PERTEMUAN 7
COMPRESSION

A. TUJUAN PEMBELAJARAN

Pada pertemuan ini akan dijelaskan mengenai kompresi data. Setelah mempelajari materi ini mahasiswa diharapkan mampu untuk:

1. Menjelaskan konsep kompresi data
2. Memahami pemodelan dan pengkodean dalam kompresi data
3. Memahami jenis kompresi data
4. Memahami teknik kompresi data .

B. URAIAN MATERI

1. Menjelaskan Konsep Kompresi Data

Kompresi data adalah sebuah proses yang mengubah data input menjadi data output dengan ukuran yang lebih kecil. Data dapat berupa karakter dalam file teks, angka yang merupakan contoh bentuk gelombang ucapan atau gambar, atau urutan angka yang dihasilkan oleh proses lain. Alasan kita membutuhkan kompresi data adalah karena semakin banyak informasi yang kita hasilkan dan gunakan dalam bentuk digital terdiri dari angka yang direpresentasikan oleh byte data.

a. Teknik Kompresi Data

1) Lossless Compression

Teknik kompresi lossless, tidak melibatkan kehilangan informasi. Jika data telah dikompresi tanpa kehilangan, maka data asli dapat dipulihkan dengan tepat dari data yang dikompresi. Kompresi lossless umumnya digunakan untuk aplikasi yang tidak dapat mentolerir perbedaan apa pun antara data asli dan data rekonstruksi.

Kompresi teks adalah area penting untuk kompresi lossless. Sangat penting bahwa rekonstruksi identik dengan teks asli, karena perbedaan yang sangat kecil dapat menghasilkan pernyataan dengan arti yang sangat berbeda. Pertimbangkan kalimat "Jangan kirim uang" dan "Jangan

kirim uang". Argumen serupa berlaku untuk file komputer dan untuk jenis data tertentu seperti catatan bank.

Jika data dalam bentuk apa pun akan diproses atau "ditingkatkan" nanti untuk menghasilkan lebih banyak informasi, integritasnya harus dijaga. Misalnya, kita mengompresi gambar radiologis menjadi lossy mode dan perbedaan antara rekonstruksi Y dan X asli tidak dapat dideteksi secara visual. Jika gambar ini kemudian disempurnakan, perbedaan yang sebelumnya tidak terdeteksi dapat menyebabkan munculnya arti fakta yang dapat menyesatkan ahli radiologi.

Data yang diperoleh dari satelit seringkali diolah kemudian untuk mendapatkan indikator numerik yang berbeda dari vegetasi, deforestasi, dan lain sebagainya. Jika data yang direkonstruksi tidak identik dengan data asli, pemrosesan dapat menghasilkan “peningkatan” atau perbedaan. Mungkin tidak mungkin untuk kembali dan mendapatkan data yang sama lagi. Oleh karena itu, tidak disarankan untuk membiarkan perbedaan apa pun muncul pada proses kompresi.

2) Lossy Compression

Teknik kompresi lossy melibatkan beberapa kehilangan informasi, dan data yang telah dikompresi menggunakan teknik lossy. Pada umumnya tidak dapat dipulihkan atau direkonstruksi dengan tepat. Sebagai imbalan untuk menerima distorsi ini dalam rekonstruksi, kita dapat memperoleh rasio kompresi yang jauh lebih tinggi daripada yang dimungkinkan dengan kompresi lossless.

b. Ukuran Kinerja

Algoritma kompresi dapat dievaluasi dengan beberapa cara yang berbeda. Kita dapat mengukur kompleksitas relatif dari algoritma memori yang diperlukan untuk mengimplementasikan algoritma, seberapa cepat kinerja algoritma pada mesin tertentu. Jumlah kompresi dan seberapa dekat algoritma tersebut menyerupai rekontruksi aslinya.

Cara yang sangat logis untuk mengukur seberapa baik algoritme kompresi kumpulan data tertentu. Dengan melihat rasio jumlah bit yang diperlukan untuk mewakili data sebelum kompresi dengan jumlah bit yang diperlukan untuk merepresentasikan data setelah kompresi. Rasio ini disebut

rasio kompresi. Misalkan menyimpan gambar yang terdiri dari larik persegi 256 × 256 piksel membutuhkan 65.536 byte. Gambar dikompresi dan versi terkompresi membutuhkan 16.384 byte. Kami akan mengatakan bahwa rasio kompresinya adalah 4:1. Kita juga dapat merepresentasikan rasio kompresi dengan mengungkapkan pengurangan jumlah data yang diperlukan sebagai persentase dari ukuran data asli. Dalam contoh khusus ini, rasio kompresi yang dihitung dengan cara ini adalah 75%.

