Traitement de l'information et vision artificielle
TP3 - Compression sans pertes et codes de Huffman.

Yonatan DELORO

Pour le 15 avril 2017

1 Compression d'une séquence de caractères

0. La distribution des fréquences d'apparitions de l'alphabet en anglais est donnée en figure 1.

1. Rappelons le principe du code Huffman qui consiste à associer aux caractères les plus fréquents les codes les plus courts. Pour ce faire, le code de Huffman se construit récursivement. S'il n'y a qu'un caractère à encoder, le code optimal est bien entendu le code vide. Sinon, supposons que l'on dispose d'un code pour un alphabet de $K-1$ caractères et qu'on l'on souhaite construire un code d'un alphabet A de K caractères. Premièrement, on trie ces caractères selon leurs fréquences ou leurs probabilités d'apparition : $p_1 \geq p_2 \geq ... \geq p_K$. On associe alors aux deux caractères les moins fréquents, de fréquences p_{K-1} et p_K, les codes respectifs $c_{K-1}0$ et $c_{K-1}1$ où $c = (c_i)_{0<i<K-1}$ correspond aux codes respectifs des $K-1$ caractères d'un alphabet B de fréquences d'apparition $p_1,\ p_2, ...\ p_{K-2},\ p_{K-1}+p_K$. Tous les autres caractères de A gardent les mêmes codes que ceux de l'alphabet de B ayant mêmes probabilités.

2. Comme on l'observe en figure 2, la représentation obtenue par codage de Huffman encode un arbre, dans lequel les feuilles correspondent aux différents caractères à encoder, et dans lequel les noeuds intérieurs ont pour poids la somme des fréquences d'apparition des feuilles qui en dérivent. Ces poids correspondent à la hauteur des rectangles dans la représentation de la figure 1. On obtient alors le mot code d'une feuille en empruntant depuis la racine les arêtes qui permettent d'y échouer, où 0 encode la branche de gauche et 1 la branche de droite depuis un noeud quelconque.

Une itération de la récursion de Huffman revient en effet à associer aux deux caractères les moins fréquents deux feuilles dérivant du noeud portant comme poids la somme de leurs fréquences d'apparition. Ce noeud intérieur est interprété à l'itération suivante comme un caractère où l'on associera à nouveau les deux symboles les moins fréquents, et ce jusqu'à remonter à une unique racine.

En figure 3, on présente les itérations de l'algorithme de Hufman pour encode un alphabet de 5 caractères a, b, c, d, e de fréquences respectives $0.1, 0.2, 0.2, 0.2, 0.25$.

Ainsi, à chaque code de Huffman correspond naturellement un arbre. Ceci montre directement que le code de Huffman est préfixe (seules les feuilles correspondent aux caractères à encoder, ou encore il n'y a pas de caractère en un noeud intérieur). Une autre manière de le voir est qu'il est instantanné (il suffit de lire le code en descendant les branches de l'arbre, dès

qu'on atterrit sur une feuille le mot code est bien terminé et la feuille nous donne le caractère associé, on peut alors passer au suivant en repartant de la racine).

3.a Les codes pour les 26 lettres de A à Z obtenues avec l'algorithme de Huffman sont donnés en figure 4.

3.b On calcule l'entropie d'une source X variable aléatoire discrète générant un caractère de l'alphabet. En notant $\mathcal{A}$ l'alphabet des 26 lettres, l'entropie est définie par :

$$H(X) = -\sum_{x \in \mathcal{A}} p(x) log_2(x)$$

et peut donc s'interpréter comme la quantité moyenne d'information délivrée par la source. L'entropie de la distribution vaut 4.1758.

3.c La longueur moyenne $R_X(c)$ du code c de Shannon s'écrit, en notant $l(m)$ la longueur du mot m :

$$R_X(c) = \sum_{x \in \mathcal{A}} p(x) l(c(x))$$

La longueur moyenne du code de Huffman vaut 4.2050.

3.d Un code de Shannon, pour une source dont les réalisations ont pour probabilités p_k, est un code préfixe dont les longueurs des mots codes sont les $l_k = -\lfloor log_2(p_k) \rfloor$. Comme $l_k \geq -log_2(p_k)$, on a $\sum_k 2^{-l_k} \leq 1$, et la réciproque de l'inégalité de Kraft donne donc l'existence d'un tel code. De plus, si on appelle $c_{Shannon}$ ce code, sa longueur moyenne pour notre alphabet $\mathcal{A}$ vaut :

$$R_X(c_{Shannon}) = -\sum_{x \in \mathcal{A}} p(x) \lfloor log_2(p(x)) \rfloor = 4.5806$$

.

