PENGANTAR KOMPRESI DATA

PUTU WIDHIARTHA
widhiartha@yahoo.com
http://widhiartha.multiply.com

Kompresi Data adalah salah satu subyek di bidang teknologi informasi yang saat ini telah diterapkan secara luas. Gambar-gambar yang anda dapatkan di berbagai situs internet pada umumnya merupakan hasil kompresi ke dalam format GIF atau JPEG. *File* video MPEG adalah hasil proses kompresi pula. Penyimpanan data berukuran besar pada *server* pun sering dilakukan melalui kompresi. Sayangnya tidak banyak mata kuliah yang memberikan perhatian pada subyek ini secara memadai. Tulisan berikut ini akan memperkenalkan tentang dasar-dasar Kompresi Data kepada anda.

Kompresi Data merupakan cabang ilmu komputer yang bersumber dari Teori Informasi. Teori Informasi sendiri adalah salah satu cabang Matematika yang berkembang sekitar akhir dekade 1940-an. Tokoh utama dari Teori Informasi adalah Claude Shannon dari Bell Laboratory. Teori Informasi mengfokuskan pada berbagai metode tentang informasi termasuk penyimpanan dan pemrosesan pesan. Teori Informasi mempelajari pula tentang redundancy (informasi tak berguna) pada pesan. Semakin banyak redundancy semakin besar pula ukurang pesan, upaya mengurangi redundancy inilah yang akhirnya melahirkan subyek ilmu tentang Kompresi Data.

Teori Informasi menggunakan terminologi entropy sebagai pengukur berapa banyak informasi yang dapat diambil dari sebuah pesan. Kata “entropy” berasal dari ilmu termodinamika. Semakin tinggi entropy dari sebuah pesan semakin banyak informasi yang terdapat di dalamnya. Entropy dari sebuah simbol didefinisikan sebagai nilai logaritma negatif dari probabilitas kemunculannya. Untuk menentukan konten

informasi dari sebuah pesan dalam jumlah bit dapat digunakan rumus sebagai berikut:

number of bits = -log base 2 (probability)

Entropy dari keseluruhan pesan adalah jumlah dari keseluruhan entropy dari seluruh symbol.

Teknik Kompresi Data dapat dibagi menjadi dua kategori besar, yaitu:

1. Lossy Compression

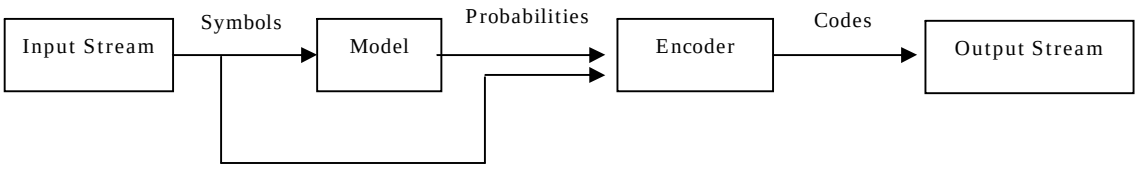
   Lossy compression menyebabkan adanya perubahan data dibandingkan sebelum dilakukan proses kompresi. Sebagai gantinya lossy compression memberikan derajat kompresi lebih tinggi. Tipe ini cocok untuk kompresi file suara digital dan gambar digital. File suara dan gambar secara alamiah masih bisa digunakan walaupun tidak berada pada kondisi yang sama sebelum dilakukan kompresi.

2. Lossless Compression

   Sebaliknya Lossless Compression memiliki derajat kompresi yang lebih rendah tetapi dengan akurasi data yang terjaga antara sebelum dan sesudah proses kompresi. Kompresi ini cocok untuk basis data, dokumen atau spreadsheet. Pada lossless compression ini tidak diijinkan ada bit yang hilang dari data pada proses kompresi.

Secara umum kompresi data terdiri dari dua kegiatan besar, yaitu Modeling dan Coding. Proses dasar dari kompresi data adalah menentukan serangkaian bagian dari data (stream of symbols) mengubahnya menjadi kode (stream of codes). Jika proses kompresi efektif maka hasil dari stream of codes akan lebih kecil dari segi ukuran daripada stream of symbols. Keputusan untuk mengindentikan symbols tertentu dengan codes tertentu adalah inti dari proses modeling. Secara umum dapat diartikan bahwa sebuah model adalah kumpulan data dan aturan yang menentukan pasangan antara symbol sebagai input dan code sebagai output dari proses kompresi. Sedangkan coding adalah proses untuk menerapkan modeling tersebut menjadi sebuah proses kompresi data.