2. Memahami Pemodelan dan Pengkodean dalam Kompresi Data

Meskipun persyaratan rekonstruksi dapat memaksa keputusan apakah skema kompresi menjadi lossy atau lossless. Skema kompresi yang tepat yang kami gunakan akan bergantung pada sejumlah faktor yang berbeda. Beberapa faktor terpenting adalah karakteristik data yang perlu dikompresi. Teknik kompresi yang akan bekerja dengan baik untuk kompresi teks mungkin tidak berfungsi dengan baik untuk mengompresi gambar. Setiap aplikasi menyajikan serangkaian tantangan yang berbeda.

Harus diingat bahwa kompresi data merupakan ilmu eksperimental. Pendekatan yang paling sesuai untuk aplikasi tertentu akan sangat bergantung pada redundansi yang melekat dalam data.

Pengembangan algoritma kompresi data untuk berbagai data dapat dibagi menjadi dua tahap. Fase pertama biasanya disebut sebagai pemodelan. Dalam fase ini, kami mencoba mengekstrak informasi tentang redundansi yang ada dalam data dan mendeskripsikan redundansi tersebut dalam bentuk model. Tahap kedua disebut pengkodean. Deskripsi model dan "deskripsi" tentang perbedaan data dari model dienkode, biasanya menggunakan alfabet biner. Perbedaan antara data dan model sering disebut sebagai residual. Dalam tiga contoh berikut, kita akan melihat tiga cara berbeda untuk memodelkan data. Kami kemudian akan menggunakan model untuk mendapatkan kompresi.

Tabel 7. Consider the following sequence of numbers {x1, x2, x3, . . . }:

9	11	11	11	14	13	15	17	16	17	20	21

Jika kita mengirim atau menyimpan representasi biner dari angka-angka ini, kita perlu menggunakan 5 bit per sampel. Namun, dengan mengeksploitasi struktur dalam data, kita dapat merepresentasikan urutan tersebut menggunakan bit yang lebih sedikit. Jika kita memplot data ini seperti yang ditunjukkan pada Gambar 1, kita melihat bahwa data tampak jatuh pada garis lurus. Oleh karena itu, model untuk data dapat berupa garis lurus yang diberikan oleh persamaan.

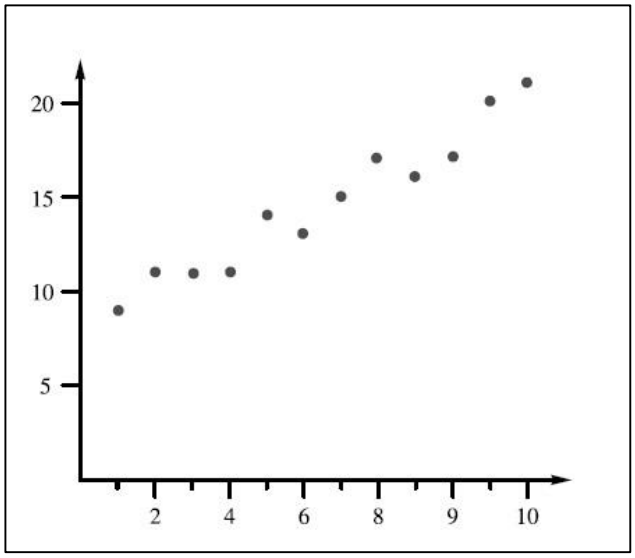
$$Xn = n+8 \; n = 1, 2, \ldots$$

Struktur deret angka khusus ini dapat dikarakterisasi dengan persamaan. Jadi, ˆx1 = 9, x1 = 9, ˆx2 = 10, x2 = 11, dan seterusnya. Untuk memanfaatkan struktur ini, mari kita periksa Perbedaan antara data dan model. Perbedaan (atau sisa) diberikan oleh urutan.

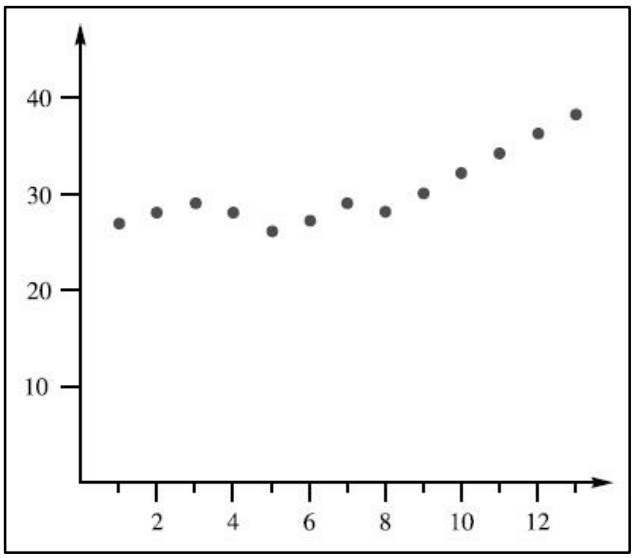
$$en = xn - \hat{x}n : 0\ 1\ 0\ -1\ 1\ -1\ 0\ 1\ -1\ -1\ 1\ 1$$