3.e Un encodage qui attribuerait le même nombre de bits pour chaque lettre aurait besoin de k bits tel que 2^k soit plus grand ou égal au nombre de lettres N à encoder (0 ou 1 pour chaque bit). Par conséquent, la longueur minimale d'un tel encodage est égal à $\lfloor log_2(26) \rfloor + 1 =$ 5 bits.

Le taux de compression $\tau_{c_{Huffman}}$ obtenu par codage $c_{Huffman}$ de Huffman vis-à-vis d'un tel encodage vaut donc : $\tau_{c_{Huffman}} = \frac{5 - R_X(c_{Huffman})}{5}$ soit 15,90 %

De même, $\tau_{c_{Shannon}} = \frac{log_2(N) - R_X(c_{Shannon})}{log_2(N)}$ soit 8,39 %

4. Pour construire un tel code, on commence toujours par trier les caractères à encoder par ordre croissant de leurs probabilités d'apparition : $p_1 \geq p_2 \geq ... \geq p_K$. Puis on divise les caractères en deux groupes équilibrés (c'est-à-dire en constituant le premier groupe des i caractères de probabilités $p_1 \geq p_2 \geq ... \geq p_i$ où i est le plus petit entier tel que $p_1 + p_2 + ... + p_i \geq p_{i+1} + ... + p_K$, ou bien le plus grand entier tel que $p_1 + p_2 + ... + p_i \leq p_{i+1} + ... + p_K$ si cela conduit à un écart plus faible). On attribue alors le préfixe 0 aux codes des caractères du premier groupe et 1 à ceux du second. On réitère la "dichotomie" par récurrence dans chacun des sous-groupes, et ce jusqu'à ce que chacun d'entre eux ne soit plus constitué que d'un seul caractère qui aura ainsi son code unique.

Les codes des caractères avec l'encodage de Shannon est donné en figure 5.

2 Codage par blocs

2.1 Codage par blocs et théorème de Shannon

1. Le théorème de Shannon appliqué à la source S énonce que :
— pour tout code univoque c, $R_S(c) \geq H(S)$
— il existe un code préfixe c vérifiant $H(S) + 1 \geq R_S(c) \geq H(S)$
où $R_S(c)$ correspond à la longueur moyenne du code c qui encode donc K lettres, et où $H(S)$ correspond à l'entropie de S.

2. L'entropie de S_1 se calcule de la manière suivante, en utilisant l'indépendance des X_i puis que les X_i sont distribués identiquement :

$$\begin{aligned}
H(S_1) &= \mathbb{E}[-log_2((X_1, ..., X_K))] \\
&= - \sum_{(x_1,...x_K)\in\mathcal{A}^K} p(x_1, ..., x_K) log_2(p(x_1, ...x_K)) \\
&= - \sum_{(x_1,...x_K)\in\mathcal{A}^K} p(x_1)...(x_K) \sum_{i=1}^{K} log_2(p(x_i)) \\
&= \sum_{i=1}^{K} \left[\prod_{1\leq j\leq K, j\neq i} \sum_{x_j\in\mathcal{A}} p(x_j) \right] \left[- \sum_{x_i\in\mathcal{A}} p(x_i) log_2(p(x_i)) \right] \\
&= \sum_{i=1}^{K} -\mathbb{E}[-log_2(X_i)] \\
&= \sum_{i=1}^{K} H(X_i) \\
&= KH(X_1)
\end{aligned}$$

Aussi le théorème de Shannon énonce que pour tout code préfixe c,

$$R_S(c) \leq H(S) + 1$$

$R_S(c)$ correspond à la longueur moyenne du code c de S qui encode donc K lettres, donc la longueur moyenne par lettre de la séquence X vaut $\frac{R_S(c)}{K}$. Ainsi, une borne supérieure pour celle-ci est donnée par :

$$\frac{R_S(c)}{K} \leq \frac{H(S)}{K} + \frac{1}{K} = H(X) + \frac{1}{K}$$

.

Deux commentaires.
i) Plus la taille des blocs K est importante, plus on peut espérer avoir un code court. On peut donc améliorer toute méthode d'encodage (vérifiant l'inégalité de Shannon $H(X) + 1 \geq R_X(c)$) en l'appliquant par blocs, aussi grands que possible.
ii)

$$H(X) \leq \frac{R_S(c)}{K} \leq H(X) + \frac{1}{K}$$

Quand K tend vers l'infini (supposons un texte infiniment long et $\mathcal{A}$ l'alphabet de tous ces textes infiniment longs), la longueur moyenne du code par lettre encodée tend donc vers $H(X)$

l'entropie de X, ce qui coïncide bien avec l'interprétation de l'entropie comme la quantité d'information moyenne délivrée par la source.

