Dictionary coder

From Wikipedia, the free encyclopedia

A **dictionary coder**, also sometimes known as a **substitution coder**, is a class of lossless data compression algorithms which operate by searching for matches between the text to be compressed and a set of strings contained in a data structure (called the 'dictionary') maintained by the encoder. When the encoder finds such a match, it substitutes a reference to the string's position in the data structure.

Methods and applications [edit]

Some dictionary coders use a 'static dictionary', one whose full set of strings is determined before coding begins and does not change during the coding process. This approach is most often used when the message or set of messages to be encoded is fixed and large; for instance, an application that stores the contents of a book in the limited storage space of a PDA generally builds a static dictionary from a concordance of the text and then uses that dictionary to compress the verses. This scheme of using Huffman coding to represent indices into a concordance has been called "Huffword".[1]

In a related and more general method, a dictionary is built from redundancy extracted from a data environment (various input streams) which dictionary is then used statically to compress a further input stream. For example, a dictionary is built from old English texts then is used to compress a book.[2]

More common are methods where the dictionary starts in some predetermined state but the contents change during the encoding process, based on the data that has already been encoded. Both the LZ77 and LZ78 algorithms work on this principle. In LZ77, a circular buffer called the "sliding window" holds the last *N* bytes of data processed. This window serves as the dictionary, effectively storing *every* substring that has appeared in the past *N* bytes as dictionary entries. Instead of a single index identifying a dictionary entry, two values are needed: the *length*, indicating the length of the matched text, and the *offset* (also called the *distance*), indicating that the match is found in the sliding window starting *offset* bytes before the current text.

LZ78 uses a more explicit dictionary structure; at the beginning of the encoding process, the dictionary only needs to contain entries for the symbols of the alphabet used in the text to be compressed, but the indexes are numbered so as to leave spaces for many more entries. At each step of the encoding process, the longest entry in the dictionary that matches the text is found, and its index is written to the output; the combination of that entry and the character that followed it in the text is then added to the dictionary as a new entry.

Examples [edit]

Example: The text to be compressed starts with "HURLYBURLY". In the first six steps of the encoding, we output the indexes for H, U, R, L, Y, and B, and we add to the dictionary new entries for HU, UR, RL, LY, YB, and BU. On the seventh step, we are at the start of "URLY"; the longest entry in our dictionary that matches the text is "UR", an entry we added on the second step. We send the index for "UR" to the output, and add an entry for "URL" to the dictionary. On the eighth step, we send the index for "LY" to the output, and assuming that a space follows HURLYBURLY in the text, we add an entry for "LY " to the dictionary. If later in the text we were to encounter the word "HURLYBURLY" again, we could encode it this time (assuming we started at the H) in no more than five indexes:- HU, RL, YB, URL, and Y.

The LZ78 decoder receives each symbol and, if it already has a previous prefix, adds the prefix plus the symbol to its own separate dictionary. It then outputs the symbol and sets the prefix to the last character of the symbol.

Wikipedia The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Deutsch
Français
한국어
Polski
Русский
Edit links

One "gotcha" here is that if the encoder sees a sequence of the form STRING STRING CHARACTER, where STRING is currently in the dictionary, it will output a symbol that is one higher than the decoder's last dictionary entry. The decoder must detect such an event and output the previous symbol plus its first character. This symbol will always be only one higher than the last numbered symbol in the decoder's dictionary.

Encoding HURLYBURLY	
H	H
U	HU
R	UR
L	RL
Y	LY
B	YB
U	BU
R	-- (UR is in the dictionary already)
L	URL
Y	RLY (doesn't matter)

Example: The encoder is encoding BANANANANA; after outputting the indexes for B, A, N and AN the encoder has in its dictionary entries for BA, AN, NA, and ANA and the decoder has entries for BA, AN, and NA. The encoder can match "ANA" so it sends the index for "ANA" and adds "ANAN" to the dictionary. However, the decoder doesn't have "ANA" in its dictionary. It must guess that this new symbol is the prefix (the last symbol it received, "AN") plus its first character ("A"). It then outputs "ANA" and adds the prefix plus the last character of the output ("A" again) to the dictionary. Decoding can continue from there.

Another dictionary coding scheme is byte pair encoding, where a byte that does not appear in the source text is assigned to represent the most commonly appearing two-byte combination. This can be done repeatedly as long as there are bytes that do not appear in the source text, and bytes that are already representing combinations of other bytes can themselves appear in combinations.

References [edit]

1. ^ Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes*. New York: Van Nostrand Reinhold, 1994. ISBN 9780442018634.
2. ^ Rodney J. Smith. *Streaming Compression System Using Dynamic Connection Groups*, US patent 5,748,955, priority date 20 December 1993.

See also [edit]

- Entropy encoding

v · t · e	Data compression methods	[hide]
Lossless	Entropy type	Unary · Arithmetic · Golomb · Huffman (Adaptive · Canonical · Modified) · Range · Shannon · Shannon–Fano · Shannon–Fano–Elias · Tunstall · Universal (Exp-Golomb · Fibonacci · Gamma · Levenshtein)
	Dictionary type	Byte pair encoding · DEFLATE · Lempel–Ziv (LZ77 / LZ78 (LZ1 / LZ2) · LZJB · LZMA · LZO · LZRW · LZS · LZSS · LZW · LZWL · LZX · LZ4 · Statistical)
	Other types	BWT · CTW · Delta · DMC · MTF · PAQ · PPM · RLE
Audio	Concepts	Bit rate (average (ABR) · constant (CBR) · variable (VBR)) · Companding · Convolution · Dynamic range · Latency · Nyquist–Shannon theorem · Sampling · Sound quality · Speech coding · Sub-band coding
	Codec parts	A-law · μ-law · ACELP · ADPCM · CELP · DPCM · Fourier transform · LPC (LAR · LSP) · MDCT · Psychoacoustic model · WLPC
Image	Concepts	Chroma subsampling · Coding tree unit · Color space · Compression artifact · Image resolution · Macroblock · Pixel · PSNR · Quantization · Standard test image
	Methods	Chain code · DCT · EZW · Fractal · KLT · LP · RLE · SPIHT · Wavelet
Video	Concepts	Bit rate (average (ABR) · constant (CBR) · variable (VBR)) · Display resolution · Frame · Frame rate · Frame types · Interlace · Video characteristics · Video quality
	Codec parts	Lapped transform · DCT · Deblocking filter · Motion compensation
Theory	Entropy · Kolmogorov complexity · Lossy · Quantization · Rate–distortion · Redundancy · Timeline of information theory	

Compression formats · Compression software (codecs)

Categories: Lossless compression algorithms