Urutan sisa hanya terdiri dari tiga angka {−1, 0, 1}. Jika kita menetapkan kode 00 hingga −1, kode 01 hingga 0, dan kode 10 hingga 1, kita perlu menggunakan 2 bit untuk mewakili setiap elemen dari urutan residual. Oleh karena itu, kita dapat memperoleh kompresi dengan mengirimkan atau menyimpan parameter model dan urutan sisa. Pengkodean dapat dilakukan secara tepat jika kompresi yang diperlukan adalah lossless, atau perkiraan jika kompresi dapat menjadi lossy.

Jenis struktur atau redundansi yang ada dalam data ini mengikuti hukum sederhana. Setelah kita mengenali hukum ini, kita dapat menggunakan struktur untuk memprediksi nilai setiap elemen dalam urutan dan kemudian menyandikan sisa. Struktur jenis ini hanyalah salah satu dari sekian jenis struktur.



Gambar 13. Urutan Nilai Data



Gambar 14. Urutan Nilai Data

a. THE HUFFMAN CODING ALGORITHM

Prosedur Huffman didasarkan pada dua pengamatan mengenai kode prefiks yang optimal:

1) Dalam kode yang optimal, simbol yang muncul lebih sering (memiliki kemungkinan kemunculan yang lebih tinggi) akan memiliki codeword yang lebih pendek daripada simbol yang lebih jarang muncul.

2) Dalam kode yang optimal, dua simbol yang paling jarang muncul akan memiliki panjang yang sama.

Sangat mudah untuk melihat bahwa pengamatan pertama benar. Jika simbol yang muncul lebih sering memiliki codeword yang lebih panjang daripada codeword untuk simbol yang lebih jarang muncul, jumlah rata-rata bit per simbol akan lebih besar daripada jika kondisinya dibalik. Oleh karena itu, kode yang menetapkan codeword yang lebih panjang ke simbol yang muncul lebih sering tidak dapat optimal.

b. NONBINARY HUFFMAN CODE

Prosedur pengkodean biner Huffman dapat dengan mudah diperluas ke kasus non-biner di mana elemen kode berasal dari alfabet m-binary, dan m tidak sama dengan dua. Ingatlah bahwa kami memperoleh algoritma Huffman berdasarkan pengamatan bahwa dalam kode awalan biner yang optimal:

1) Simbol yang muncul lebih sering (memiliki probabilitas kemunculan yang lebih tinggi) akan memiliki codeword yang lebih pendek daripada simbol yang lebih jarang muncul.
2) Kedua simbol yang paling jarang muncul akan memiliki panjang yang sama.

c. ADAPTIVE HUFFMAN CODING

Pengkodean Huffman membutuhkan pengetahuan tentang kemungkinan urutan sumber. Jika pengetahuan ini tidak tersedia, pengkodean Huffman menjadi prosedur dua jalur: statistik dikumpulkan pada lintasan pertama, dan sumber dikodekan pada lintasan kedua. Untuk mengubah algoritma ini menjadi *one-pass* prosedur, Faller dan Gallagher secara independen mengembangkan algoritma adaptif untuk membangun kode Huffman berdasarkan statistik dari simbol yang sudah ditemukan. Ini kemudian diperbaiki oleh Knuth dan Vitter.

Secara teoritis, jika kita ingin mengkodekan simbol (k + 1) -th menggunakan statistik dari simbol k pertama, kita dapat menghitung ulang kode tersebut menggunakan prosedur pengkodean Huffman setiap kali sebuah simbol dikirimkan. Namun, ini tidak akan menjadi pendekatan yang sangat praktis karena banyaknya komputasi yang terlibat.

d. *GOLOMBS CODE*

Kode Golomb-Rice milik keluarga kode yang dirancang untuk menyandikan bilangan bulat dengan asumsi bahwa semakin besar bilangan bulat, semakin rendah kemungkinan terjadinya. Kode paling sederhana untuk situasi ini adalah kode unary. Kode unary untuk bilangan bulat positif n hanyalah n 1s diikuti oleh 0. Jadi, kode untuk 4 adalah 11110, dan kode untuk 7 adalah 11111110. Kode unary sama dengan kode Huffman untuk alpabet semi-infinite {1, 2, 3,. . . } dengan model probabilitas. Karena kode Huffman sudah optimal, kode unary juga optimal untuk model probabilitas ini.