Dengan menggunakan Huffman coding sebagai contoh, sebuah proses kompresi akan nampak seperti gambar berikut ini:



Gambar I Diagram Statistical Model dengan Huffman Code

Pada Huffman coding, symbols yang sering muncul akan diubah menjadi code dengan jumlah bit kecil, sedangkan yang jarang muncul akan menjadi code dengan jumlah bit besar.

I.Coding

Melakukan proses encoding dengan menggunakan ASCII atau EBDIC yang merupakan standar dalam proses komputasi memberikan kelemahan mendasar apabila dilihat dari paradigma kompresi data. ASCII dan EBDIC menggunakan jumlah bit yang sama untuk setiap karakter, hal ini menyebabkan banyak bit yang ”terbuang” untuk merepresentasikan karakter-karakter yang sebenarnya jarang muncul pada sebuah pesan.

Salah satu cara mengatasi permasalahan di atas adalah dengan menggunakan Huffman-coding. Huffman-coding yang dikembangkan oleh D.A. Huffman mampu menekan jumlah redundancy yang terjadi pada sebuah pesan yang panjangnya tetap. Huffman-coding bukanlah teknik yang paling optimal untuk mengurangi redundancy tetapi Huffman-coding merupakan teknik terbaik untuk melakukan coding terhadap symbol pada pesan yang panjangnya tetap.

Permasalahan utama dengan Huffman-coding adalah hanya bisa menggunakan bilangan bulat untuk jumlah bit dari setiap code. Jika entropy dari karakter tersebut adalah 2,5 maka apabila melakukan pengkodean dengan Huffman-coding karakter tersebut harus terdiri dari 2 atau 3 bit. Hal tersebut membuat Huffman-coding tidak bisa menjadi algoritma paling optimal dalam mengatasi redundancy ini. Pada contoh berikut ini dapat diamati contoh dari Huffman-code dengan asumsi bahwa karakter yang sering muncul seperti E,A dikodekan dengan jumlah bit lebih pendek daripada karakter yang jarang muncul seperti X,Q, dan Z.

Symbol	Huffman Code
E	100
A	101
.....	.....
X	01101111
Q	01101110001
Z	01101110000

Tabel 1 Contoh Coding dengan Algoritma Huffman

Huffman-coding memang kurang efisien karena untuk melakukan pengkodean kita harus menggunakan bilangan bulat. Walaupun demikian Huffman-coding sangat mudah digunakan dan sampai awal era 90-an di mana prosesor komputer masih sulit untuk melakukan komputasi bilangan pecahan Huffman-coding dianggap paling rasional untuk diaplikasikan pada proses kompresi data. Setelah kelahiran prosesor yang mampu melakukan operasi bilangan pecahan dengan cepat maka banyak algoritma coding baru bermunculan dan salah satunya adalah Arithmatic-coding.

Arithmatic coding lebih kompleks dan rumit dibandingkan Huffman-coding, tetapi di sisi lain Arithmatic-coding memberikan optimalisasi yang lebih tinggi. Arithmatic-coding memungkinkan jumlah bit dalam pecahan karena arithmatic coding tidak bekerja per karakter dari pesan tetapi langsung pada keseluruhan pesan. Misalnya entropy dari karakter e pada pesan adalah 1,5 bit maka pada output code pun jumlah bit-nya adalah 1,5 dan bukan 1 atau 2 seperti pada Huffman-coding. Contoh berikut ini dapat memberikan gambaran optimalisasi yang didapatkan antara menggunakan Huffman-coding dan Arithmatic-coding:

Symbol	Frekuensi Kemunculan pada Pesan	Entropy	Jumlah Bit dengan Huffman	Total Bit dengan Huffman	Total Bit dengan Arithmatic
E	20	1.26 bit	1 bit	20	25.2
A	20	1.26 bit	2 bit	40	25.2
X	3	4.00 bit	3 bit	9	12.0
Y	3	4.00 bit	4 bit	12	12.0
Z	2	4.58 bit	4 bit	8	9.16
				89	83.56

Tabel 2 Perbandingan antara Huffman Coding dan Arithmatic Coding

Algoritma Shannon-Fano dan Algortima Huffman

Walaupun saat ini sudah bukan lagi proses coding yang menghasilkan kompresi paling optimal namun algoritma Shannon-Fano dan Algoritma Huffman adalah dua algoritma dasar yang sebaiknya dipahami oleh mereka yang mempelajari tentang kompresi data.

