles de la source), on peut amener le nombre moyen de symboles de l'alphabet de code par symbole de la source S aussi près que l'on veut du minimum absolu. Ce résultat est connu comme premier théorème de Shannon. L'effet de codes étendus sera étudié à l'exercice 10.

2.3 Compression et codage de source en pratique

Le code de Huffman est optimal, mais présente deux inconvénients majeurs. Tout d'abord, la construction de l'arbre binaire est extrêmement complexe en nombre d'opérations. Ensuite, elle nécessite la connaissance des probabilités d'émission de chaque symbole de la source, ce qui est pratiquement impossible à obtenir pour des sources réelles. L'algorithme de Huffman ne peut être utilisé qu'avec des alphabets de source de taille raisonnable (Par exemple, le codage des opérations sur le processeur Intel 432 fait appel à des techniques du type Huffman pour la compression des programmes). De tels alphabets sont obtenus après plusieurs étapes préliminaires de compression utilisant d'autres algorithmes, dont certains sont mentionnés ci-dessous et seront étudiés dans les cours de traitement du signal, des images, de l'audio, ainsi naturellement que dans ceux de théorie de l'information et du codage.

2.3.1 Codage par longueur de plage ("Run length coding") et fax

Un algorithme de codage assez simple pouvant s'appliquer à des images graphiques à deux niveaux (noir (N), blanc (B)), dont l'exemple typique est le fax, est le codage par longueur de plage ("Run length coding"). L'image est découpée en carrés élémentaires, les pixels. Prenons une ligne d'une telle image, qui est donc une succession de pixels blancs et noirs. Par exemple, prenons une ligne

BBBBNNNBBBNNBBBBNNNBBBBBBBBBBBBBBBBBBBBBBBNNNNBBBBBBN

Au lieu d'encoder chaque pixel blanc par un bit 0 et chaque pixel noir par un bit 1, ce qui ne procure aucune compression, on compte le nombre de pixels blancs et noirs successifs (qu'on appelle plages), et on code cette longueur par un entier, en notation décimale. En se mettant d'accord qu'une ligne commence par un pixel blanc, on aurait pour l'exemple ci-dessous la séquence

4 3 3 2 4 3 24 4 6 1

Il y a d'autres algorithmes de codage par longueur de plage possibles, nous en verrons un exemple dans l'exercice 13.

Ensuite on code ces symboles entiers par un code binaire approprié, soit à longeur constante, soit plus efficacement par un code de Huffman (les symboles de la source sont les entiers qui représentent les longueurs des plages de pixels blancs et noirs). Une des normes recommandées par le CCITT (Comité Consultatif International Téléphonique et Télégraphique, devenu ITU, International Telecommuncation Union) propose un algorithme de Huffman modifié pour le codage des plages. La longueur d'une ligne valant 1728 pixels, les longueurs de plages p sont décomposées en base 64 comme

$$p = 64m + n$$

avec $m = 0, 1, \ldots, 27$ et $n = 0, 1, \ldots, 63$. Les probabilités d'apparition des différents symboles m, n sont ajustées à partir de huit images tests, qu'on peut trouver sur le site web `http ://www.cs.waikato.ac.nz/ singlis/ccitt.html`. Quelques mots-codes sont pour les plages blanches courtes ($m = 0$) sont

$s_i = (n, \mathrm{B})$	$\longrightarrow$	c_i
1	$\longrightarrow$	000111
2	$\longrightarrow$	0111
3	$\longrightarrow$	1000
4	$\longrightarrow$	1011
5	$\longrightarrow$	1100
6	$\longrightarrow$	1110
	$\vdots$	
24	$\longrightarrow$	0101000
	$\vdots$	

et pour les plages noires

$s_i = (n, \mathrm{N})$	$\longrightarrow$	c_i
1	$\longrightarrow$	010
2	$\longrightarrow$	11
3	$\longrightarrow$	10
4	$\longrightarrow$	011
5	$\longrightarrow$	0011
6	$\longrightarrow$	0010
	$\vdots$	

Dans la ligne donnée en exemple ci-dessus, on aurait donc

$$10111010001110111001010000111100010$$

2.3.2 Quelques autres algorithmes de codage

Pour des textes ASCII, on utilise un troisième algorithme de compression, après le codage par longueur de plage, et avant un codage de Huffman, qui est l'algorithme de Lempel-Ziv. Cet algorithme sera vu dans le cours de théorie de l'information en 4ème année. Les commandes Unix `compress` et `zip` (`winzip` sur Windows) utilisent cet algorithme.