$$P[k] = \frac{1}{2k}$$

3. Memahami Jenis Kompresi Data

a. MEDIA-SPECIFIC COMPRESSION

Media specific compression dirancang khusus untuk data media seperti gambar, audio, video, dan sejenisnya. Kemungkinan besar, jenis file dan kompresor ini membentuk sebagian besar konten yang dikirim, diterima, dimanipulasi, disimpan, dan ditampilkan oleh aplikasi Anda kepada pengguna. Pepatah lama, "Sebuah gambar bernilai ribuan kata," benar-benar benar dalam hal kompresi data: gambar 1024 x 1024 RGB adalah 3 MB data. Jika Anda menganggap huruf berenkode ASCII, Anda dapat menampilkan 3.145.728 huruf untuk ukuran yang sama.

Untuk memasukkannya ke dalam konteks, buku terkenal The Hobbit terdiri dari 95.022 kata. Jika Anda mengasumsikan ukuran kata rata-rata 5 huruf, itu berarti kira-kira 475.110 karakter. Anda dapat memasukkan buku itu sekitar 6 kali ke dalam satu gambar 1024 × 1024.

Inilah sebabnya mengapa sebagian besar kompresi media menggunakan algoritma kompresi lossy. Algoritme kompresi lossy adalah jenis transformasi data yang mengurangi kualitas media dalam upaya membuat konten lebih dapat dikompresi. Misalnya, gambar 1024 × 1024, masing-masing menggunakan 8 bit untuk saluran merah, hijau, dan biru menjadi 24 bit per piksel, dan karenanya 3 MB. Namun, jika Anda hanya menggunakan 4 bit per saluran, Anda akan berakhir dengan 12 bit per piksel,

sehingga total footprint menjadi 1,5 MB, sekaligus mengurangi kualitas warna gambar.

b. GENERAL-PURPOSE COMPRESSION

Kompresi umumnya adalah algoritme seperti DEFLATE, GZIP, BZIP2, LZMA, dan PAQ, yang menggabungkan berbagai transformasi lossless untuk menghasilkan penghematan untuk file nonmedia seperti teks, kode sumber, data serial, dan konten biner lainnya yang tidak akan mentolerir kompresi data lossy. Ada banyak penelitian yang sehat di bidang ini Berhenti dengan Tolak Ukur Kompresi dalam Teks Besar menunjukkan kumpulan kompresi tujuan umum yang semuanya telah ditugaskan untuk mengompresi file teks besar, untuk mengukur bagaimana mereka menumpuk satu sama lain. Dan algoritma baru terus dikembangkan. Upaya Google dalam menyempurnakan algoritme GZIP menghasilkan kelompok kompresi yang disebut Snappy, Zopfli, Gipfeli, dan Brotli, 3 dengan masing-masing berfokus pada kompresi yang lebih baik, persyaratan memori yang lebih rendah, atau dekompresi yang lebih cepat.

Sebagian besar konten Internet yang kita unduh setiap hari telah dikompresi dengan salah satu algoritme berikut. Tumpukan HTTP standar memungkinkan paket data untuk dikodekan dengan GZIP, BZIP, dan sekarang, kompresor Brotli (selama server dan klien mendukungnya), yang berarti laman web, file JavaScript, tweet, dan daftar toko kemungkinan besar muncul di perangkat Anda setelah didekompresi.

4. Memahami teknik kompresi data

Sebelum beralih ke gangguan kompresi ke setiap bagian aplikasi mewah kita, penting untuk mencatat semua *trade-off* dan kasus penggunaan yang terlibat. Tidak setiap algoritme cocok untuk setiap kasus penggunaan, dan dalam beberapa kasus, implementasi yang berbeda dari format kompresi yang sama mungkin lebih sesuai dengan kebutuhan kita.

a. Skenario Penggunaan Kompresi

Mari kita mulai diskusi ini dengan menyetel tahapan di mana data dikompresi, disimpan, dan didekompresi. Ini sangat penting untuk memahami dari mana data berasal dan ke mana perginya, karena interaksi penting antara

pembuat enkode dan dekoder, yang akan kita bicarakan lebih lanjut nanti. Pertama, mari kita lihat empat skenario umum.

