RiceRLE Codex Prototype Documentation

Rice Encoding and Decoding

This section explores Rice encoding, a technique characterized by its adjustable parameter, ***k***. The code utilizes specialized functions: ***rice_encode()*** for encoding messages and ***rice_decode()*** for decoding them. A crucial aspect is verifying the implementation's accuracy. Messages are encoded and decoded using the same ***k*** value to ensure the original message remains unchanged. Additionally, the application is tested for robustness against modified messages with appended zeros or ones at the beginning.

The application also includes helper functions: ***bits_to_bytes()*** and ***bytes_to_bits()***. These functions convert between lists of numbers and strings of bits, and vice versa. They have been rigorously tested to confirm their reliability in data conversion without information loss.

Another valuable feature is the ***is_files_equal()*** function. It performs a bit-by-bit comparison of two files to determine if they are identical. Testing has shown its effectiveness in correctly identifying both matching and distinct files.

Encoding and Decoding Process

The encoding process reads an input file byte by byte. Each byte is Rice encoded, and the encoded messages are combined into a single string. This string is then converted back into bytes and saved to an output file. The decoding process essentially reverses these steps, ensuring accurate retrieval and storage of the decoded messages.

Impact of the k Parameter

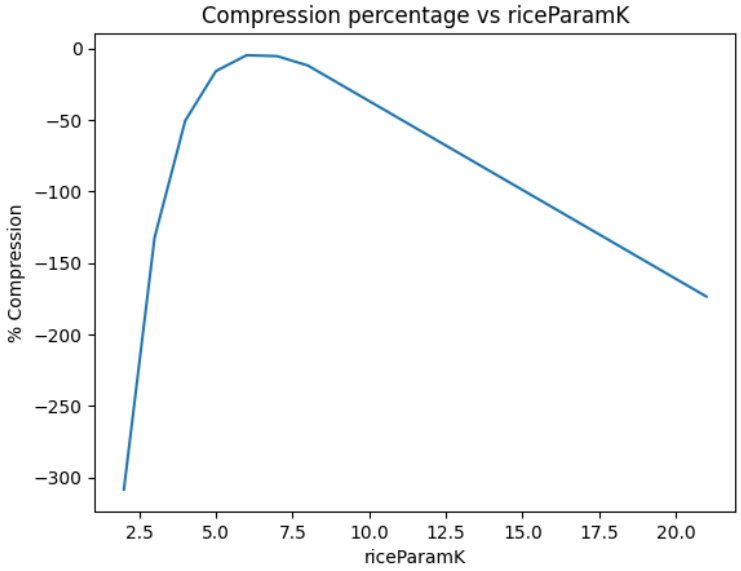
An important observation is that the encoded file size increases with a smaller ***k*** value. For instance, with ***k*** set to 4 bits, the file size grows roughly 1.5 times the original size. At ***k*** equal to 2 bits, the size quadruples. The ***k*** parameter significantly affects how the input data is divided into unary and binary parts within the code words. A larger ***k*** allocates more bits to the binary part. The optimal ***k*** value depends on the statistical properties of the input data, such as the prevalence of zeros and the distribution's variance. Experimenting with various ***k*** values is recommended to find the most suitable parameter for a given dataset, aiming to optimize encoding efficiency as demonstrated in the provided table.

	Original size	Rice (riceParamK = 4 binaryBits)	Rice (riceParamK = 2 binaryBits)	% Compression (riceParamK = 4 binaryBits)	% Compression (riceParamK = 2 binaryBits)
Sound1.wav	1002088	1516265	4115718	-51.3106	-310.714
Sound2.wav	1008044	1575347	4348595	-56.2776	-331.389

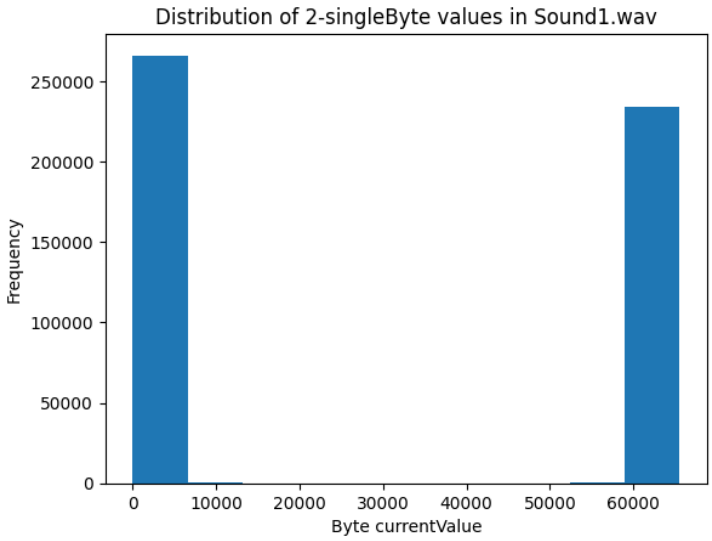
Further development implementation

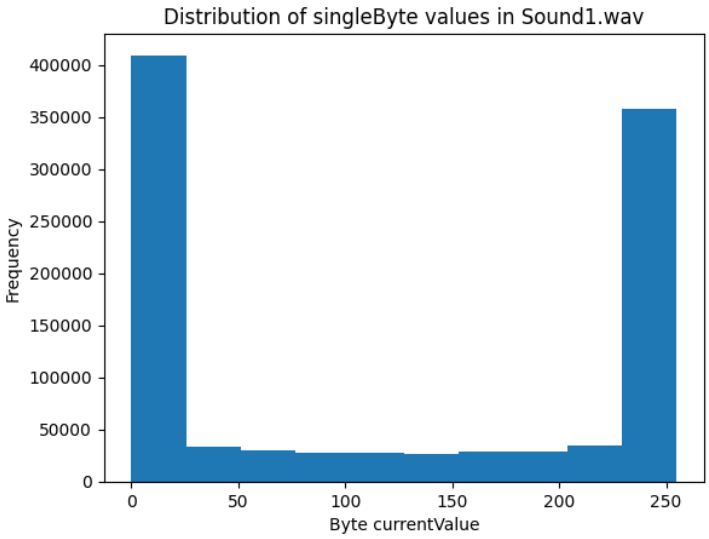
This report investigates the influence of the ***k*** parameter on the compression ratio using Rice coding. The experiment involved adjusting ***k*** values and observing the resulting compression ratios, which are illustrated in a detailed plot.

As observed, the best performance was achieved at ***k=6.*** However, the compression ratio of -4.6% indicates a slight increase in file size after compression. This highlights a limitation of Rice coding, particularly its inefficiency in compressing files without a dominance of smaller values. Rice coding inherently benefits datasets with more frequent occurrences of smaller values. Further analysis revealed that the data distribution resembled the original dataset, characterized by extended sequences of zeros and ones.



A proposed solution to enhance compression efficiency involves combining Rice coding with Run-Length Encoding (RLE). Experiments showed that using RLE alone can double the file size. However, a combined approach utilizing both RLE and Rice coding is suggested to potentially mitigate this issue.





The exploration concludes by examining the standard deviation of the "Sound1.wav" file, which is 112. This significant variance in data values suggests that segmenting the original file could pave the way for optimizing compression algorithm parameters specific to each segment. For Rice coding, this could involve employing different ***k*** values tailored to distinct file chunks, aiming for improved compression efficiency.

	Original size	RLE+Rice Encoded	% Compression
Sound1.wav	1002088	1931642	-92.7617
Sound2.wav	1008044	1953034	-93.7449