Tous les algorithmes de compression décrits ci-dessus sont sans pertes (lossless compression), ce qui veut dire qu'ils sont uniquement décodables. Pour des sources audio ou vidéo, il est prohibitif d'utiliser un algorithme de compression sans pertes, et les premières étapes de compression s'effectuent avec une perte d'information (lossy compression). Ainsi, les standards de compression audio/video JPEG, MPEG, MP3, etc se font avec pertes. Ils seront étudiés dans les cours de traitement des signaux, audio et images, lors des 3ème et 4ème années.

2.4 Références Bibliographiques

T. Cover and J. Thomas, *Elements of Information Theory*. Wiley & Sons, New York, 1991.

R. W. Hamming, *Coding and information theory*. Prentice-Hall, Englewood Cliffs, NJ, 1986.

R. B. Ash, *Information Theory*. Dover Publications Inc, New York, 1990.

Gérard Battail, *Théorie de l'Information*. Ed Masson, 1997.

C. E. Shannon, *Prediction and Entropy of Printed English*. Bell System Technical Journal, Vol 30 (1951), pp.50-64 Reprinted in D. Slepian, editor, Key Papers in the Development of Information Theory, IEEE Press, NY, 1974.

2.5 Exercices

Rappel : Pour une série arithmético-géométrique, on a

$$\sum_{i=0}^{+\infty}(a+id)\,r^i = \frac{a}{1-r} + \frac{rd}{(1-r)^2}$$

si $-1 < r < 1$.

1. Soit un alphabet $S = [s_1, s_2, s_3, s_4]$. Quelle est, parmi les deux distributions de probabilité suivantes, celle qui donnera la plus grande entropie :

(a) $p_1 = 0.5$, $p_2 = 0.25$, $p_3 = 0.125$ et $p_4 = 0.125$ ou

(b) $p_1 = 0.5$, $p_2 = 0.3$, $p_3 = 0.1$ et $p_4 = 0.1$?

2. Quelle serait l'entropie d'une source pouvant délivrer un nombre infini de symboles ayant les probabilités

$$p_1 = 1/2, p_2 = 1/4, p_3 = 1/8, \ldots, p_i = 1/2^i, \ldots ?$$

3. Etablir la propriété P3 de l'entropie d'une source étendue pour $n = 2$.

4. Une image de télévision monochrome est décomposée en 625 lignes horizontales et chaque ligne est décomposée à son tour en 625 pixels dont les intensités correspondent à la source représentée. Ces intensités sont uniformément quantifiées par 256 niveaux de probabilité égale. Les luminosités de tous les points sont supposées être indépendantes, et 25 images sont transmises par seconde. Quel est le débit d'information de la source d'images ?

5. Un système de télévision transmet en 20 msec une image monochrome dont les points ont une intensité uniformément quantifiée sur 10 niveaux de probabilité égale. Combien de temps mettra ce système pour transmettre une image en couleurs, s'il y a (en plus des 10 niveaux de contraste) 30 teintes de couleur, supposées à nouveau équiprobables ?

6. Les codes suivants sont-ils à décodage unique ? à décodage instantané ?

	Code D	Code E	Code F
s_1	0	0	00
s_2	01	10	01
s_3	011	001	10
s_4	111	0011	111

7. Une source S génère sept symboles dont les probabilités sont

$$p_1 = p_2 = 1/3, \quad p_3 = p_4 = 1/9, \quad p_5 = p_6 = p_7 = 1/27.$$

a) Construire un code binaire ($[X] = [0, 1]$) de Huffman et calculer son efficacité. Y a-t-il plusieurs codes de Huffman possibles ?

b) Construire un code binaire de Shannon-Fano et calculer son efficacité.

8. Un code à longueur constante peut-il avoir une efficacité égale à 1 ? Si oui, donner un exemple de source pour laquelle c'est le cas, si non, expliquer pourquoi.

9. Quel est, des deux codes de Huffman présentés comme exemples à la section 2.2.7, celui qui vous semble préférable ?

10. Une source binaire génère les symboles s_1 et s_2 avec respectivement les probabilités $p_1 = 0.9$ et $p_2 = 0.1$. Les deux symboles sont indépendants.

a) Quelle est l'efficacité du code de Huffman de cette source ?

b) Calculer la croissance de l'efficacité si au lieu de coder symbole par symbole comme en a), on codait par groupe de $n = 2$ symboles.

c) Répéter la question b) pour des groupes de $n = 3$ symboles.

d) Quelle est la longueur moyenne minimale des mots-code par symbole binaire que l'on peut espérer atteindre en faisant des groupes de n symboles avec n aussi grand que l'on veut ?

11. Démontrer l'inégalité de Gibbs (2.3). Conseil : remarquer que pour tout réel y,

$$e^y \geq 1 + y$$

ou encore en prenant le logarithme naturel de cette expression, si $y > -1$,

$$y \geq \ln(1 + y)$$

En particulier, en prenant $y = (b_i/a_i) - 1$, on peut démontrer l'inégalité de Gibbs.

