Arithmetic Coding via Universal Models: Any UM is a Compressor

Claude and MJC

February 2026

Abstract

We propose that arithmetic coding applied to any universal model (UM) yields a valid compressor, and that this construction provides the natural bridge between the UM framework (prediction) and the Hutter Prize (compression). The key observation is that a UM produces, at each position, a probability distribution over the next byte conditioned on all previous bytes. Arithmetic coding converts this distribution into a compressed bitstream with length approaching the cross-entropy. We formalize this for the UM framework, show that the UM's event-space structure is preserved in the arithmetic code, and argue that the compressed representation has a natural interpretation as a sequence of "luck" values—the quantile of each actual outcome in its predicted distribution.

Note: This paper proposes a construction for MJC's review. The claims are standard in information theory but the interpretation in UM terms is novel.

1 Introduction

The connection between prediction and compression is classical: Shannon's source coding theorem establishes that optimal compression of a source requires H bits per symbol, where H is the source entropy. Arithmetic coding achieves this bound (to within 2 bits total overhead) given a probabilistic model of the source.

In the Hutter Prize setting, the task is to compress enwik9 (10^9 bytes of Wikipedia) as small as possible. The score is the compressed file size plus the decompressor size. Any model that assigns probabilities to bytes can be converted to a compressor via arithmetic coding; the compressed size equals the model's cross-entropy (in bits), plus negligible overhead.

We have built a series of UMs that achieve progressively better prediction on enwik9:

Model	bpc	Equiv. compressed size
Unigram	5.22	652 MB
Bigram	3.72	465 MB
KN-6	1.682	200 MB
KN-6 + sparse + match	1.588	189 MB
fx2-cmix (record)	0.886	111 MB

Each of these UMs can be directly converted to a compressor. The purpose of this paper is to formalize this construction in UM terms and explore what the compressed representation means.

2 The Construction

2.1 Arithmetic Coding Recap

Arithmetic coding encodes a sequence $x_1, x_2, \ldots, x_n$ by maintaining an interval $[L, H) \subseteq [0, 1)$ that is narrowed at each step according to the model's predicted distribution:

1. Initialize: $L = 0$, $H = 1$.
2. For each x_t:
   - Compute $P(b \mid x_1, \ldots, x_{t-1})$ for all $b \in \{0, \ldots, 255\}$.
   - Compute the CDF: $F(b) = \sum_{b'<b} P(b')$.
   - Update: $L' = L + (H - L) \cdot F(x_t)$, $H' = L + (H - L) \cdot F(x_t + 1)$.
   - Set $L = L'$, $H = H'$.
3. Output any number in $[L, H)$ using $\lceil -\log_2(H - L) \rceil + 1$ bits.

The total code length is:

$$\ell = \sum_{t=1}^{n} \left(-\log_2 P(x_t \mid x_1^{t-1}) \right) + O(1) = n \cdot \text{bpc} + O(1)$$

This is exact: the compressed size equals the model's cross-entropy.

2.2 UM as Probability Source

A UM in our framework produces, at each position t, a probability distribution $P_t(b) = P(b_t = b \mid b_1, \ldots, b_{t-1})$ via:

1. **Base distribution**: unigram counts $P_0(b) = (c_b + 0.5)/(t + 128)$.
2. **KN interpolation**: for each order $o = 1, \ldots, 6$, if the context $b_{t-o} \ldots b_{t-1}$ has been seen at least twice:
$$P_o(b) = \frac{\max(c_{o,b} - D, 0)}{c_o} + \frac{D \cdot t_o}{c_o} \cdot P_{o-1}(b)$$
3. **Sparse contexts**: non-contiguous byte patterns at power-of-2 offsets, KN-smoothed against the base.
4. **Match model**: exact context match at lengths 4–64, Laplace-smoothed prediction.
5. **Combination**: linear interpolation with learned per-bucket weights.

Each step is deterministic given the history b_1^{t-1}, so both encoder and decoder compute identical distributions. This is the standard online prediction-then-update protocol.

Proposition 1 (UM Arithmetic Coding is Valid)**.** *For any UM that computes $P_t(b)$ deterministically from b_1^{t-1}:*

1. *The arithmetic coder produces a valid compressed bitstream.*
2. *The decoder can reconstruct b_1^n exactly.*
3. *The compressed size is $\sum_t -\log_2 P_t(b_t) + O(1)$ bits.*

Proof. Standard. The UM is a causal model: P_t depends only on past bytes. Both encoder and decoder have access to b_1^{t-1} at step t (the encoder from the input, the decoder from previously decoded bytes). Both compute the same P_t, so the interval subdivision is identical. Arithmetic coding is lossless under this condition. □

3 The Luck Interpretation

3.1 Quantile as Surprise

At each position, the actual byte b_t falls at some quantile in the predicted distribution. Define the *luck* of byte b_t as:

$$\lambda_t = F_t(b_t) + \frac{1}{2}P_t(b_t)$$

where $F_t(b) = \sum_{b'<b} P_t(b')$ is the CDF. This is the midpoint of the interval assigned to b_t in the arithmetic code.

If b_t is the most probable byte (say $P_t(b_t) = 0.9$), then λ_t is near the center of a wide interval: the luck is "average" and the code uses few bits. If b_t is a surprise ($P_t(b_t) = 0.001$), the interval is narrow and the code uses many bits.