Aussi, on voit que, de part la borne inférieure du théorème de Shannon, on peut rendre optimale en terme de longueur moyenne par symbole toute méthode d'encodage préfixe vérifiant l'inégalité de Shannon ($H(X) + 1 \geq R_S(c)$) en l'appliquant par blocs et en faisant tendre la taille du bloc K vers l'infini (sous hypothèse du texte infiniment long).

2.2 Implémentation sur l'alphabet anglais

1.a Les codes pour les 26 paires de lettres de AA à ZZ obtenues avec l'algorithme de Huffman sont donnés en figure 6.

1.b L'entropie de la distribution sur les paires vaut 8.3514.

1.c La longueur moyenne par lettre du code c de Huffman vaut $R_S(c)/2$, où $R_S(c)$ correspond à la longueur moyenne du code de S sur les paires de lettres. On a : $R_S(c)/2 = 4.1909$.

1.d De même, la longueur moyenne par lettre du code c de Shannon vaut

$$-\frac{1}{2} \sum_{(x,y)\in\mathcal{A}^2} p(x,y)\lfloor log_2(p(x,y))\rfloor = 4.4268$$

.

1.e Avec le même argument que pour 1.3.e, un encodage qui attribuerait le même nombre de bits pour chaque paire de lettre aurait besoin de $\lfloor log_2(26^2)\rfloor + 1 = 10$ bits pour encoder une paire, soit toujours 5 bits pour encoder une lettre.

Les taux de compressions obtenus par codages de Huffman et de Shannon vis-à-vis d'un tel encodage sont respectivement : $\tau_{c_{Huffman}} = 16.18\%$ et $\tau_{c_{Shannon}} = 11.46\%$.

On résume les différentes valeurs obtenues avec blocs de 2 et sans blocs dans le tableau suivant :

Grandeur - Codage	par lettre	par paire
Entropie	4.1758	8.3514
Longueur du code de Huffman par lettre	4.2050	4.1909
Taux de compression Huffman	15.90%	16.18%
Longueur du code de Shannon par lettre	4.5806	4.4268
Taux de compression Shannon	8.39%	11.46%

On retrouve que l'entropie de la distribution sur les paires de lettres vaut le double de celle sur les lettres, comme on l'a montré par le calcul en 2.1.2 pour des variables i.i.d. On observe qu'en considérant des blocs de taille 2, on obtient bien des codes plus courts pour les deux méthodes Huffman et Shannon. Aussi, on remarque que l'augmentation du taux de compression avec les paires pour la méthode de Huffman n'est pas aussi importante que pour le code de Shannon, puisqu'on était déjà près avec Huffman simple de la borne inférieure de la longueur moyenne que constitue l'entropie.

2. Dans le code par blocs de taille 2,un certain nombre de codes vont par paires de codes de même longueur qui diffèrent seulement au dernier bit. En effet, puisque l'on a supposé que les variables des paires étaient i.i.d, "AB" et "BA" ont même probabilités d'apparition

$p('A') * p('B')$. Par conséquent, le tri initial des probabilités d'apparition fait apparaître $N^2 - N$ égalités (toutes les paires de lettres différentes) et N inégalités.

Supposons par exemple que "XZ" et "ZX" soient les caractères les moins fréquents de la liste, Huffman regroupe ces deux caractères comme les feuilles des deux branches 0 et 1 d'un noeud portant la somme de leurs fréquences. Par conséquent, leurs codes sont donc identiques à l'exception du dernier bit.

Cela ne se produit pas de cette façon pour toutes les paires. Supposons au contraire qu'à une certaine itération, "ZZ" soit le caractère le moins fréquent, et "XZ" le deuxième caractère moins fréquent, au même titre que "ZX". Alors l'algorithme de Huffman regrouperait "ZZ" et "XZ" d'abord, et "ZX" serait rattaché à l'itération suivante à un noeud intérieur ou un autre caractère. "XZ" et "ZX" auraient donc des séquences différant soit de longueur, soit de plus d'un bit strictement.

En observant les codes, on observe ainsi que "YZ" et "ZY" forment une telle paire, encodées respectivement par 0111011010110110 et 0111011010110111 (premier cas décrit). En revanche, ce n'est pas le cas pour "AB" et "BA" encodés respectivement par 1101000111 et 1101001100 (second cas)

3. La fonction huffman maintient une liste triée de probabilités. Pour une implémentation plus efficace, on peut envisager de ne pas trier les caractères selon leurs fréquences, mais de calculer à chaque itération les deux noeuds les moins fréquents. Ainsi, on opère un tri partiel sur des listes qui sont de tailles de plus en plus faibles.