12. Soit S une source d'information pouvant délivrer $M+N$ symboles $s_1, \ldots, s_M, s_{M+1}, \ldots, s_{M+N}$, avec les probabilités $p_1, \ldots, p_M, p_{M+1}, \ldots, p_{M+N}$. Soit S' une source d'information pouvant délivrer $M + 1$ symboles $s'_1, \ldots, s'_M, s'_{M+1}$, avec les probabilités $p'_1, \ldots, p'_M, p'_{M+1}$. Supposons que $p'_i = p_i$ pour $1 \leq i \leq M$ et que $p'_{M+1} = p_{M+1} + \ldots + p_{M+N}$.

a) Montrer que $H(S) \geq H(S')$. (Conseil : observez que $p'_{M+1} \geq p_i$ pour tout $M+1 \leq i \leq M+N$)

b) Montrer que $H(S) \leq H(S') + p'_{M+1} \log_2 N$. (Conseil : partez l'inégalité de Gibbs, avec les $a_i = p_i$ et les b_i judicieusement choisis).

c) Peut-on avoir $H(S) = H(S') + p'_{M+1} \log_2 N$? Si oui, donner un exemple de source S (et donc S') qui vérifie cette égalité.

13. Soit S une source d'information délivrant 8 symboles $s_1, s_2, \ldots, s_8$ indépendamment les uns des autres, avec les probabilités $p_i = P(S = s_i)$ valant respectivement, pour $1 \leq i \leq 8 : p_1 = 1/2$, $p_2 = 1/4$, $p_3 = 1/8$, $p_4 = 1/16$, $p_5 = 1/32$, $p_6 = 1/64$, $p_7 = p_8 = 1/128$.

a) Calculer l'entropie de cette source.

b) Donner un code optimal à longueur *constante* (les 8 mots du code ont la même longueur aussi courte que possible pour un code à décodage unique). Que vaut son efficacité ?

c) Donner un code optimal à longueur *variable* (les 8 mots du code ne doivent pas nécessairement avoir la même longueur, mais la longueur moyenne doit être aussi courte que possible pour un code à décodage unique). Que vaut son efficacité ?

d) Si vous pouvez coder la source non pas symbole par symbole comme aux deux sous-questions précédentes, mais en groupant les symboles par paire (donc vous codez les paires de symboles $s_i s_j$ pour tout $1 \leq i, j \leq 8$ par un mot-code, au lieu de coder chaque symbole s_i par un mot-code pour $1 \leq i \leq 8$), vous obtenez 64 mots-codes. Dans le cas où tous les 64 mots doivent avoir la même longueur (code optimal longueur constante), ce codage par paire de symboles de la source S vous permet-il d'obtenir une efficacité du code plus élevée que celle trouvée à la sous-question 2 ? Pourquoi ?

e) Même question que d), mais pour un code optimal à longueur variable : dans le cas où tous les 64 mots ne doivent pas avoir la même longueur, le codage par paire de symboles de la source S vous permet-il d'obtenir une efficacité du code plus élevée que celle trouvée à la sous-question 3 ? Pourquoi ?

14. Une source binaire S représente une ligne de fax, dans laquelle la probabilité d'apparition d'un pixel noir est p, et donc celle d'un pixel blanc est $(1-p)$, avec $0 < p < 1$. La couleur d'un pixel est indépendante de celle des autres pixels (ce qui en réalité est faux, mais nécessiterait de travailler avec les chaînes de Markov, qui seront vues au cours de modèles stochastiques pour les communications). Soit $H(S)$ l'entropie de cette source.

On code une suite de pixels générés par cette source par le codage par longueur de plage suivant : on compte le nombre de pixels blancs précédant un pixel noir. Ainsi, la séquence

BBBBNNNBBBNNBBBBNNNBBBBBBBBBBBBBBBBBBBBBBBNNNNBBBBBBN

est encodée comme

4 0 0 3 0 4 0 0 24 0 0 0 6

Soit S' la source obtenue après ce codage par longueur de plage, qui peut donc générer les symboles $s'_i = i$ pour tout $i \in \mathbb{N}$, si on fait l'hypothèse supplémenataire que la longueur de la ligne est infinie.

a) Calculer la probabilité d'émission de la longeur i, c'est-à-dire $P(S' = i)$, à partir de p (Observer que la longueur est i si et seulement si la source S a émis i pixels blancs suivis d'un pixel blanc).

b) Calculer l'entropie $H(S')$ de la source S'.

c) Comparer $H(S')$ et $H(S)$.

d) Quels sont les avantages/désavantages de ce code par longueur de plage, sur celui décrit dans la section 2.3.1 ?