Data Compression - Report

lrfk99

Version 2021.01.18

1 Adaptive Huffman coding

Adaptive Huffman coding, Huffman codes are dynamically recreated for better and faster compression. Algorithm Vitter Dictionary: Symbols are words, coded as 1 or 2 bytes, usually as a preprocessing step.

2 PPM

Prediction by Partial Matching (PPM) is appropriate for a large English tex file since there is a large element of predictability both in terms of the letters which follow the preceding few letters, as well as the words which often follow on from the last few words. This is where PPM excels. It works by using a statistical model, which can be thought of as a table, of how likely a given character is to appear next, given the context of the previous n characters, where n is the length of the context. This model is 'trained' on other English tex files, and stored as a separate json file, to be used by the encoder and decoder. The model is stored using a 3-dimensional dictionary data structure, since this is the fastest way of accessing a large dataset in this situation, but it can be thought of as a 2-dimensional table for the purpose of explaining how it works. The columns of the table are the different context lengths and each entry under the column n contains the context, an n length string of characters that precede a certain character, the character which follows the context, and the number of times we have seen this context followed by the given character. The maximum context length n can be varied depending on the situation, but I am using a maximum context length of 5. Given more time I would further investigate how increasing this maximum up to 10 affects the tradeoff between time taken for the encoding and decoding processes and the level of compression achieved. The encoder uses this statistical model of similar tex files to predict how likely it is that the current character follows the previous n characters. This likelihood is encoded using Arithmetic Coding rather than Huffman Coding, since its codewords encode messages rather than being limited to symbols. There is also no need to work out all of the tags of a given length, meaning greater efficiency.

Prediction by Partial Matching (PPM) Order-n e.g. o0,o1,o2,... - symbols are bytes, modeled by frequency distribution in context of last n bytes order-n, modeled in longest context matched, but dropping to lower orders for byte counts of 0. Since the .tex file will be typeset in English, it is possible to predict the next character, given the previous character(s). As such, a statistical model of english text is appropriate to use. This statistical model can be 'trained' on lots of english text such as Alice in Wonderland, and other .tex documents such as the lecture notes for this module. When using PPM, there are three methods for assigning frequencies to the escape symbol. The most appropriate method for this scenario is Method C since it takes into account the fact that some contexts can be followed by virtually any other character by giving the escape symbol an appropriate count, whilst not reducing the count of other symbols. I considered using Dynamic Markov compression, which uses predictive arithmetic coding similar to prediction by partial matching (PPM), except that the input is predicted one bit at a time rather than one byte at a time

3 LZW

Lempel-Ziv-Welch (LZW) is appropriate for a large tex file since there are likely to be repeated occurrences of both sequences of letters (words) and sequences of words (sentences). This is where LZW excels, since it build up longer and longer sequences of characters in its dictionary.

4 Context mixing

See Further. bits modeled by combining predictions of independent models The cutting edge in lossless compression in terms of compression ratio is achieved by combining the next-symbol predictions of two or more statistical models, producing more accurate predictions than any model individually. Prediction by Partial Matching (PPM) Order-n e.g. o0,o1,o2,... - symbols are bytes, modeled by frequency distribution in context of last n bytes order-n, modeled in longest context matched, but dropping to lower orders for byte counts of 0. Since the .tex file will be typeset in English, it is possible to predict the next character, given the previous character(s). As such, a statistical model of english text is appropriate to use. This statistical model can be 'trained' on lots of english text such as Alice in Wonderland, and other .tex documents such as the lecture notes for this module. When using PPM, there are three methods for assigning frequencies to the escape symbol. The most appropriate method for this scenario is Method C since it takes into account the fact that some contexts can be followed by virtually any other character by giving the escape symbol an appropriate count, whilst not reducing the count of other symbols. I considered using Dynamic Markov compression, which uses predictive arithmetic coding similar to prediction by partial matching (PPM), except that the input is predicted one bit at a time rather than one byte at a time Bits modeled by PPM Explanation

5 Lempel-Ziv-Welch

Symbols are strings. Lempel-Ziv-Welch (LZW) LZ77 - repeated strings are coded by offset and length of previous occurence LZ Welch - repeats are coded as indexes into dynamically built dictionary Reduced Offset LZ - LZW with multiple small dictionaries selected by context LZ predictive - ROLZ with dictionary size of l

6 Burrows-Wheeler Transform (BWT)

The Burrows-Wheeler Transform will rearrange a character string into runs of similar characters. This makes it very easy to use move-to-front transform and then run-length encoding. Since in this scenario we are able to scan the entire file before encoding it, The BWT improves the efficiency of the compression algorithm without storing any extra data other than the position of the first original character. we can use the Burrows-Wheeler transform (BWT) converts a list of symbols into a much more structured list. Order-n e.g. o0,o1,o2,... - symbols are bytes, modeled by frequency distribution in context of last n bytes Symbol Ranking - Order-n, modeled by time since last seen. Burrows-Wheeler Transform - Bytes are sorted by context, then modeled by order-0 Symbol Ranking check this

7 Move-to-front transform

The move-to-front (MTF) transform allows for encoding a stream of bytes, and performs particularly well on sorted byte streams.

8 Run-length encoding

Run-length encoding (RLE) is where the actual compression will happen, after the data has been transformed twice by BWT and MTF.

this is good . . Benchmark
[1]

References

[1] Matt Mahoney. Large text compression benchmark, 2020.