

Information, Calcul et Communication

Module 2 : Information et Communication

Information, Calcul et Communication

Leçon 2.3 : Compression de données (1ère partie)

O. Lévêque – Faculté Informatique et Communications

Introduction

Lors des deux leçons précédentes, nous avons répondu aux questions suivantes :

- ▶ Comment représenter / capter la réalité physique avec des bits ?
filtrage, échantillonnage, quantification
- ▶ Comment restituer cette réalité à partir de bits ?
reconstruction, th. d'échantillonnage, importance du filtrage

Lors des deux leçons à venir, nous allons tenter de répondre aux questions suivantes :

- ▶ Comment mesurer la quantité d'information présente dans des données ?
- ▶ Comment stocker des données en utilisant le moins d'espace possible (sans ou avec pertes) ?

Introduction

Pourquoi donc vouloir compresser des données ?

- ▶ pour réduire l'espace mémoire utilisé lors du stockage de données
- ▶ pour réduire le temps de transmission et les problèmes de congestion lors de la transmission des données

Cependant, avec les progrès de la technique, ne suffit-il pas d'attendre un peu pour avoir de meilleures performances ?

Certes, mais on veut toujours exploiter un système au maximum de ses capacités !

Introduction

Quels types de données peuvent être comprimées ?

- ▶ le langage
- ▶ le son (voix/musique), les images (photos), les vidéos
- ▶ tous types de données numériques

Le principe de base derrière la compression de données est la **suppression de la redondance** présente dans ces données.

Introduction

Le français est plein de redondance ! Pour preuve, ce texte :

Selon une étude de l'Université de Cambridge, l'ordre des lettres dans un mot n'a pas d'importance, la seule chose importante est que la première et la dernière soient à la bonne place. Le reste peut être dans un désordre total et vous pouvez toujours lire sans problème. C'est parce que le cerveau humain ne lit pas chaque lettre elle-même, mais le mot comme un tout.

Pourquoi donc tant de redondance dans la langue française ?

- ▶ pour pouvoir se comprendre, tout simplement !
(« capacité du canal »)
- ▶ pour être capable de lire un texte même s'il contient des fautes d'orthographe...



On distingue deux types de compression :

- ▶ **la compression sans perte**, lorsqu'on désire retrouver l'intégralité des données stockées sous forme comprimée.

Exemples : billets pour un concert, déclaration d'impôts, bulletins de vote, articles scientifiques

- ▶ **la compression avec pertes**, lorsqu'on n'est pas tant à cheval que ça sur les détails et qu'on s'autorise un peu de *distorsion*.

Exemples : émissions podcastées et morceaux de musique en format mp3, partage de photos sur le web, vidéos youtube...

Plan

Plan détaillé des deux leçons à venir :

Aujourd'hui :

- ▶ notion d'entropie
- ▶ compression sans perte
- ▶ algorithme de Shannon-Fano

La semaine prochaine :

- ▶ algorithme de Shannon-Fano (bis)
- ▶ analyse de performance – théorème de Shannon
- ▶ compression optimale : code de Huffman
- ▶ compression avec pertes

Entropie

Voici une séquence de 16 lettres :

A B C D E F G H I J K L M N O P

Jeu n° 1 : Vous devez deviner quelle lettre j'ai choisi au hasard en posant un *nombre minimum de questions*, auxquelles je ne peux répondre que par **oui** ou par **non**.

Solution : 4 questions sont nécessaires (cf. leçon 1.1 : algorithme de dichotomie). On dit que l'entropie de cette séquence est égale à **4**.

Remarquez que $16 = 2^4$, autrement dit : $4 = \log_2(16)$

Entropie

Voici une autre séquence de 16 lettres (sans compter les espaces) :

I L F A I T B E A U A I B I Z A

Jeu n° 2 : Le jeu est le même qu'avant !

Remarques :

- ▶ Je choisis la *position* de la lettre au hasard, de manière uniforme.
- ▶ Vous ne devez deviner que la lettre elle-même, pas sa position.