1. Algoritma Shannon-Fano

   Teknik coding ini dikembangkan oleh dua orang dalam dua buah proses yang berbeda, yaitu Claude Shannon di Bell Laboratory dan R.M. Fano di MIT, namun karena memiliki kemiripan maka akhirnya teknik ini dinamai dengan mengggabungkan nama keduanya. Pada dasarnya proses coding dengan algoritma ini membutuhkan data akan frekuensi jumlah kemunculan suatu karakter pada sebuah pesan. Tiga prinsip utama yang mendasari algoritma ini adalah:

   a. Simbol yang berbeda memiliki kode yang berbeda
   b. Kode untuk symbol yang sering muncul memiliki jumlah bit yang lebih sedikit dan sebaliknya symbol yang jarang muncul memiliki kode dengan jumlah bit lebih besar.
   c. Walaupun berbeda jumlah bit-nya tetapi kode harus tetap dikodekan secara pasti (tidak ambigu).

   Berikut adalah langka-langkah Algoritma Shannon-Fano

   1. Buatlah tabel yang memuat frekuensi kemunculan dari tiap karakter.
   2. Urutkan berdasar frekuensi tersebut dengan karakter yang frekuensinya paling sering muncul berada di atas dari daftar (descending).
   3. Bagilah 2 tabel tersebut dengan jumlah total frekuensi pada bagian atas mendekati jumlah total frekuensi pada bagian bawah (lihat tabel 3).
   4. Untuk bagian paro atas berikan kode 0 dan pada paro bawah berikan kode
   5. Ulangi langkah 3 dan 4 pada masing-masing paro tadi hingga seluruh symbol selesai dikodekan.

Symbol	Jumlah	
A	15	0
B	7	0
		Pembagian Pertama
C	6	1
D	6	1
E	5	1

Tabel 3 Langkah Pertama Algoritma Shannon Fano

Symbol	Jumlah			
A	15	0	0	
				Pembagian Kedua
B	7	0	1	
				Pembagian Pertama
C	6	1	0	
				Pembagian Ketiga
D	6	1	1	0
				Pembagian Keempat
E	5	1	1	1

Tabel 4 Langkah Awal Algoritma Shannon Fano

Dari proses pengkodean tersebut kita mendapatkan serangkaian kode untuk merepresentasikan kelima simbol A, B, C, D, E sebagai berikut:
A → 00 ; B → 01; C → 10 ; D → 110 ; dan E → 111

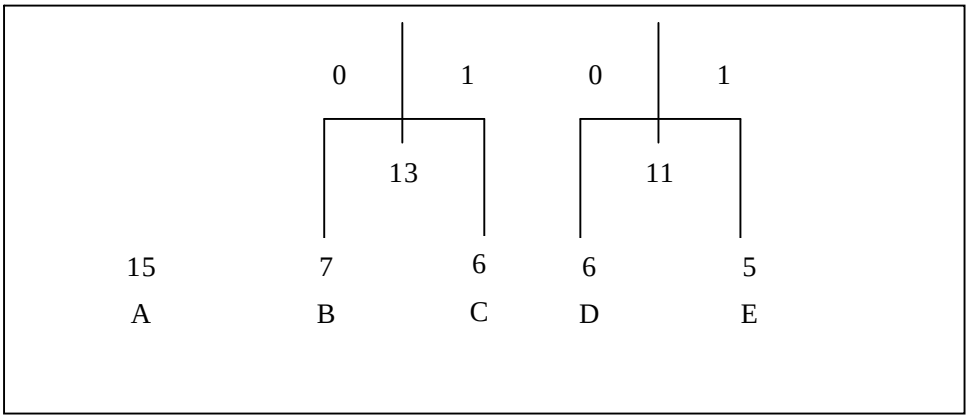
2. Algoritma Huffman

Algoritma Huffman memiliki kemiripan karakteristik dengan Algoritma Shannon-Fano. Masing-masing simbol dikodekan dengan deretan bit secara unik dan simbol yang paling sering muncul mendapatkan jumlah bit yang paling pendek. Perbedaan dengan Shannon-Fano adalah pada proses pengkodean. Jika algoritma Shannon-Fano membangun tree dengan pendekatan top-down, yaitu dengan memberikan bit pada tiap-tiap simbol dan melakukannya secara berurutan hingga seluruh leaf mendapatkan kode bit masing-masing. Sedangkan algoritma Huffman sebaliknyamemberikan kode mulai dari leaf secara berurutan hingga mencapai root.