On peut également préférer stocker ces fréquences dans un arbre binaire.

3 Entropie conditionnelle

1. On a, avec les convention $p(y|x)p(x) = 0$ si $p(x) = 0$ et toujours $0.log_2(0) = 0$:

$$\begin{aligned}
H(X,Y) &= -\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y) log_2(p(x,y)) \\
&= -\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y) log_2(p(y|x)p(x)) \\
&= -\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y) log_2(p(y|x)) - \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y) log_2(p(x)) \\
&= -\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y) log_2(p(y|x)) - \sum_{x\in\mathcal{X}} log_2(p(x)) \sum_{y\in\mathcal{Y}} p(x,y) \\
&= -\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y) log_2(p(y|x)) - \sum_{x\in\mathcal{X}} log_2(p(x)) p(x) \\
&= H(Y|X) + H(X)
\end{aligned}$$

2. La proposition (4.13) du cours (immédiate avec formule des probabilités conditionnelles) donne :

$$I(X,Y) + H(X,Y) = H(X) + H(Y)$$

Par conséquent, on obtient en soustrayant l'égalité de la question 3.1.,

$$I(X,Y) = H(Y) - H(Y|X)$$

3.

— L'information mutuelle $I(X,Y)$ peut s'écrire comme une divergence de Kullback-Leibler entre deux distributions sur $X \times Y$:

$$I(X,Y) = \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y) log_2 \frac{p(x,y)}{p(x)p(y)} = D(p_{X,Y}||p_X p_Y)$$

La proposition (4.9) du cours (qui réécrit la divergence comme l'espérance d'une certaine variable aléatoire et lui applique l'inégalité de Jensen) nous assure qu'une divergence est toujours positive.
Ainsi :

$$I(X,Y) \geq 0 \Leftrightarrow H(Y|X) \leq H(Y)$$

— De même, on peut écrire :

$$\begin{aligned}
H(Y|X,Z) - H(Y|X) &= - \sum_{(x,y,z)\in\mathcal{X}\times\mathcal{Y}\times\mathcal{Z}} p(x,y,z) log_2(p(y|x,z)) + \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} log_2(p(y|x))p(x,y) \\
&= - \sum_{(x,y,z)\in\mathcal{X}\times\mathcal{Y}\times\mathcal{Z}} p(x,y,z)[log_2(p(y|x,z)) - log_2(p(y|x))] \\
&= - \sum_{(x,y,z)\in\mathcal{X}\times\mathcal{Y}\times\mathcal{Z}} p(x,y,z) log_2 \frac{p(y|x,z)}{p(y|x)} \\
&= - \sum_{(x,y,z)\in\mathcal{X}\times\mathcal{Y}\times\mathcal{Z}} p(x,y,z) log_2 \frac{p(x,y,z)}{p(x,z)p(y|x)}
\end{aligned}$$

En vérifiant que $p_{X,Z}p_{Y|X}$ est bien une distribution de probabilités sur $X \times Y \times Z$:

$$\sum_{(x,y,z)\in\mathcal{X}\times\mathcal{Y}\times\mathcal{Z}} p(x,z)p(y|x) = \sum_{(x,z)\in\mathcal{X}\times\mathcal{Z}} p(x,z) \sum_{y\in\mathcal{Y}} p(y|x) = \sum_{(x,z)\in\mathcal{X}\times\mathcal{Z}} p(x,z) = 1$$

on peut donc écrire $H(Y|X,Z) - H(Y|X)$ comme l'opposé d'une divergence entre deux distributions sur $X \times Y \times Z$:

$$H(Y|X,Z) - H(Y|X) = -D(p_{X,Y,Z}||p_{X,Z}p_{Y|X})$$

Grâce à la proposition (4.9) qui assure la positivité de la divergence, on a bien :

$$H(Y|X,Z) \leq H(Y|X)$$

4 Compression pour une source stationnaire ergodique

1. Pour une source stationnaire ergodique on a :

$$\begin{aligned}
h_c(n+1) &= H(X_{n+1}|X_n,...,X_1) \\
&\leq H(X_{n+1}|X_n,...,X_2) && \text{(Q 3.3)} \\
&= H(X_{n+1},...,X_2) - H(X_n,...,X_2) && \text{(Q 3.1)} \\
&= H(X_n,...,X_1) - H(X_{n-1},...,X_1) && \text{(X stationnaire)} \\
&= H(X_n|X_{n-1},...,X_1) = h_c(n) && \text{(Q 3.1)}
\end{aligned}$$