Combien de questions binaires sont-elles nécessaires en moyenne pour deviner la lettre ?



I L F A I T B E A U A I B I Z A

Solution : **classer** les lettres **dans l'ordre décroissant du nombre d'apparitions** dans la séquence :

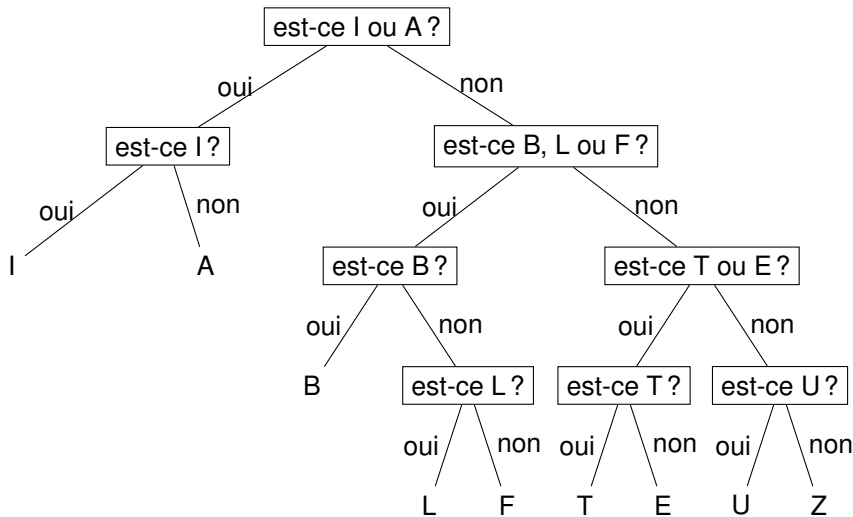
lettre	I	A	B	L	F	T	E	U	Z
nb d'apparitions	4	4	2	1	1	1	1	1	1

L'idée est la même que précédemment : on **sépare** l'ensemble des lettres **en deux parties égales** en termes de *nombre d'apparitions*. Sur l'exemple cela donne :

Question n° 1 : est-ce que la lettre est un I ou un A ?

- ▶ Si la réponse est **oui** : **question n° 2** : est-ce que la lettre est un I ?
- ▶ Si la réponse est **non** : **question n° 2** : est-ce que la lettre est un B, un L ou un F ? etc.

Arbre des questions



Entropie

I L F A I T B E A U A I B I Z A

lettre	I	A	B	L	F	T	E	U	Z
nb d'apparitions	4	4	2	1	1	1	1	1	1
nb de questions	2	2	3	4	4	4	4	4	4

Nombre de questions à poser en moyenne :

$$= 2 \times \frac{4}{16} \times 2 + 1 \times \frac{2}{16} \times 3 + 6 \times \frac{1}{16} \times 4 = \frac{16 + 6 + 24}{16} = \frac{46}{16} = 2.875$$

On dit que l'entropie de cette séquence est égale à 2.875.

Entropie

Voici encore une autre séquence de 16 lettres :

A A A A A A A A A A A A A A A A

Jeu n° 3 : Le jeu est encore le même qu'avant.

Cette fois-ci, *aucune* question n'est nécessaire pour deviner la lettre choisie !

On dit que l'entropie de cette séquence est égale à 0.

Entropie

I L F A I T B E A U A I B I Z A

lettre	I	A	B	L	F	T	E	U	Z
nb d'apparitions	4	4	2	1	1	1	1	1	1
nb de questions	2	2	3	4	4	4	4	4	4

Remarques :

- ▶ Pour deviner une lettre qui apparaît 1 fois sur 16, on a besoin de 4 questions. $4 = \log_2(16)$
- ▶ Pour deviner une lettre qui apparaît 2 fois sur 16 (c.-à-d. $1/8$), on a besoin de 3 questions. $3 = \log_2(8)$
- ▶ Pour deviner une lettre qui apparaît 4 fois sur 16 (c.-à-d. $1/4$), on a besoin de 2 questions. $2 = \log_2(4)$
- ▶ En résumé, pour deviner une lettre qui apparaît avec une probabilité p , on a besoin de $\log_2\left(\frac{1}{p}\right)$ questions.