b. *Compressed Offline, Decompressed On-Client*, Dalam skenario pertama ini, data dikompresi di suatu tempat yang tidak terkait dengan klien, lalu didistribusikan ke klien, tempat data tersebut didekompresi untuk digunakan.
c. *Compressed On-Client, Decompressed In-Cloud,* Sebagian besar aplikasi media sosial modern menghasilkan banyak konten di klien dan kemudian mendorongnya ke cloud untuk diproses dan didistribusikan ke sesama pengguna sosial lainnya. Dalam situasi ini, beberapa kompresi ringan biasanya dilakukan pada klien, untuk mengurangi jumlah overhead dalam komunikasi keluar. Misalnya, mengambil paket data sosial, dan membuat serialisasi dengan format serialisasi biner, lalu mengompresi gzip sebelum mengirimnya ke server.
d. *Compressed In-Cloud, Decompressed On-Client,* Data yang dikompresi di cloud dibagi menjadi dua bucket utama, yang memiliki karakteristik yang sangat berbeda, yaitu:
    1) *Dynamic data that is generated by the cloud resource*, ketika klien meminta hasil dari beberapa operasi database, atau file server mengirimkan data tata letak dinamis, klien menunggu konten dibuat. Waktu yang dibutuhkan server untuk membuat dan mengompresi data itu sangat penting; jika tidak, klien akan mengalami latensi jaringan.
    2) *Large data that's passed off to the cloud for efficient computing*, Pentingnya skenario ini sering kali didorong untuk memastikan minimalisasi bit untuk media yang ada. Misalnya, bayangkan memiliki dua gigabyte file PNG yang perlu dikonversi ke gambar WebP dengan 10 resolusi berbeda, atau 1.200 jam video yang harus dikonversi ke H.264 sebelum ditampilkan.
e. *Compressed On-Client, Decompressed On-Client,* Dalam kasus ini, klien menghasilkan data, mengompresnya, dan kemudian mengirimkannya ke klien lain untuk mendekompresi.

a. Kebutuhan Kompresi

Seperti yang telah kami tunjukkan saat menjelajahi algoritme yang berbeda, sangat penting untuk memahami bahwa tidak semua algoritme dan format kompresi berlaku untuk semua jenis data. Misalnya, menerapkan

Huffman ke data gambar tidak akan menghasilkan tingkat penghematan yang dapat dilakukan dengan menerapkan algoritme kompresi image2 lossy.

Mencocokkan algoritme yang tepat dengan jenis data yang tepat sangat penting untuk memaksimalkan hasil kompresi yang Anda inginkan, dengan pengorbanan yang harus Anda lakukan. Dan yang perlu di catat adalah :

1) Ketahui data anda, bukan hanya jenis datanya, tetapi juga struktur internalnya, dan khususnya, cara penggunaannya.
2) Ketahui opsi algoritme anda sehingga dapat memilih kompresi yang tepat
3) Yang terpenting, ketahuilah apa yang anda butuhkan.

b. Compression Ratio

Rasio kompresi, yaitu jumlah konten yang dikompresi, sering kali merupakan faktor terpenting saat mengevaluasi opsi kompresi. Karena tujuan utama dari kompresi adalah untuk menghasilkan bentuk data yang paling tersusun

c. Compression Performance

Kinerja kompresi adalah tentang berapa lama waktu yang dibutuhkan untuk memasukkan data ke dalam bentuk terkompresi. Kinerja kompresi sangat penting dalam lingkungan yang sensitif terhadap latensi, baik klien bertanggung jawab atas kompresi, atau server mengompresi data yang menunggu klien.

Biasanya ada dua metrik evaluasi yang perlu diperhatikan dalam hal ini: kecepatan CPU dan memori. Kecepatan CPU dari sistem encoding penting karena ini menentukan seberapa cepat data dapat dikompresi. Dan jumlah memori yang tersedia menjadi penting jika sangat terbatas, seperti yang terjadi pada perangkat.

C. SOAL LATIHAN/ TUGAS

1. Jelaskan definsi compression yang Anda ketahui!
2. Sebutkan dan jelaskan teknik dalam kompresi!
3. Apa yang dimaksud dengan Huffman coding!
4. Sebutkan dan jelaskan 4 compression usage skenario!

D. REFERENSI

Colt McAnlis, Aleks Haecky (2016), Understanding Compression Data Compression for Modern Developers Tersedia di : https://z-lib.org/

Khalid Sayood (2017), Introduction to Data Compression.Tersedia di: https://z-lib.org/

Jean-Guillaume Dumas, Jean-Louis Roch, Éric Tannier, Sébastien Varrette (2015), Foundations of Coding Compression, Encryption, Error Correction.Tersedia di : https://z-lib.org/