D'où la décroissance de la suite h_c.
De plus, on peut montrer qu'une entropie conditionnelle est également positive :

$$\begin{aligned} H(Y|X) &= -\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y)log_2(p(y|x)) = -\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x)p(y|x)log_2(p(y|x)) \\ &= -\sum_{x\in\mathcal{X}} p(x)\sum_{y\in\mathcal{Y}} p(y|x)log_2(p(y|x)) = \sum_{x\in\mathcal{X}} p(x)H(Y|X=x) \end{aligned}$$

L'entropie d'une variable aléatoire étant positive par la proposition (4.6) (en tant qu'espérance d'une variable aléatoire positive), $H(Y|X=x)$ est positive pour tout x ce qui achève la preuve.
Ainsi, on a :

$$\forall n \quad h_c(n) \geq 0$$

Par conséquent, h_c est donc une suite décroissante minorée donc elle converge, vers une limite qu'on nomme H_∞

2. On a en utilisant la question 3.1 :

$$\begin{aligned} H(X_n,...,X_1) &= H(X_n|X_{n-1},...,X_1) + H(X_{n-1},...,X_1) \\ &= h_c(n) + H(X_{n-1},...,X_1) \end{aligned}$$

Par récurrence immédiate,

$$H(X_n,...,X_1) = \sum_{k=1}^{n} h_c(k)$$

Et ainsi :

$$h(n) = \frac{1}{n}H(X_n,...,X_1) = \frac{1}{n}\sum_{k=1}^{n} h_c(k)$$

Par 4.1, comme $h_c(k) \underset{k\to+\infty}{\longrightarrow} H_\infty$, on a, par comparaison des sommes partielles avec $H_\infty \neq 0$ terme général d'une série divergente,

$$\frac{1}{n}\sum_{k=1}^{n} h_c(k) \underset{n\to+\infty}{\sim} \frac{n}{n}H_\infty = H_\infty$$

Donc : $h(n) \underset{n\to+\infty}{\longrightarrow} H_\infty$

3. Soit $k \in [|1,n|]$. On a :

$$\begin{aligned} H(X_n,...,X_1) &= H(X_n|X_{n-1},...,X_1) + H(X_{n-1},...,X_1) && \text{(Q 3.1)} \\ &\leq H(X_n|X_{n-1},...,X_1) + H(X_{n-1},...,X_{n-k+1}) && \text{(Q 3.3)} \\ &= H(X_n|X_{n-1},...,X_1) + H(X_{k-1},...,X_1) && \text{(X stationnaire)} \end{aligned}$$

On a donc montré :

$$\forall k \in [|1,n|], \qquad H(X_n,...,X_1) \leq H(X_n|X_{n-1},...,X_1) + H(X_{k-1},...,X_1)$$

4. Avec la question 4.3, on a :

$$\forall k \in [|1, n|], \quad nh(n) \leq (n-1)h(n-1) + h_c(k)$$

En sommant, on obtient :

$$n^2 h(n) \leq n(n-1)h(n-1) + \sum_{k=1}^{n} h_c(k)$$

Mais $nh(n) = \sum_{k=1}^{n} h_c(k)$ (cf. Q 4.2) d'où :

$$n^2 h(n) \leq n(n-1)h(n-1) + nh(n)$$

Soit :

$$n(n-1)nh(n) \leq n(n-1)h(n-1) \Leftrightarrow h(n) \leq h(n-1)$$

D'où la décroissance de la suite $h(n)$ qui correspond à l'entropie moyenne par symbole pour une source stationnaire. Cela est très pertinent pour le codage par blocs. En effet, si n est la taille du blocs, le théorème de Shannon donne l'existence d'un code préfixe c dont la longueur moyenne vérifie :

$$H(X_1, ..., X_n) \leq R_S(c) \leq H(X_1, ..., X_n) + 1$$

et dont la longueur moyenne par lettre vérifie donc :

$$h(n) \leq \frac{R_S(c)}{n} \leq h(n) + \frac{1}{n}$$

Ainsi, on ne peut que diminuer la longueur moyenne du code optimal pour une source stationnaire en augmentant la taille du bloc.

5. Quand la taille du bloc n tend vers l'infini (vers la taille du texte pour un texte infiniment long), la longueur moyenne par lettre du code converge donc vers la limite de $h(n)$ quand n tend vers l'infini, c'est-à-dire H_∞ le taux entropique. Et on ne peut pas faire mieux en terme de longueur, de part la borne inférieure donnée par le théorème de Shannon et de part la décroissance de $h(n)$.

