L'informatique des entrepôts de données

Daniel Lemire

SEMAINE 6
Compression dans les bases de données

6.1. Présentation de la semaine

Nous avons déjà vu à la semaine précédente le codage par plage, et le rôle important que cette technique de compression joue dans la construction de certains index, comme les index bitmaps. Il existe cependant plusieurs techniques de compression qui sont importantes dans les entrepôts de données.

La compression procure plusieurs bénéfices. D'abord, elle permet de stocker plus d'information sur des disques. Mais ce n'est pas la fonction la plus importante en ce qui nous concerne. La compression permet aussi de passer plus de données du disque aux microprocesseurs. En effet, des données compressées se chargent souvent plus rapidement en mémoire. De plus, la compression permet de conserver plus de données en mémoire.

Nous verrons cette semaine plusieurs techniques de compression populaires.

6.2. Théorie de l'information

Étant donné un fichier, jusqu'à quel point pouvons-nous le compresser? Intuitivement, plus un document contient d'information, plus il sera difficile de le compresser. En effet, un fichier qui ne contiendrait qu'une seule lettre (`a`) répétée des milliers de fois ne contiendrait pas beaucoup d'information et serait donc facile à compresser (par codage par plage).

Exemple 1. *J'ai un texte de 3 mots « A », « B » et « C ». Je choisis d'abord de représenter le premier mot sous la forme binaire « 00 », le second sous la forme « 10 » et le dernier sous la forme « 11 ». La séquence « ABCA » devient alors « 00101100 » ou « 00-10-11-00 ». Il a donc fallu 8 bits pour stocker les 4 mots, donc une moyenne de 2 bits par mot. Shannon savait qu'il était possible d'être encore plus efficace. En effet, il a représenté le premier mot avec un seul bit : « 0 ». La séquence « ABCA » devient alors « 010110 » ou « 0-10-11-0 », donc 6 bits pour stocker 8 mots, ce qui représentait un gain substantiel ! Est il est possible de faire mieux?*

L'étude de cette mesure de la quantité d'information, en fonction de la compressibilité, fait l'objet d'une théorie générale, la théorie de l'information. On doit la théorie de l'information à Shannon [**?**]. Il fut le premier à proposer une mesure d'entropie pour l'information: c'est-à-dire une limite fondamentale à la compression des données.

A lire : `http://fr.wikipedia.org/wiki/Theorie_de_l'information`

Considérons une chaîne de n caractères où le caractère x apparaît $f(x)$ fois. La probabilité de sélectionner ce caractère, au hasard, est donc de $p(x) = f(x)/n$. La quantité d'information associée au caractère x est de $-\log p(x)$[1]. L'entropie de Shannon de la chaîne de caractères est de $-\sum_x p(x) \log p(x)$ où la somme est sur l'ensemble des caractères. La quantité d'information dans le texte est de $-\sum_x np(x) \log p(x)$. Cette quantité d'information est un seuil minimal: la plupart des techniques de compression statistiques (basées sur la fréquence des caractères) ne permettent pas d'utiliser moins de $-\sum_x np(x) \log p(x)$ bits.

Exemple 2. *Prenons un texte de 2 mots, l'un ayant une fréquence de 500 et l'autre, de 250, pour un total de 750 (*$n = 750$*). Nous avons alors* $p(1) = 500/750 = 2/3$ *et* $p(2) = 250/750 = 1/3$*. La quantité d'information du premier mot est de* $-\log_2(2/3) \approx 0,58$ *et la quantité d'information du second mot est de* $-\log_2(1/3) \approx 1,58$*. Pour coder le texte, il faut donc au moins* $(-(2/3)\log_2(2/3) - (1/3)\log_2(1/3)) \times 750 \approx 688$ *bits, soit environ 86 octets. Vous pouvez tester ce résultat en générant un texte de seulement 2 mots qui comporte les fréquences prescrites. Vous constaterez que, quels que soient vos efforts de compression avec zip ou avec un autre outil, vous n'arriverez pas, sans tricher, à un fichier qui fait moins de 688 bits.*

[1]En informatique, le logarithme est toujours en base deux.

6.3. Compression statistique

Il y a plusieurs techniques de compression statistiques qui sont utilisées dans les entrepôts de données tel que le codage de Huffman et la compression Lempel-Ziv-Welch [?]. Ces techniques compressent efficacement les données, surtout lorsque certains caractères, ou certains chaînes de caractères sont fréquemment répétées. Par contre, elles sont relativement lentes bien qu'il existe des stratégies permettant d'accélérer la décompression [?].

