COMPRESSION PAR DICTIONNAIRES

1

1. Principes de la compression par dictionnaire
2. Dictionnaire Statique
3. Dictionnaire adaptatif
   - LZ77-LZSS
   - LZW : phase stationnaire et phase adaptative

COMPRESSION AVEC DICTIONNAIRE

2

Principe : groupe de symboles codé par index dans dictionnaire
$C : A^* \rightarrow \{0,1\}^*$

1. Dictionnaire statique
   Dictionnaire de référence, partagé par compresseur et décompresseur
2. Dictionnaire adaptatif
   Le dictionnaire est "défini" par le fichier à compresser : apprentissage dynamique des répétitions
   Exemple : Acronymes introduits lors de la première occurrence (CCC)
   Adaptativité inversible (fonction de décompression)

J. Ziv and A. Lempel *"A universal algorithm for data compression"* IEEE Transactions on Information Theory May 1977

DICTIONNAIRE STATIQUE

3

1. Codes postaux (corpus particulier – méthodes ad hoc)
2. Dictionnaire de référence : Petit Robert 1993
   Pages $< 2^{12}$, 2 Colonnes, Mots par colonnes $< 2^6$
   Algorithmes et scoubidous $\rightarrow 56/2/13//822/2/5//2055/1/5$
   25 caractères (200 bits) compressés en $3*(12+1+6) = 57$ bits
3. Dictionnaire par taille de mot : en francais taille des mots entre 1 et 25
   On connaît distribution des mots selon leur taille (9000 mots de 6 lettres) 9000 à comparer aux $26^6 > 100$ millions ($\sim 2^{30}$) de possibilités
   $\rightarrow$ codage sur 14 bits par mot (+ 5 bits pour taille) au lieu de 30 bits

Limitations et inconvénients : dictionnaire figé ; rendre compte des conjugaisons diverses, exceptions ....

DICTIONNAIRE ADAPTATIF LZ77-LZSS

4

LZ77-LZSS : Fenêtre fixe ou coulissante sur le fichier
Ex : AIDE-TOI-LE-CIEL-T-AIDERA 25car sur 8 bits = 200 bits
1A1I1D1E1-1T1O1I1-1L0(3,2)1C1I1E1L0(4,2)1-0(0,4)1R1A
20 bits + 17car sur 8 bits + 3couples(pos, $long$) sur 5+2 bits = 177 bits
- fenêtre de taille $2^k \Rightarrow k$ bits pour pos ;
- $long$ sur n bits $\Rightarrow$ on peut compresser séquences de 2^n car.

Difficultés :
- gestion et représentation de la fenêtre coulissante
- compression : rechercher, en toutes positions, la plus longue séquence de la fenêtre pour coder la suite du texte. Décompression pas de pb !
- complexité : $tailleFen * longMax$ si fenêtre structurée séquentiellement
- complexité : $\log_2(tailleFen) * longMax$ si fenêtre structurée en arbre (suffixe). On peut alors doubler taille fenêtre sans augmenter trop le temps de recherche.

LZ78- LZW

LZ78- LZW : Dictionnaire = Ensemble (Hachage-Arbre digital) de toutes les séquences déja rencontrées (potentiellement illimité).
Méthodes adaptatives : phase transitoire (gagne en compression par l'apprentissage) puis stationnaire (ne sert à rien de continuer à adapter).

Algorithme en 2 phases

1. Phase adaptative : construit un dictionnaire tout en commençant le codage
2. Phase stationnaire : le dictionnaire n'évolue plus, on l'utilise pour poursuivre le codage

Le compresseur et le décompresseur construisent le même dictionnaire !
Dictionnaire = table associative,
La fonction de hachage $h : A^+ \rightarrow [0,\ldots,m-1]$ *est connue par le compresseur et le décompresseur.*

PHASE STATIONNAIRE

Définition : Un ensemble F de mots est dit *préfixiel* ssi lorsque $f \in F$, alors tous les préfixes de f sont dans F

Le dictionnaire de LZW est un ensemble préfixiel F de séquences du texte T à coder. Chaque $f \in F$ est codé par son index (numéro) dans la table.