Toutefois, on ne peut augmenter à volonté la taille des blocs. En pratique,

— cela suppose de connaître la probabilités de toutes les N^K combinaisons de lettres possibles (si N est la taille de l'alphabet $\mathcal{A}$ et K la taille du bloc), ce qui est de plus en plus difficile et coûteux quand K devient grand : cela nécessite en amont de lire de très longs textes pour avoir des estimations précises (dans la limite improbable où $K = N$, il faut connaître la probabilité de tous les N^N textes possibles).

— le codage et le décodage nécessitent respectivement de construire et de conserver sous la main un arbre où figurent tous les chemins depuis la racine vers toutes les feuilles qui sont les N^K combinaisons de lettres possibles, ce qui peut vite poser de sérieux problèmes de mémoire !

6.b Notant $l(w)$ la longueur du mot w et $c(X)$ est le code de X, l'ergodicité nous assure que :

$$\frac{1}{n} \sum_{i=1}^{n} l(c(X_i)) \xrightarrow[n \to +\infty]{p.s.} \mathbb{E}[l(c(X_1))] = R_{X_1}(c)$$

Autrement dit, on peut approcher la longueur moyenne du code d'un texte en faisant la moyenne des longueurs des codes d'un grand nombre de textes générés par la source. L'ergodicité est donc importante pour s'assurer a priori que la compression fonctionnera.

7.a Si la séquence de symboles suit une chaîne de Markov d'ordre 1, alors, par propriété de Markov,

$$\begin{aligned} h_c(n) = H(X_n|X_{n-1},...,X_1) &= - \sum_{(x_1,...,x_n)\in\mathcal{X}^n} p(x_1,...,x_n) log_2 p(x_n|x_{n-1},...,x_1) \\ &= - \sum_{(x_1,...,x_n)\in\mathcal{X}^n} p(x_1,...,x_n) log_2 p(x_n|x_{n-1}) \\ &= - \sum_{(x_{n-1},x_n)\in\mathcal{X}^2} p(x_{n-1},x_n) log_2 p(x_n|x_{n-1}) \\ &= H(X_2|X_1) \end{aligned}$$

Si on appelle μ la distribution de X_1 (loi initiale) et Q la matrice de transition de la chaîne de Markov, on a donc :

$$H_\infty = \lim_{n\to\infty} h(n) = H(X_2|X_1) = - \sum_{(x,y)\in\mathcal{X}^2} \mu(x) Q(x,y) log_2 Q(x,y)$$

7.b Sous l'hypothèse de Markov énoncée en 7.a., on peut compresser la source avec des codes de Huffman ou de Shannon en considérant des blocs de taille 2 et en prenant pour probabilité du bloc $x_i x_j$: $p(x_i x_j) = Q(x_i, x_j) = P(X_2 = x_j | X_1 = x_i)$ (au lieu de $p(x_i)p(x_j) = \mu(x_i)\mu(x_j)$ dans le cas de l'indépendance vue précédemment)

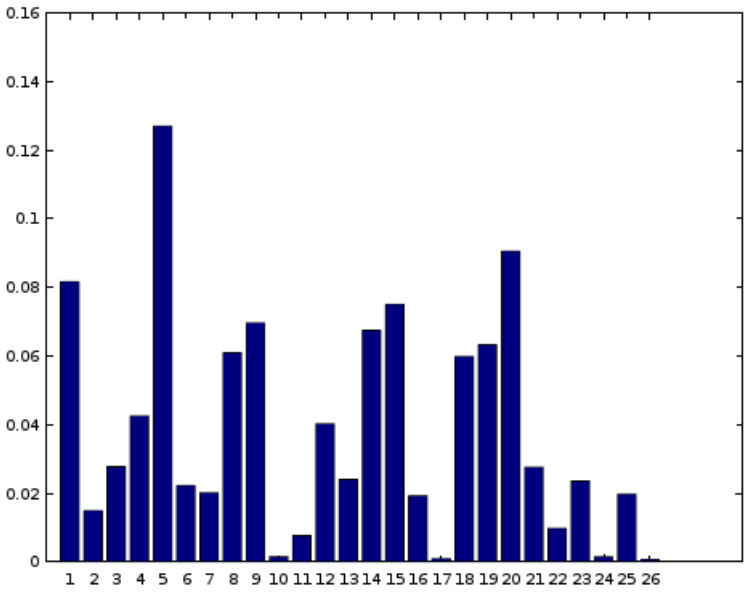


FIGURE 1 – Distribution des fréquences d'apparitions des caractères en langue anglaise

.

