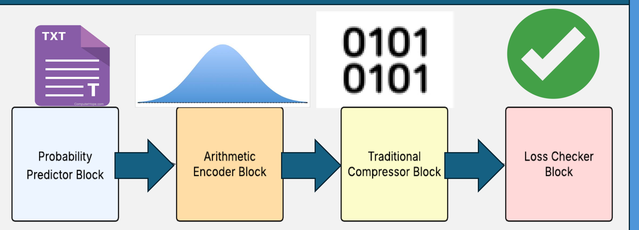
Approaching Entropy Limits with Learned Lossless Compression

Ruchir Kafle
Advisor: Kevin Crowthers, Ph.D.

Graphical Abstract



Background

Data

- Significant amount of data being stored globally.
- 126 zettabytes in 2022 to 284 zettabytes in 2027. (Sun et al., 2025)
- Inefficient and expensive.

Traditional & Learned Compression

- Algorithms that search for patterns in text and condense them. Often scale with file size.
- Learned compression uses neural networks to predict (Sun et al., 2025).
- Often use arithmetic encoders for compression.
- Efficient, as letters are not independent of each other (Mao et al., 2022).

Entropy

- The smallest possible size a piece of data can theoretically be stored within, according to Shannon's information theory (1948).
- Formulas from Shannon (1948) can be used to calculate theoretical entropy of data.
- The goal of every data compressor.

Issues

- Bad semantics and short-term memory in RNNs and transformers. (Goyal et al., 2018)

Methodology

1. Determine model technology stack
2. Proof of concept
3. LLM compressor
4. Full scale model
5. Proof of losslessness & entropy

Research Question

Can a data compressor condense text data down to its entropy limit?

Hypothesis

A hybrid model with a greater memory and semantic understanding of text can approach the entropy limit.

Data compressors continue to reach new lows. Compressors utilizing a Large Language Model probability predictor show significant improvement in compression ratio and file size. These models approach what is theoretically defined by entropy.

Results & Analysis

Uncompressed

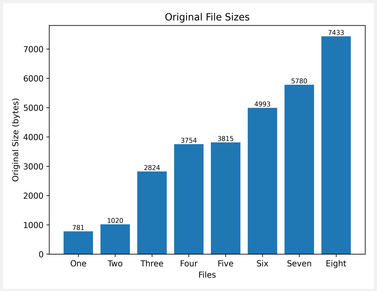


Figure 1: The sizes of the eight files used in testing the model. Each ranged from 781 bytes to 7433 bytes in ascending order, usually by a difference of 1000-2000 bytes.

Post-Compression

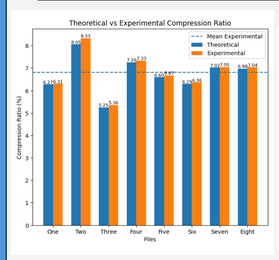


Figure 2: During file compression, a theoretical file size was computed using Shannon's formula. After compression, the experimental file size was found. These metrics were compared to the original file sizes in bytes to find in what percentage the file was able to be represented.

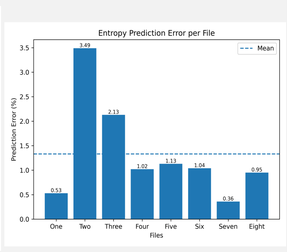


Figure 3: Post-compression, the computed theoretical compressions were compared with the experimental results to find the error. Error, like compression ratio, stayed extremely low. Error at all does show issues with the arithmetic encoder.

Time

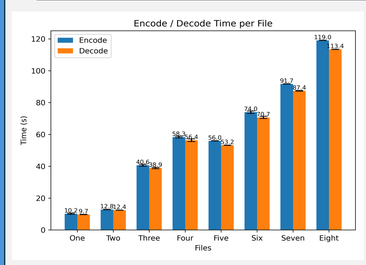
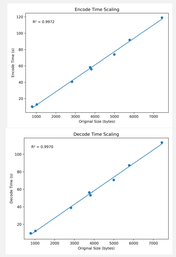


Figure 4: The time it took a file to compress and decompress was measured. Compression and decompression time stayed extremely close to each other, only differing by at worst a few seconds.



Figures 5 & 6: These graphs show seemingly linear correlation between file size and compression or decompression time.

Competitor Analysis

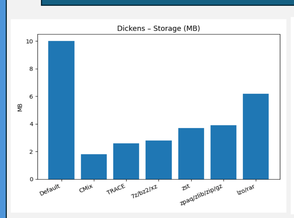


Figure 7: Various traditional and learned compressors were run against dickens, a 10-megabyte text file to find compression ratio. TRACE and dickens were received from Mao et al. (2018), TRACE being a transformer-based compressor. CMIX was received from Byron et al. (n.d.).

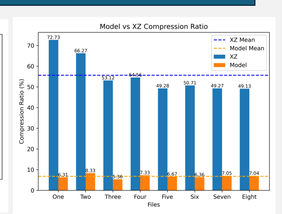
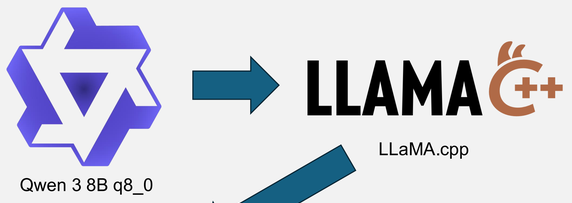


Figure 8: Each trial was run against XZ, the standard Linux data compressor. The large language model-based compressor consistently and significantly outperformed XZ. XZ's compression ratio does decrease as file size grows, suggesting need to compare on larger file scales.

Tech Stack



Conclusions & Applications

Statistical Analysis

- Paired t-test resulted in t-statistic of **15.666**. Resulted in **p < 0.001 (p = 1.045E-06)**.
- Evidence shows the difference between previous compression models and this large language-based model are **statistically significant**.
- Requires further testing as compression ratios of XZ and other compressors are sketchy at low file sizes.

Conclusions

- Compressors utilizing large language models compress files much smaller than other compressors today, approaching entropy.
- Large language model-based compressors do not struggle with compressing small files as traditional compressors do.
- **Greater semantic understandings will allow for greater compression.**

Applications

- Businesses storing abundances of data can free up storage.
- Consumers may be able to use to store more data and transmit it with less bandwidth.

Future Steps

- Finish implementation of full model (add traditional compressors).
- Data collection on larger files.
- LLMs with greater semantics?
- Purely theoretical analysis using Shannon's formula?