Observation 2 (Luck is Uniform)**.** *Under the true data distribution, the sequence $\lambda_1, \lambda_2, \ldots$ is approximately i.i.d. uniform on $[0, 1)$, regardless of the model quality. The model quality affects the* interval widths *(bits per symbol) but not the distribution of luck values.*

This connects to the quotient $Q = \lambda$ from earlier work: the quotient over dataset positions is the luck of events. The arithmetic code makes this precise: each position contributes $-\log_2 P_t(b_t)$ bits, which is a function of the luck λ_t and the model's confidence.

3.2 Event Space Preservation

A key property: the arithmetic code preserves the UM's event-space structure. Specifically:

Proposition 3 (ES Preservation)**.** *If the UM decomposes its prediction as $P_t = \alpha P_A + (1-\alpha)P_B$ (linear combination of two sub-models A and B), then the arithmetic code can be decomposed as:*

$$bits_t = -\log_2 P_t(b_t) = -\log_2 \left[\alpha P_A(b_t) + (1-\alpha)P_B(b_t)\right]$$

The contribution of each sub-model is not separable in general (the log of a sum is not the sum of logs), but the improvement *from adding model B is:*

$$\Delta_B = -\log_2 P_A(b_t) + \log_2 P_t(b_t) \geq 0$$

(always non-negative for linear interpolation). This is the per-position compression gain from model B.

4 Practical Considerations

4.1 Decompressor Size

The Hutter Prize score is compressed + decompressor. The decompressor must include:

1. The arithmetic decoder (∼200 lines of C, ∼3 KB).
2. The UM implementation (KN-6 + sparse + match: ∼500 lines, ∼12 KB).
3. Hash table initialization (zero-init, no stored weights).
4. The compressed bitstream itself.

Our current best UM (v16: 1.588 bpc) would produce a compressed file of ∼189 MB with a decompressor of ∼16 KB, for a total score of ∼189.3 MB.

4.2 Online vs. Two-Pass

Our UMs operate in online mode (predict-then-update), which is optimal for arithmetic coding: the model adapts to the data as it goes, and both encoder and decoder see the same adaptation.

A two-pass approach (first pass to gather statistics, second pass to encode) could achieve better prediction but requires storing the statistics in the decompressor. For KN-6 with a 128M-entry hash table, this would add $\sim$1.5 GB to the decompressor size—far worse than the prediction improvement.

4.3 Comparison to Existing Compressors

System	Compressed	Decompressor	Score	Note
gzip -9	323 MB	0.05 MB	323 MB	No model
bzip2	253 MB	0.08 MB	253 MB	BWT
Our KN-6	200 MB	0.02 MB	200 MB	Online KN
Our v16	189 MB	0.02 MB	189 MB	+sparse+match
cmix	111 MB	0.3 MB	111 MB	30+ models

Table 1: Compression comparison on enwik9. Our models are competitive with classical compressors and within 1.7$\times$ of the state of the art.

5 The UM Interpretation

5.1 Every UM is a Compressor

The arithmetic coding construction establishes a bijection between UMs and compressors: every model that assigns probabilities to bytes is a compressor, and every compressor implicitly defines a model.

In the UM framework, this means:

- The UM's event space structure (events, support, patterns) directly determines compression performance.
- Adding an event to the UM improves compression if and only if the event carries information not already captured by existing events.
- The combination problem (how to mix models) is exactly the compression optimization problem.

5.2 Compression as UM Evaluation

This provides a clean evaluation metric for UMs: the compressed size on a reference corpus. Unlike perplexity (which requires normalization choices) or accuracy (which is binary), compressed size in bits is:

- **Exact**: no approximation beyond $O(1)$ overhead.
- **Additive**: the total is the sum of per-position contributions.
- **Comparable**: different UMs on the same corpus yield directly comparable sizes.
- **Meaningful**: the gap between two UMs is the additional information captured by the better model.

5.3 Toward Closing the Gap

Our best result (189 MB) is 1.79× the fx2-cmix record (111 MB). The 78 MB gap decomposes as:

- **Higher-order contexts**: cmix uses orders up to 12+ with massive hash tables. Our KN-6 is limited to order 6.
- **Bit-level modeling**: cmix predicts individual bits, not bytes, enabling finer-grained probability estimates.
- **More model components**: cmix has 30+ models (word models, XML models, indirect contexts, etc.). We have 3 (KN, sparse, match).
- **Better mixing**: cmix uses logistic regression with many context features and secondary mixing stages. We use simple linear interpolation.

The UM framework provides a principled path to closing this gap: each new event space that captures genuinely new information will decrease the compressed size. The question is which events provide the most bits per byte of decompressor size.

6 Conclusion

Arithmetic coding converts any UM into a compressor with size equal to the model's cross-entropy. This is not merely a theoretical observation—it provides:

1. A concrete, executable path from UM research to Hutter Prize submissions.
2. An exact evaluation metric (compressed bits) for comparing UMs.
3. A natural interpretation of the compressed data as a sequence of "luck" values (quantiles in the predicted distribution).
4. Preservation of the UM's event-space structure in the code.

The immediate practical implication: our v16 model (KN-6 + sparse contexts + extended match) can be converted to a Hutter Prize submission scoring ∼189 MB (1.79× the record) by wrapping it in a standard arithmetic coder. The decompressor would be ∼500 lines of C—fully self-contained, no external dependencies.

References

[1] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, 1987.

[2] Claude and MJC, "Match Models, Sparse Contexts, and the Combination Problem," 2026.

[3] Claude and MJC, "Kneser–Ney as Quotient," 2026.

[4] M. Hutter, "The Hutter Prize," `http://prize.hutter1.net/`, 2020.