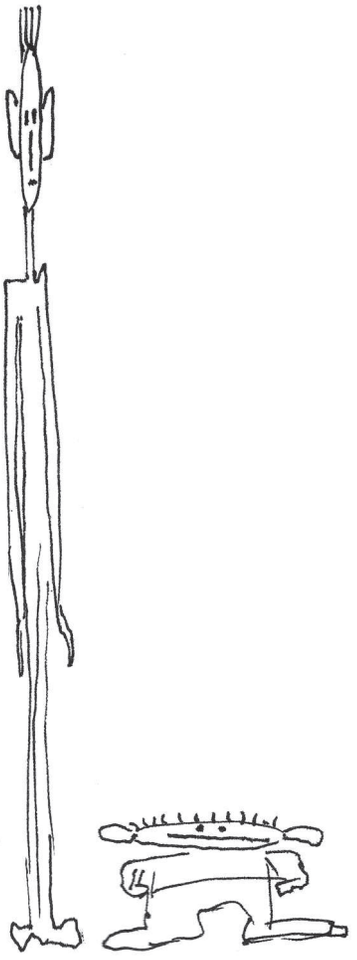
Współczesne metody kompresji bezstratnej

Przemysław SKIBIŃSKI[*]



Oto przykład: biorąc jako kontekst słowo „bezstrat”, możemy z dużym prawdopodobieństwem przewidzieć, że następnym symbolem będzie „n”. Gdybyśmy rozważali krótszy kontekst, np. „rat”, to prawdopodobieństwo pojawienia się „n” byłoby dużo niższe, ponieważ możemy spodziewać się także symboli „a” (np. w słowie „rata”), „l” („ratler”), „o” („ratować”), „u” („ratusz”), „y” („ratyfikacja”) itp.

Celem kompresji danych jest zmniejszenie rozmiaru składowanych danych lub czasu potrzebnego do ich transmisji. Kompresja może być bezstratna, co oznacza, że jest to proces w pełni odwracalny i dane po dekompresji są identyczne z danymi oryginalnymi. Kompresja jest nazywana stratną, gdy dane po dekompresji są jedynie przybliżeniem oryginalnych danych. Algorytmy stratne osiągają zwykle lepszą efektywność kompresji, jednak ten rodzaj kompresji stosuje się głównie w kompresji obrazów, muzyki i filmów, gdzie pewna utrata danych jest akceptowalna lub niezauważalna.

Proces kompresji danych możemy podzielić na dwie fazy: modelowanie i kodowanie. W fazie modelowania próbuje się znaleźć regularności, podobieństwa, nadmiarowość i inne korelacje. W tej fazie dane przekształcane są do postaci pośredniej, w której usunięto korelacje. Tak przekształcone dane są w kolejnej fazie kodowane za pomocą tzw. kodowania entropijnego, do którego zaliczamy kodowanie Huffmana i kodowanie arytmetyczne. Udowodniono już optymalność pewnych cech algorytmów entropijnych, dlatego większość badań prowadzonych w dziedzinie kompresji dotyczy fazy modelowania. Faza ta jest znacznie trudniejsza od fazy kodowania, ponieważ zależy od rodzaju kompresowanych danych.

Współcześnie używane metody bezstratnej kompresji danych można podzielić na cztery klasy: LZ77, BWT, PPM i CM. Najpopularniejszym algorytmem jest **LZ77** (od nazwisk autorów Lempel, Ziv i roku powstania 1977), znany głównie z programów zip i gzip. Główną ideą tego algorytmu jest używanie wcześniej zakodowanych danych jako słownika. W każdym kroku kodowania w słowniku wyszukuje się dopasowania z danymi do zakodowania. Następnie entropijnie koduje się jedynie miejsce oraz długość dopasowania. Algorytm LZ77 charakteryzuje się średnim stopniem kompresji, co jest akceptowalne przy jego wysokiej prędkości kompresji, kilka razy większej prędkości dekompresji oraz małych wymaganiach pamięciowych.

Kolejną metodą kompresji bezstratnej jest **BWT** (Burrows–Wheeler Transform, rok 1994), znana przede wszystkim z programu bzip2. Główną składową tej metody jest transformacja, która dzieli dane na bloki i permutuje dane w tych blokach w celu uzyskania dłuższych serii identycznych symboli oraz większej ilości serii. W następnym kroku tak przetransformowane dane przekształcane są na liczby całkowite przy użyciu heurystyki MTF (Move-to-Front) i kodowane entropijnie. Heurystyka MTF zawiera listę wszystkich możliwych symboli, przy czym ostatnio występujące symbole przenoszone są na początek listy. Przy założeniu, że te same symbole występują blisko siebie, są one zamieniane przez MTF w małe liczby całkowite, które odpowiadają pozycji danego symbolu na liście. Algorytm BWT charakteryzuje się lepszym stopniem kompresji niż LZ77, jednak jest wolniejszy oraz ma większe wymagania pamięciowe.

Następną metodą bezstratnej kompresji danych jest **PPM** (Prediction by Partial Matching, rok 1984). Kodowanie PPM polega na określeniu prawdopodobieństwa warunkowego symbolu do zakodowania w kontekście poprzedzających go symboli. Nowy symbol jest kodowany arytmetycznie, zajmując liczbę bitów odwrotnie proporcjonalną do przypisanego mu prawdopodobieństwa warunkowego. Zwiększenie długości kontekstu powoduje lepsze określenie prawdopodobieństwa warunkowego nowego symbolu, co wiąże się także z lepszą kompresją, jednak także z większym zużyciem pamięci.