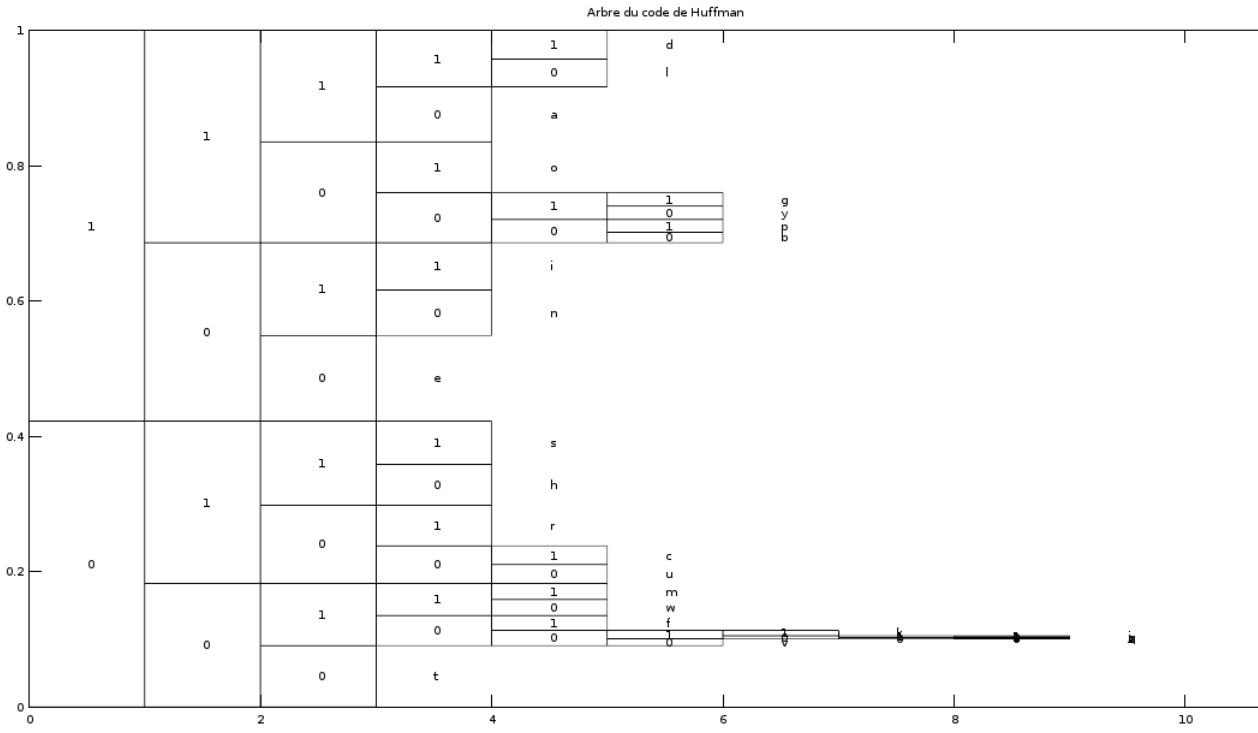
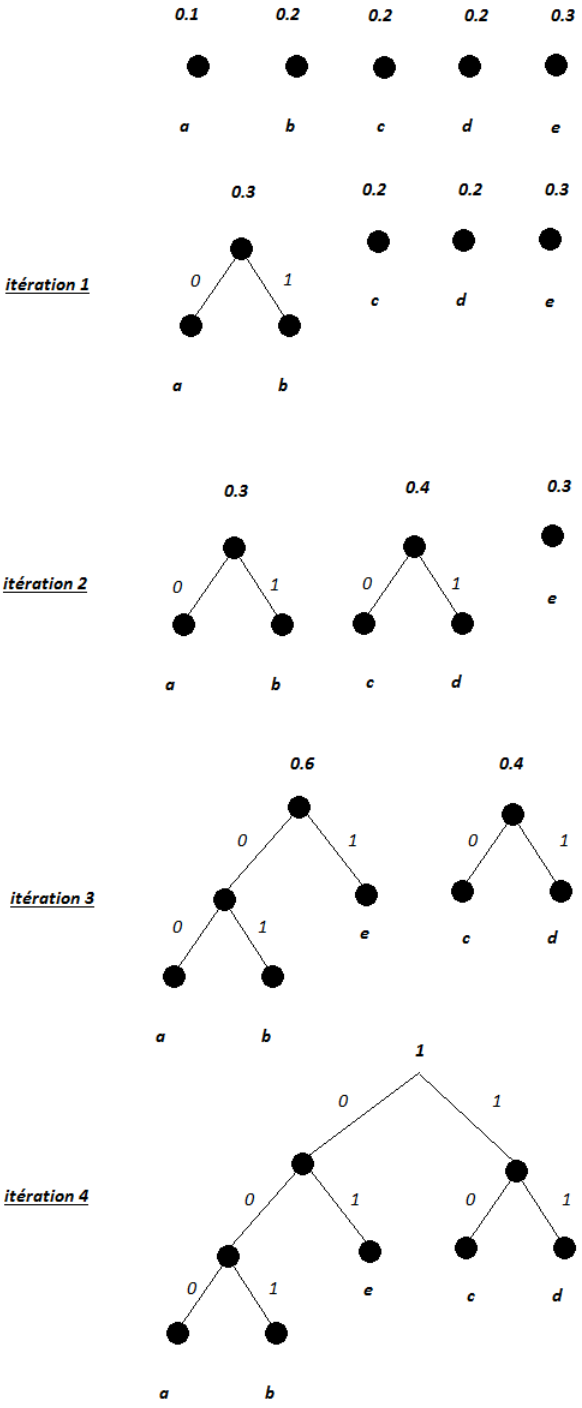