En phase stationnaire (Le dictionnaire est connu des 2 cotés)

– La compression consiste à déterminer le plus long préfixe de T qui est dans le dictionnaire F et transmettre son index dans F, et recommencer sur la suite de T.
– Pour la décompression il suffit de remplacer chaque index reçu par son contenu dans la table.

EXEMPLE : T = ...ababcbababaaaaaaaabbabaabca

index	*f*	*(père, lettre)*
1	a	
2	b	
3	c	
4	ab	1b
5	ba	2a
6	abc	4c
7	cb	3b
8	bab	5b
9	baba	8a
10	aa	1a
11	aaa	10a

T= ... ababcbababaaaaaaaabbabaabca

STRUCTURES DE DONNÉES

$T = ...ababcbababaaaaaaaabbabaabca$
$C(T)$= 4 6 9 5 12 10 2 9 6 1
Fonction de hachage (collisions résolues) :
$h(a) = 1, h(b) = 2, h(c) = 3, h(ab) = 4, h(ba) = 5, h(abc) = 6, h(cb) = 7,$
$h(bab) = 8, h(baba) = 9, h(aa) = 10, h(aaa) = 11$

1. A la compression,
   – pour reconnaître le plus long préfixe de T dans F : **Arbre digital**,
   – stocker dans la table la valeur du père (index) et la dernière lettre (gain de place)
2. A la décompression
   – la table est suffisante (pas besoin d'arbre digital)
   – décoder le contenu (père, lettre)

PHASE ADAPTATIVE

Au départ la table F contient les lettres de A.

1. Le compresseur
   - détermine le plus long préfixe f de T dans F
   - transmet l'index de f dans F
   - ajoute dans F la séquence fx (où x est le caractère suivant dans T)
   - recommence à lire T à partir de x
2. Le décompresseur, lit le premier numéro et renvoie la lettre, puis
   - Lit un numéro i et consulte la table à cet index
   - s'il y a un contenu, le décode en w_i, puis ajoute à la table (par hachage) une entrée formée de w_{i-1} (ou son index), suivi de la première lettre de w_i
   - sinon ajoute à cet endroit w_{i-1} (ou son index), suivi de **sa** première lettre. Cas de **chevauchement** : $w_{i-1} = bw$ et $w_i = bwb$.

EXEMPLE

Initialement la table contient les lettres : $h(a) = 1, h(b) = 2, h(c) = 3$

$T = ababcbababaaaaaaa$
$C(T)$= 1 2 4 3 5 8 1 10 11
Fonction de hachage (collisions résolues) :
$h(ab) = 4, h(ba) = 5, h(abc) = 6, h(cb) = 7,$
$h(bab) = 8, h(baba) = 9, h(aa) = 10, h(aaa) = 11$

- Compression
- Décompression

STATISTIQUE VS. DICTIONNAIRE

- Dictionnaire code les redondances de séquences

- apprentissage des répétitions $\neq$ méthodes statistiques d'ordre supérieur ordre infini, en ne stockant que ce qui est nécessaire.

- Exemple : fichier uniforme contenant 1000 fois le même caractère C
  - Huffman : C codé sur 1 bit $\Rightarrow$ 1000bits,
  - LZW : A chaque étape ajoute un préfixe contenant un C de plus : $1 + \ldots + k = 1000 \Rightarrow k \sim 45$adresses $\Rightarrow$ 45 * 7bits + 8bits $<$ 350 bits

STATISTIQUE VS. DICTIONNAIRE

A l'inverse il existe des cas où la compression statistique est mieux adaptée que LZW :

Fichier d'octets $\rightarrow$ 8 fichiers binaires (le premier formé de tous les premiers bits, le deuxième formé de tous les deuxièmes bits,. . ..).

Compression des 8 fichiers par Huffman OK, mais pas par LZW, car en découpant en 8 "plans" on a pu casser des régularités.

Succession de deux compressions :
- Huffman suivi de LZW : améliore souvent la compression
- LZW suivi de Huffman : NON car le hachage de LZW "cache" les fréquences.