Algorytm PPM ma wysoki stopień kompresji, podobny czas kompresji i dekompresji oraz wymagania pamięciowe, które rosną wykładniczo w stosunku do długości kontekstu. Jednakże wraz ze wzrostem szybkości komputerów i ilości posiadanej przez nie pamięci algorytm PPM znalazł praktyczne zastosowanie w znanych archiwizerach WinRAR i WinZip.

Ostatnią klasą metod bezstratnej kompresji danych jest **CM** (Context Mixing), której najbardziej znanym przedstawicielem jest **PAQ**. Algorytmy z klasy CM

[*]Instytut Informatyki, Uniwersytet Wrocławski

używają wielu różnych modeli, które znajdują korelacje w danych. Podczas kodowania preferowane są modele, które dawały do tej pory najlepsze rezultaty.

Twórcą algorytmu PAQ, który powstał w roku 2002, jest Matt Mahoney. Algorytm PAQ bazuje na algorytmie PPM, ale używa alfabetu binarnego, a nie 8-bitowego, co obniża prędkość kompresji, jednak znacznie upraszcza kodowanie i daje nowe możliwości. W PAQ1, oprócz predykcji (jak w PPM), pojawił się model słownikowy (działający podobnie do algorytmu LZ77), model dla słów w językach naturalnych oraz model dla rekordów stałej długości. Każdy z tych modeli określa w kolejnym kroku prawdopodobieństwo pojawienia się jedynki. Ostateczne prawdopodobieństwo zależy od wszystkich modeli, przy czym modele lepiej sprawdzające się w przeszłości mają większy wpływ. Algorytm PAQ1 jest wolniejszy od PPM i ma gorszy stopień kompresji. Jednak od wersji PAQ6 metoda ta rywalizuje już z najlepszymi implementacjami algorytmu PPM, które do tej pory zwyciężały z konkurentami.

Najnowsza wersja algorytmu PAQ, czyli PAQ8 z roku 2006 ([1]), obsługuje już 16 różnych modeli. Oprócz modeli znanych z PAQ1, zawiera on specjalne modele dla plików wykonywalnych, dla 24-bitowych nieskompresowanych obrazów, dla 2-bitowych obrazów (np. faksów), a nawet dla skompresowanych obrazów JPEG.

W PAQ8 wyjścia z każdego modelu, czyli prawdopodobieństwa pojawienia się jedynki, przechodzą przez kilka (do czterech) z kilkuset dostępnych sieci neuronowych, które są tak zaprojektowane, że faworyzują modele lepiej sprawdzające się w przeszłości, przez co lepiej określają prawdopodobieństwo, co umożliwia później lepszą kompresję.

Wydajność algorytmów kompresji jest mierzona trzema głównymi kryteriami: efektywnością kompresji, złożonością (wymagania sprzętowe, złożoność algorytmiczna) i prędkością (czas kompresji i dekompresji). Wydajność testuje się na ustalonych zbiorach plików zwanych korpusami. W 2006 r. pojawił się nowy korpus, składający się jedynie z jednego pliku o wielkości 10^8 bajtów, będącego początkiem archiwum angielskiej wersji Wikipedii. Jego głównym celem jest wsparcie badań w dziedzinach sztucznej inteligencji oraz przetwarzania języków naturalnych, co, według autorów, wiąże się z modelowaniem danych przy kompresji tekstu. W celu zachęty dla naukowców ustanowiono Nagrodę Huttera ([2]), z funduszem 50 000 euro, która nagradza za polepszenie najlepszych dotychczas wyników kompresji tego pliku.

Program	Metoda	Rozmiar po kompresji (B)	Czas kompresji (sek.)	Czas dekompresji (sek.)	Pamięć (MB)
gzip 1.3.5 -9	LZ77	36 445 248	10,7	5,6	1,4
bzip2 1.0.2 -9	BWT	29 008 736	37,9	12,9	8
ppmd J1 -m256 -o10	PPM	21 388 296	88,0	89,5	256
paq8f -7	CM	18 289 559	4896,0	4973,4	854
paq8hp8 -8	CM	16 372 960	6469,3	6669,3	1849

Wyniki kompresji pliku z konkursu o Nagrodę Huttera (100 000 000 B).

Tabela przedstawia efektywność kompresji, czas kompresji oraz wymagania pamięciowe dla głównych metod bezstratnej kompresji na pliku z Nagrody Huttera. Wszystkie metody, oprócz ostatniej, są metodami uniwersalnymi. Ostatnia metoda, oparta na PAQ8, jest zgłoszona do Nagrody Huttera i wykorzystuje specjalne słowniki oraz ma modele zoptymalizowane pod ten konkretny plik.

[1] Matt Mahoney, Program PAQ8, `http://www.cs.fit.edu/~mmahoney/compression`

[2] Marcus Hutter, Nagroda Huttera, `http://prize.hutter1.net`

Można zauważyć, że w zastosowaniach, w których prędkość kompresji oraz wymagania pamięciowe nie są najważniejsze, algorytm PPM sprawdza się najlepiej, uzyskując 41% poprawę kompresji w stosunku do metody LZ77. Natomiast algorytm PAQ8 uzyskuje poprawę rzędu 50% lub więcej, jednak przy prędkości 500 razy mniejszej od LZ77. Algorytm PAQ8 uzyskuje obecnie jedne z najwyższych stopni kompresji, jednak jego wysokie wymagania pamięciowe (liczone w setkach megabajtów) oraz bardzo niska prędkość kompresji czynią go nieatrakcyjnym z praktycznego punktu widzenia. Podobna sytuacja była z algorytmem PPM, który ponad 10 lat czekał na zoptymalizowaną implementację oraz zwiększenie mocy obliczeniowej komputerów, aż wreszcie znalazł praktyczne zastosowanie.