Entropie – Définitions



Définition approximative (on y reviendra en fin de semaine prochaine) :

L'entropie est égale au **nombre moyen de questions** nécessaires pour deviner une lettre choisie au hasard dans une séquence.

Définition formelle :

Soit X une distribution de probabilités p_j
(par construction $0 \leq p_j \leq 1$ pour tout j , et $p_1 + \dots + p_n = 1$).

L'entropie de X est définie par :

$$\begin{aligned} H(X) &= p_1 \log_2 \left(\frac{1}{p_1} \right) + \dots + p_n \log_2 \left(\frac{1}{p_n} \right) \\ &= - \left(p_1 \log_2 (p_1) + \dots + p_n \log_2 (p_n) \right) \end{aligned}$$

Note : par prolongement par continuité, $p_j \log_2 (p_j) = 0$ pour $p_j = 0$.

Entropie – Définitions plus générales

- ▶ La formule précédente reste valable même si les p_j ne sont pas estimés sur une (petite) séquence de caractères, mais proviennent d'une estimation sur un texte plus grand ou même de notre connaissance du procédé qui a généré les événements observés (lettres, mots, ...).

Tout ce qui compte, ce sont simplement les probabilités p_j de chaque lettre (peu importe d'où elles viennent).

- ▶ En réalité, l'**entropie** calculée ici est simplement celle du « jeu » consistant à tirer une lettre du « mot » X au hasard.

Pour de vraies séquences, en tant que telles, la définition de l'entropie est bien plus complexe et intègre toutes les dépendances entre sous-séquences de lettres, c.-à-d. les probabilités de toutes les sous-séquences de n lettres.

Mais ceci nécessite des outils mathématiques plus avancés... (probabilités conditionnelles et limites).

Entropie

- ▶ Origine en physique (Boltzmann, 1872) :
 - ▶ L'entropie mesure le **désordre** dans un système physique.
 - ▶ Ludwig Eduard Boltzmann (1844-1905)
 - ▶ ardent défenseur de l'existence des atomes
 - ▶ père de la physique statistique
-
- ▶ Théorie de l'information (Shannon, 1948) :
 - ▶ L'entropie mesure la « **quantité d'information** » contenue dans un signal.
 - ▶ Claude Edwood Shannon (1916-2001)
 - ▶ mathématicien, ingénieur électricien, cryptologue,
 - ▶ père de la théorie de l'information



Entropie

Interprétation :

A B C D E F G H I J K L M N O P $H(X) = 4 \text{ bit}$

I L F A I T B E A U A I B I Z A $H(X) = 2.875 \text{ bit}$

A A A A A A A A A A A A A A A A $H(X) = 0$

Plus il y a de lettres différentes, plus il y a de désordre,
plus il y a de *nouveauté*,
plus il y a d'« *information* » dans le message.

Plus il y a de lettres semblables, moins il y a de désordre,
plus il y a de *redondance* et donc moins d'« *information* » dans le message.

Redondance : $R(X) = 1 - \frac{H(X)}{\log_2(n)}$ (n : nombre de lettres différentes dans X)



Une remarque, et quelques propriétés de l'entropie :

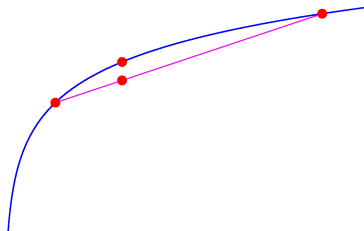
$$H(X) = p_1 \log_2 \left(\frac{1}{p_1} \right) + \dots + p_n \log_2 \left(\frac{1}{p_n} \right)$$

- ▶ Pour une probabilité d'apparition $0 < p \leq 1$ donnée, $\log_2 \left(\frac{1}{p} \right) \geq 0$, mais ça n'est pas forcément un nombre entier (p. ex. si $p = 1/3$).
- ▶ $H(X) \geq 0$
et $H(X) = 0$ si et seulement si toutes les lettres sont les mêmes.
- ▶ Si n est la taille de l'alphabet utilisé, $H(X) \leq \log_2(n)$
et $H(X) = \log_2(n)$ si et seulement si toutes les lettres sont différentes (et donc *équiprobables*).