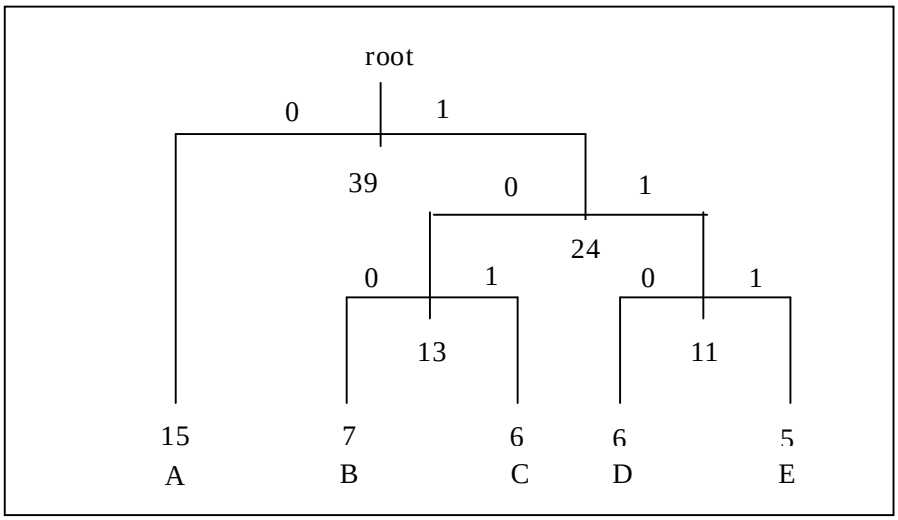
Prosedur untuk membangun tree ini sederhana dan mudah dipahami. Masing-masing simbol diurutkan sesuai frekuensinya, frekuensi ini dianggap sebagai bobot dari tiap simbol, dan kemudian diikuti dengan langkah-langkah sebagai berikut:

1. Dua node bebas dengan bobot terendah dipasangkan.
2. Parent node untuk kedua node pada langkah sebelumnya dibuat. Jumlahkan frekuensi keduanya dan gunakan sebagai bobot.
3. Sekarang parent node berperan sebagai node bebas.
4. Berikan kode 0 untuk node kiri dan 1 untuk node kanan.
5. Ulangi langkah di atas sampai hanya tersisa satu node. Sisa satu node ini lah yang disebut sebagai root.

Untuk lebih jelasnya tentang algoritma Huffman dapat diamati pada gambar berikut ini:



Gambar II Langkah Awal Algoritma Huffman



Gambar III Langkah Akhir Algoritma Huffman

Dari algoritma Huffman didapatkan hasil pengkodean sebagai berikut:
A→ 0; B →100; C → 101; D →110; E →111

Pada kondisi apapun algoritma Huffman akan menghasilkan jumlah bit yang lebih kecil atau sama dengan hasil pengkodean dengan algoritma Shannon-Fano, karena itu Huffman dianggap jauh lebih optimal untuk digunakan pada proses kompresi data. Demikian pula untuk contoh di atas yang hasilnya dapat kita amati pada tabel berikut ini.

Symbol	Jumlah	Kode Shannon Fano	Kode Huffman	Ukuran Shannon-Fano	Jumlah Bit Shannon-Fano	Ukuran Hufman	Jumlah Bit Huffman
A	15	00	0	2	30	1	15
B	7	01	100	2	14	3	21
C	6	10	101	2	12	3	18
D	6	110	110	3	18	3	18
E	5	111	111	3	15	3	15
			TOTAL BIT		89		87

Tabel 5 Perbandingan Algoritma Huffman dan Shannon-Fano

II. Modeling

Jika coding adalah roda dari sebuah mobil maka modeling adalah mesinnya. Sebaik apapun algoritma untuk melakukan coding tanpa model yang baik kompresi data tidak akan pernah terwujud. Kompresi Data Lossless pada umumnya diimplementasikan menggunakan salah satu dari dua tipe modeling, yaitu statistical atau dictionary-based. Statistical-modeling melakukan prosesnya menggunakan probabilitas kemunculan dari sebuah symbol sedangkan dictionary-based menggunakan kode-kode untuk menggantikan sekumpulan symbol.