FIGURE 2 – Arbre associé au code de Huffman de l'alphabet anglais

.



Caractère	Fréquence	Code
a	0.1	000
b	0.2	001
c	0.2	10
d	0.2	11
e	0.25	01

Le code de Huffman est optimal au sens où il n'existe pas de code univoque dont la longueur moyenne est strictement inférieure.

Construction du code de Huffman, et représentation de l'arbre associé à chaque itération, sur un alphabet très simple

FIGURE 3 – Sur exemple, construction pas à pas du code de Huffman et de l'arbre associé pour un alphabet très simple de 5 caractères

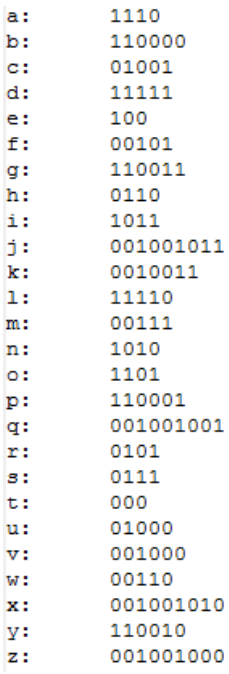


FIGURE 4 – Codes de Huffman des lettres de A à Z

.

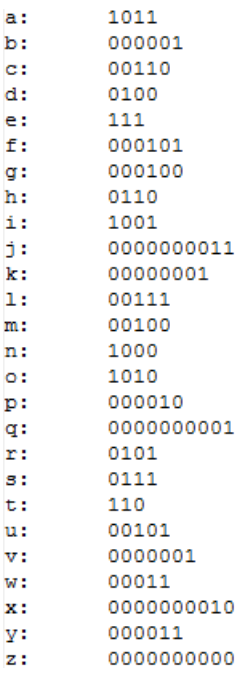


FIGURE 5 – Codes de Shannon des lettres de A à Z

.

aa:	0100111	yp:	00110011010
ab:	1101000111	yq:	1101000110101001
ac:	101111111	yr:	1100100001
ad:	01011011	ys:	1101110011
ae:	1110101	yt:	011001110
af:	011010111	yu:	10110011100
ag:	010010110	yv:	001110000010
ah:	11011001	yw:	01110110111
ai:	0000101	yx:	1000001101100000
aj:	1001000100111	yy:	00111010011
ak:	11011110010	yz:	01110110101101110
al:	01001000	za:	10001111000110
am:	100100000	zb:	0000000001010110
an:	11111100	zc:	1111101101001000
ao:	0011010	zd:	100100010110111
ap:	001110101	ze:	0010100010101
aq:	11011110001000	zf:	10101010000011100
ar:	11010100	zg:	10000011011011100
as:	11100111	zh:	00010010011111
at:	0111001	zi:	01010000101010
au:	101111100	zj:	110001111100101101100001
av:	0100000110	zk:	010111010111111000
aw:	100001001	zl:	100000110110011
ax:	1001000100100	zm:	11000111111001000
ay:	010001000	zn:	01000111010100
az:	10001111000101	zo:	01011101011110
ba:	1101001100	zp:	01110110101010000
bb:	011001111100	zq:	100000110110001010100
bc:	01001101001	zr:	00000000010111
bd:	11011111100	zs:	00111000000010
be:	011111011	zt:	10101010000110
bf:	111100011110	zu:	11011110001111110
bg:	110100011000	zv:	011101101010100010
bh:	0110101101	zw:	10111111011000010
bi:	1010010000	zx:	100000110110001011101
bj:	0001111110100000	zy:	01110110101101111
bk:	0110110011111	zz:	000000000101011100000

FIGURE 6 – Aperçu des codes de Huffman des paires de lettres de AA à ZZ

.