Démonstrations :

- ▶ $H(X) \geq 0$: Remarquer simplement que $0 \leq p_j \leq 1$ implique que $p_j \log_2 \left(\frac{1}{p_j} \right) \geq 0$, et donc que $H(X) \geq 0$.

- ▶ $H(X) \leq \log_2(n)$: Remarquer que la fonction $f(x) = \log_2(x)$ est concave pour $x \geq 0$:



- ▶ En particulier, cela implique que :

$$\alpha \log_2(x_1) + (1 - \alpha) \log_2(x_2) \leq \log_2(\alpha x_1 + (1 - \alpha) x_2)$$

pour tout $x_1, x_2 > 0$ et $0 \leq \alpha \leq 1$

- ▶ Autrement dit, si $0 \leq p_1, p_2 \leq 1$ et $p_1 + p_2 = 1$, alors :

$$p_1 \log_2(x_1) + p_2 \log_2(x_2) \leq \log_2(p_1 x_1 + p_2 x_2) \quad \text{pour tout } x_1, x_2 > 0$$

Démonstrations : (suite)

- ▶ Plus généralement encore, si $0 \leq p_j \leq 1$ et $p_1 + \dots + p_n = 1$, alors

$$p_1 \log_2(x_1) + \dots + p_n \log_2(x_n) \leq \log_2(p_1 x_1 + \dots + p_n x_n)$$

pour tout $x_1, \dots, x_n > 0$.

- ▶ En appliquant cette inégalité avec $x_j = \frac{1}{p_j}$, on obtient finalement :

$$\begin{aligned} H(X) &= p_1 \log_2\left(\frac{1}{p_1}\right) + \dots + p_n \log_2\left(\frac{1}{p_n}\right) \\ &\leq \log_2\left(\frac{p_1}{p_1} + \dots + \frac{p_n}{p_n}\right) = \log_2(1 + \dots + 1) = \log_2(n) \end{aligned}$$

QED

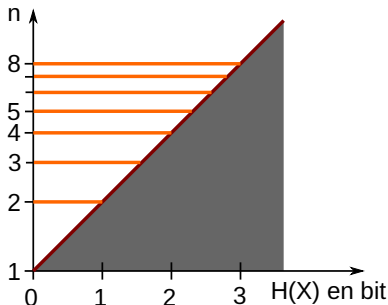
Propriétés de l'entropie



$$0 \leq H(X) \leq \log_2(n)$$

Il y a donc deux façons d'augmenter l'entropie :

- ▶ rendre les probabilités plus équilibrées (équiprobables)
- ▶ augmenter n (nombre de valeurs possibles) ; mais...



Entropie

Mais à quoi tout cela peut-il bien servir ?

Compression sans perte

Comme on l'a vu dans l'introduction, la compression de données permet :

- ▶ de réduire l'espace mémoire utilisé lors du stockage de données
- ▶ de réduire le temps de transmission et les problèmes de congestion lors de la transmission des données

Le principe de base derrière la compression de données est la suppression de la **redondance** contenue dans ces données ; les lettres ou les mots qui reviennent souvent dans un message sont abrégés.

Lorsqu'on parle de compression **sans perte**, on demande que le message d'origine puisse être récupéré dans son intégralité après le processus de compression.

Compression sans perte

Exemples d'algorithmes de compression sans perte :

- ▶ **Langage SMS** : « slt », « tqt », « mdr », etc ; les mots qui reviennent souvent sont réduits à de courtes séquences de lettres ; mais *normalement* on sait toujours de façon non ambiguë de quel mot d'origine il s'agit
- ▶ **Code Morse** : « a » = .- « e » = . « s » = ... « t » = -
tandis que « x » = -..- « z » = - -..