A lire : `http://fr.wikipedia.org/wiki/Codage_de_Huffman`

A lire : `http://fr.wikipedia.org/wiki/Lempel-Ziv-Welch`

6.4. Codage des différences

Le codage des différences compresse les données en stockant les différences entre les valeurs successives plutôt que les valeurs elle-mêmes. Si les valeurs sont triées, ou partiellement triées (par block), ce mode de compression peut être très efficice. Elle s'applique pour plusieurs types de données, incluant même les nombres à virgule flottante [?, ?, ?, ?]. On l'utilise dans les bases de données orientées-colonnes [?, ?, ?, ?].

Formellement, le code fonctionne comme suit. Nous voulons coder une liste d'entiers $x_1, x_2, \ldots$ dans $\{0, 1, 2, \ldots, N-1\}$. La première valeur est toujours stockée sans compression. Ensuite, on stocke les valeurs $x_{i+1} - x_i \bmod N^2$ pour $i = 1, 2, \ldots$. On s'attend à ce que la plupart du temp, cette différence soit petite. Il se trouve qu'on peut stocker en mémoire de petits entiers en utilisant peu de bits [?, ?, ?, ?]. La technique la plus simple est le codage à octet variable [?, ?, ?, ?] : on stocke les premiers 7 bits de la valeur, puis on utilise un bit supplémentaire pour indiquer si un octet supplémentaire sera nécessaire pour le stockage, et ainsi de suite.

6.5. Codage des chaînes fréquentes

Il existe une stratégie de compression remarquablement simple qui est souvent très efficace. Le codage des longues chaînes répétées [?] est utilisé par Google [?] et dans le système de bases de données IBM DB2. Le principe est fort simple : on traverse l'ensemble de la base de données, et on cherche de longues chaînes de caractères répétées. On construit alors un dictionnaire de ces chaînes, et chaque fois qu'on en rencontre une dans une table, on la remplace par un identifiant correspondant à la chaînes en question.

[2] On peut aussi stocker la valeur $x_i \oplus x_{i+1}$ où $\oplus$ est le ou exclusif.

Exemple 3. *Prenons la chaîne de caractères* `aaabbbacdbacdbacdbabbbb`*. On découvre la chaîne de caractères* `bacd`*. On lui associe donc un identifiant, par exemple l'entier 1, et on substitue cet identifiant dans la chaîne originale:* `aaabb111babbbb`*.*

6.6. Compression des préfixes

Il se produit souvent que, dans une séquence de valeurs, plusieurs préfixes sont récurrents. Prenons par exemple la séquence AAABB, AAABCC, AAACCC. La compression des préfixes calcule d'abord une chaîne de caractères qui permettra d'identifier des préfixes. Dans ce cas particulier, prenons la chaîne AAABCC. On prend alors chaque chaîne de caractères dans notre séquence et on la compare avec cette chaîne de référence. D'abord on constate que la chaîne AABB réutilise les 4 premiers caractères de la chaîne de référence: on peut donc la stocker comme étant 4B. La chaîne AAABCC est indentique à la chaîne de référence: il n'y a donc rien à stocker. La dernière chaîne, AAACCC, peut être codée comme étant `3ccc`.

A consulter : `http://msdn.microsoft.com/en-us/library/cc280464.aspx`

6.7. Questions d'approfondissement

(a) Le site Twitter permet aux utilisateurs de publier des messages de 140 caractères ou moins. Est-ce que la compression Lempel-Ziv-Welch est appropriée dans ce cas?

(b) Soit les nombres de 0 à $2^{32}-1$. En utilisant 32 bits par nombre, combien faut-il d'octets pour les stocker? En utilisant le codage des différences, quel est le taux de compression?

6.8. Réponses suggérées

(a) La compression Lempel-Ziv-Welch ne compresse que les chaînes de caractères suffisamment longues. Malheureusement, elle risque d'être inefficace sur des chaînes de 140 caractères.

(b) Il faut $2^{32} \times 4 = 2^{34}$ octets pour stocker la liste sans compression. Avec le codage des différences, chaque différence est de 1, ce qui ne requiert qu'un seul octet. Le taux de compression est donc de 1:4.