1. Statistical Modeling

Pada bentuk paling sederhananya, statistical-modeling menggunakan tabel statis yang berisi probabilitas kemunculan suatu karakter atau symbol. Tabel ini pada awalnya bersifat universal, sebagai contoh pada bahasa Inggris karakter yang paling sering muncul adalah huruf “e” maka karakter ini memiliki

probabilitas tertinggi pada file teks yang berbahasa Inggris.

Menggunakan tabel universal pada akhirnya tidak memuaskan para ahli kopresi data karena apabila terjadi perubahan pada subyek yang dikompresi dan tidak sesuai dengan tabel universal maka akan terjadi penurunan rasio kompresi secara signifikan.

Akhirnya muncul modeling dengan menggunakan tabel yang adaptif, di mana tabel tidak lagi bersifat statis tetapi bisa berubah sesuai dengan kode. Pada prinsipnya dengan model ini, sistem melakukan penghitungan atau scan pada keseluruhan data setelah itu barulah membangun tabel probabilitas kemunculan dari tiap karakter atau symbol.

Model ini kemudian dikembangkan lagi menjadi adaptive statistical modeling di mana sistem tidak perlu melakukan scan ke seluruh symbol untuk membangun tabel statistik, tetapi secara adaptif melakukan perubahan tabel pada proses scan karakter per karakter.

2. Dictionary Based Modeling

Jika statistical model pada umumnya melakukan proses encode simbol satu per satu mengikuti siklus: baca karakter → hitung probabilitas → buat kodenya maka dictionary-based modeling menggunakan mekanisme yang berbeda. Dictionary-based modeling membaca input data dan membandingkannya dengan isi dictionary. Jika sekumpulan string sesuai dengan isi dictionary maka indeks dari dictionary entry lah yang dikeluarkan sebagai output dan bukan kodenya.

Sebagai perumpamaan dari dictionary-based dapat digunakan makalah ilmiah sebagai contoh. Saat kita membaca makalah ilmiah kita sering membaca nomor-nomor referensi yang bisa kita cocokkan dengan daftar pustaka di belakang. Hal ini mirip dengan proses pada dictionary-based modeling.

Demikianlah tulisan ini sebagai sebuah upaya mengenalkan subyek Kompresi Data. Sangat banyak yang bisa dipelajari lebih lanjut tentang subyek ini. Pengembangan dan riset lanjutan pun bisa dilakukan apabila kita tertarik untuk mempelajarinya lebih lanjut. Berbagai algoritma untuk proses coding maupun modeling terus bermunculan dan menimbulkan berbagai harapan untuk kompresi data dengan rasio yang lebih optimal dan proses yang lebih cepat.

Referensi:

Nelson, M., Gailly, J.L., The Data Compression Book, Second Edition. M&T Books, New York, 1996

Blelloch, G.E., Introduction to Data Compression. Computer Science Department, Carnegie Mellon University, 2001

Profil Penulis

Putu Ashintya Widhiartha lahir pada bulan Juli tahun 1977, meraih gelar sarjana Komputer dari Teknik Informatika Institut Teknologi Sepuluh November (ITS) Surabaya tahun 2000. Gelar Master of Engineering bidang Computer Science diraih dari Ritsumeikan University Jepang tahun 2006 dengan beasiswa JICA JDS. Profesi utama sejak tahun 2001 hingga saat ini adalah pegawai negeri sipil (PNS) dengan jabatan fungsional sebagai pamong belajar pada kelompok studi teknologi informasi di Balai Pengembangan Pendidikan Nonformal dan Informal (BPPNFI) Regional IV Surabaya. Selain itu juga merangkap sebagai dosen luar biasa pada jurusan Teknik Informatika Universitas Widya Kartika Surabaya, Teknik Informatika Institut Informatika Indonesia Surabaya dan Teknik Komputer Politeknik NSC Surabaya. Pada tahun 2008 menjadi salah satu nominator peraih Indonesia ICT Award (INAICTA) untuk kategori e-education.