Association et comparaison de méthodes de compression d'un flux de données

Positionnement thématique

Informatique pratique, Informatique théorique, Probabilités, Étude statistique

Mots-clés

Compression de données	Data compression
Méthode adaptative	Adaptive method
Méthode stationnaire	Stationary method
Codage arithmétique	Arithmetic coding
Modélisation	Modeling
Transformée	Transform

Bibliographie commentée

La compression de données sans pertes consiste à réduire la taille d'un document quelconque sans dégrader l'information qu'il contient, c'est-à-dire être capable de restituer parfaitement l'information initiale au cours de la décompression[1], [2], [3].

La compression se décompose alors en deux phases, la modélisation et le codage des données. La première phase consiste à définir un modèle pour les données, le plus proche de la réalité possible, afin d'estimer au mieux les probabilités d'apparition des différents symboles qui composent la source[1], ces dernières définissant alors la compression théorique maximale. Or la recherche de la modélisation parfaite n'est pas un problème "calculable" (or. *computable*), c'est plutôt, dans une certaine mesure, un problème d'intelligence artificielle[1]. En effet, les meilleurs estimations de la probabilité d'apparition d'un symbole suivant le contexte ne peuvent être effectuées qu'avec une compréhension parfaite de la source (et donc du type de documents : texte, photo...). La recherche de la meilleure modélisation se limitera donc ici à une étude expérimentale.

La seconde phase, le codage, consiste à associer un code binaire à chaque symbole ou groupe de symboles de la source en fonction de sa probabilité d'appartion calculée avec le modèle. Or contrairement à l'étape précédente, c'est un problème résolu, dont on connaît les résultats optimaux atteignables[1]. Le célèbre code de Huffman[1], [2], [3], [4], que j'utiliserai au moins pour le début, est proche de la solution optimale, et s'il force à coder chaque symbole sur un nombre entier de bits, il reste cependant relativement simple à implémenter. La solution optimale peut elle être atteinte avec le codage arithmétique ou le codage asymétrique[1], qui sont tous deux utilisés aujourd'hui dans les meilleurs algorithmes de compression.

Une troisième phase, optionnelle, dite "transformée" (et qui se fond parfois dans l'étape de codage), peut prendre place dans le processus. Elle consiste alors à appliquer une transformation réversible sur les données de manière à simplifier le modèle ou à réduire la taille de l'ensemble. On peut par exemple citer la transformée de Burrows-Wheeler qui permet – de manière réversible – de changer l'ordre des symboles de la source de manière à placer le plus souvent possible des symboles identiques côte-à-côte[1].

C'est alors le chaînage et l'interfaçage de ces deux ou trois méthodes (modèle, éventuellement transformée et codage) qui permettent de créer un algorithme de compression. Par exemple, le fameux bzip2, très utilisé sous Linux (extension (`.tar`)`.bz2`), utilise une association de transformée de Burrows-Wheeler et de codage de Huffman (entre autres)[1].

Enfin, certains des meilleurs algorithmes à l'heure actuelle comme paq8px (1$^{\text{er}}$ sur *Maximum Compression benchmark* en 2009)[1] tentent de reconnaître le type de données afin d'adapter la méthode de compression. Je ne m'intéresserai pas à ce genre de méthodes ici, trop avancées et trop compliquées à évaluer à mon niveau. Je me contenterai d'étudier des algorithmes "universels", qui appliquent le même traitement à tout type d'entrée, et qui donnent néanmoins des résultats convaincants.

Problématique retenue

Le chaînage de méthodes de compression n'est en général pas possible car à la sortie d'une première étape de compression, les données perdent la régularité et les structures qui sont justement utilisées pour produire un modèle efficace. Il s'agira donc de vérifier dans quelle mesure le chaînage est efficace (ou non), ainsi que de mesurer les gains apportés par une étape supplémentaire de transformée. Je comparerai enfin les différents résultats entre eux et avec les standards actuels.

Objectifs du TIPE

1. Association de méthodes de compression : j'essayerai de chaîner différents algorithmes de compression (ou de récursivement compresser les mêmes données avec le même algorithme) pour mesurer les gains éventuels.
2. Mesures : je mesurerai l'impact du choix modèle - [transformée] - codage sur les performances de l'algorithme de compression aussi bien au niveau du taux de compression que de la vitesse d'exécution.
3. Optimisations : dans cette partie très expérimentale, l'objectif sera d'ajuster les différents paramètres des meilleurs algorithmes que j'aurai implémenté et de les comparer aux meilleurs solutions industrielles actuelles.

Les tests seront effectués sur le *Calgary corpus*, qui malgré son âge, nous permettra de disposer de nombreux résultats concernant les solutions déjà existantes, qui ont quasiment toutes été testées sur ce dernier.

Les langages utilisés pour l'étude seront le Python pour le prototypage, et le C pour l'implémentation des méthodes retenues (pour lesquelles des mesures de taux de compression et de temps d'exécution seront effectuées).

Références

[1] Matt Mahoney. *Data Compression Explained*. `http://mattmahoney.net/dc/dce.html` : Dell, 2010.

[2] Pierre Brémaud. *Introduction aux Probabilités*. ISBN 3-540-13612-6. Springer, 1997.

[3] T. W. Körner. *Coding and Cryptography*. T. Part II. `http://www.dpmms.cam.ac.uk/~twk/Shan.pdf` : DPMMS, 2017.

[4] Vincent Beaudoin. "Développement de nouvelles techiques de compression de données sans perte". Mém.de mast. `http://www.theses.ulaval.ca/2009/25945/25945.pdf` : Université de Laval - Québec, 2008.

[5] Jean-Pierre Tillich. "Algorithmes de codage de source adaptatifs". In : *Théorie de l'information*. T. Cours 4. `https://www.rocq.inria.fr/secret/Jean-Pierre.Tillich/enseignement/X2015/cours4.pdf`, 2016.

[6] Université Pierre et Marie Curie. "Huffman dynamique". In : *Algorithmique Avancée*. T. Master. premier lien de la recherche Google `[huffman dynamique jussieu]`, 2010.

[2] et [3] sont des documents généraux qui nous ont permis d'entrer dans le domaine de la compression de données et de comprendre ses règles essentielles. [4] permet de découvrir de nouvelles voies pour améliorer les algorithmes existants, avec une certaine originalité.

[5] et [6] sont des documents techniques qui nous ont apporté des informations fiables pour l'implémentation d'algorithmes adaptatifs, c'est-à-dire qui ne nécessitent pas une première lecture totale de la source, et qui évoluent donc au fur et à mesure de la lecture.

Enfin, [1] est une "mine d'or" qui nous a permis de beaucoup progresser dans notre travail, avec une approche pratique, permettant de comprendre comment fonctionnent les meilleurs algorithmes de compression aujourd'hui, ainsi que d'avoir de bonnes bases pour réaliser les nôtres.

Notons que l'auteur de *Data Compression Explained*[1], Matt Mahoney, est un expert reconnu de la compression de données ainsi que le créateur et mainteneur de la famille d'algorithmes de compression ZPAQ, qui sont en tête de la plupart des *benchmarks*.