Le concept d'entropie permet de comprimer des données de manière *systématique et optimale*.

Compression sans perte

Revenons sur la plage à Ibiza, et supposons que vous vouliez envoyer ce message par SMS à un ami :

IL FAIT BEAU A IBIZA

Le but du jeu est maintenant de *minimiser le nombre de bits* nécessaires pour représenter ce message.

Remarque importante :

La personne qui reçoit votre message doit être capable de le lire !

- ✋ *Avant* l'envoi du message, l'expéditeur et le destinataire se sont mis d'accord sur un code commun.

Exemple : A='001', B='101', etc.

Voyons comment construire un tel code, qui soit efficace.

Compression sans perte

I L F A I T B E A U A I B I Z A

1^{re} idée : Utiliser le code ASCII étendu (cf. leçon 1.4) : chaque lettre est représentée par 8 bits ; $16 \times 8 = 128$ bits sont donc nécessaires pour représenter ce message.

2^e idée : Remarquer que le message à représenter n'est composé que de 9 lettres différentes : on n'a donc besoin que de $\lceil \log_2(9) \rceil = 4$ bits par lettre ; et donc $16 \times 4 = 64$ bits au total.

3^e idée : Remarquer que certaines lettres sont plus fréquentes que d'autres... Comment en tirer partie ?

Algorithme de Shannon-Fano



- ▶ Robert Fano (1917–2016)
- ▶ professeur au MIT
- ▶ autre pionnier de la théorie de l'information



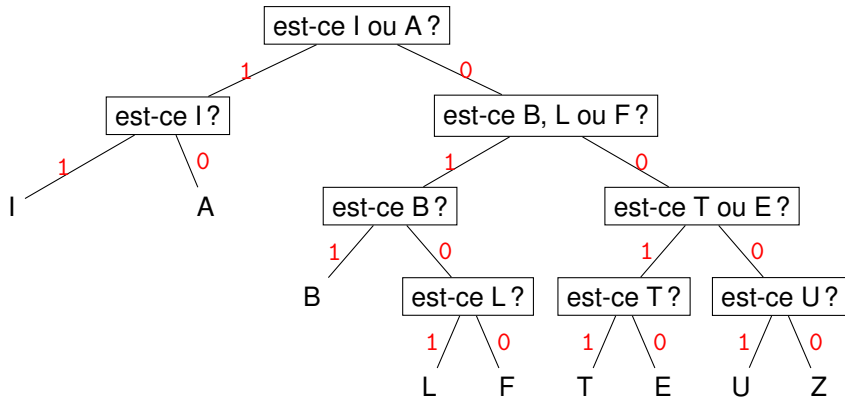
Pour attribuer une séquence de bits (un « mot de code ») à chaque lettre, nous suivons les deux règles ci-dessous :

- ▶ **Règle n° 1** : le nombre de bits attribués à chaque lettre est égal au nombre de questions nécessaires pour la deviner.
(voir slide 9/32)
- ▶ **Règle n° 2** : les bits 0 ou 1 sont attribués en fonction des réponses obtenues aux questions. Plus précisément, le bit n° j est égal à 1 ou 0 selon que la réponse à la j^{e} question était oui ou non.

Algorithme de Shannon-Fano

IL FAIT BEAU A IBIZA

lettre	I	A	B	L	F	T	E	U	Z
nb d'apparitions	4	4	2	1	1	1	1	1	1



Algorithme de Shannon-Fano

Ce qui donne le code suivant :

lettre	I	A	B	L	F	T	E	U	Z
nb de questions	2	2	3	4	4	4	4	4	4
mot de code	11	10	011	0101	0100	0011	0010	0001	0000

Le message « IL FAIT BEAU A IBIZA » s'écrit ainsi :

11 0101 0100 10 ...

Observation n° 1 : Dans le code, aucun mot de code n'est le préfixe d'un autre. Donc le message reçu est parfaitement décodable, et ceci au fur et à mesure.

Algorithme de Shannon-Fano

lettre	I	A	B	L	F	T	E	U	Z
nb d'apparitions	4	4	2	1	1	1	1	1	1
taille du code	2	2	3	4	4	4	4	4	4
mot de code	11	10	011	0101	0100	0011	0010	0001	0000

Observation n° 2 : Le nombre de bits utilisés est égal à :

$$8 \times 2 + 2 \times 3 + 6 \times 4 = 16 + 6 + 24 = 46$$

Par rapport à la représentation « brute » qui nécessite 64 bits, on économise donc environ 25% de bits.

Vu que le message a 16 lettres, le nombre moyen de bits utilisés par lettre est égal à $46/16 = 2.875 = H(X)$ = entropie du message !

Nous montrerons la semaine prochaine que l'entropie est une **borne inférieure** au nombre moyen de bits par lettre dont on a besoin : aucun autre algorithme de codage non ambigu et sans perte ne permet de descendre plus bas que ça.

En pratique... (1/3)

Note : pour décoder, il faut évidemment avoir le même arbre (il faut s'être mis d'accord des deux cotés au préalable : on ne « unzip » pas un fichier « .rar »).

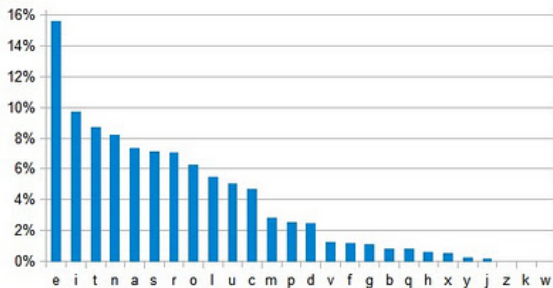
Question : Doit-on toujours recréer un code « depuis le début » pour représenter un message ?

Réponse : Non ! On peut simplement utiliser un code basé sur les probabilités d'apparition des lettres dans la langue française :

On applique l'algorithme précédent à un très gros texte de référence (plusieurs milliers de romans) pour trouver l'arbre que l'on utilise ensuite dans les algorithmes de codage et de décodage de toute séquence.

En pratique... (2/3)

En pratique, on applique l'algorithme de calcul des codes à un très gros texte de référence (plusieurs milliers de romans) pour trouver l'arbre que l'on utilise ensuite dans les algorithmes de codage et de décodage de toute séquence.



(tiré de la déclaration universelle des droits de l'homme)

En pratique... (3/3)

En pratique, on applique l'algorithme de calcul des codes à un très gros texte de référence (plusieurs milliers de romans) pour trouver l'arbre que l'on utilise ensuite dans les algorithmes de codage et de décodage de toute séquence.

Avec cet algorithme, on obtient les performances suivantes :

- ▶ Dans un texte :
 - ▶ 15-25% de réduction avec les lettres, comme vu précédemment
 - ▶ jusqu'à 75% de réduction si on utilise les mots à la place des lettres !
- ▶ Dans un fichier de données : 25-30% de réduction (p. ex. : fichiers archivés .zip pour l'envoi d'emails)

Conclusion temporaire

- ▶ L'**entropie** mesure la « *quantité d'information* » présente dans un message : **nombre moyen de questions** nécessaires pour deviner une lettre choisie au hasard dans ce message

Note : se généralise :

- ▶ à n'importe quelle distribution de probabilité
- ▶ à l'ensemble des sous-séquences (pas uniquement une seule lettre)
- ▶ Pour représenter un message par une séquence de bits, l'**algorithme de Shannon-Fano** exploite les nombres d'apparitions.

Dans un cas particulier étudié, il utilise en moyenne un nombre de bits par lettre égal à l'entropie du message.

La semaine prochaine :

- ▶ algorithme de Shannon-Fano (bis)
- ▶ analyse de performance – théorème de Shannon
- ▶ compression optimale : code de Huffman
- ▶ compression avec pertes