The Sequence as an Integer: Direct Construction from 256^t to Fourier on Words

Claude and MJC

February 2026

Abstract

We extend the $E \to \mathbb{N} \to Q$ bridge from single-position events to entire sequences. A byte sequence $b_1, \ldots, b_n$ encodes directly as the integer $e = \sum_t b_t \cdot 256^{t-1}$, which is simply the sequence read as a base-256 number. The quotient $Q = e/|E|$ with $|E| = 256^n$ places this integer in $[0, 1)$—and this is exactly the interval that arithmetic coding under a uniform model selects. Under a non-uniform model, each step of arithmetic coding narrows the interval, and the extra entropy that arithmetic coding "carries" in its sliding window of precision bits has a direct interpretation as the quotient's accumulated state.

The connection to Kneser–Ney is then explicit: KN's sliding context window is the model's version of the coder's precision window. Both carry forward a bounded amount of state; both discard information beyond their horizon; both accumulate higher-order structure in the quotient space.

We identify the sum view from v2 as a superposition in the λ-quotient event space, where $\lambda = 1$ counts time in byte units (or $\lambda = 8$ for bit-level events). Words form another quotient space at variable frequency. By enforcing fixed word length (padding or accumulating with a reset event), words become periodic—and in speech, rhythm suggests this is what actually happens. Mapping both letter and word phases onto the same unit circle via $\theta = 2\pi Q$ makes Fourier analysis and log-counting under phase natural.

This is v3. v1 established AC via UMs; v2 built the $E \to \mathbb{N} \to Q$ bridge and prime powers; v3 extends E through time and develops the word–letter Fourier connection.

1 The Sequence as an Integer

1.1 Direct Encoding: $e = \sum b_t \cdot 256^{t-1}$

The simplest encoding of a byte sequence is to read it as a number in base 256:

$$e = \sum_{t=1}^{n} b_t \cdot 256^{t-1} = b_1 + 256\, b_2 + 256^2\, b_3 + \cdots + 256^{n-1}\, b_n.$$

This is a bijection between byte sequences of length n and integers in $\{0, 1, \ldots, 256^n - 1\}$. The maximum is $|E| = 256^n$. The quotient:

$$Q = \frac{e}{|E|} = \frac{e}{256^n} = \sum_{t=1}^{n} \frac{b_t}{256^t} \in [0, 1).$$

This is a base-256 "decimal" expansion. The first byte b_1 determines the first base-256 digit (the coarsest $1/256$ interval), the second byte b_2 refines within that interval, and so on.

Proposition 1 (Base-256 = Uniform AC)**.** *The quotient $Q = e/256^n$ is identical to the interval position selected by arithmetic coding under a uniform model ($P(b) = 1/256$ for all b). Each byte narrows the interval by factor $1/256$, and after n bytes the interval has width 256^{-n}, centered at Q.*

Under the uniform model, every sequence of length n uses exactly $8n$ bits. No compression occurs. But the encoding is already interesting: the quotient Q places every possible enwik9 on the unit interval, and the actual enwik9 has a specific address $Q^* \in [0, 1)$.

1.2 For enwik9: a 2.4-Billion-Digit Integer

Enwik9 has $n = 10^9$ bytes. The integer e is:

$$e = \sum_{t=1}^{10^9} b_t \cdot 256^{t-1},$$

an integer with $\lceil 10^9 \cdot \log_{10} 256 \rceil \approx 2.408 \times 10^9$ decimal digits. Its quotient $Q^* = e/256^{10^9}$ has 10^9 base-256 digits, which are simply the bytes of enwik9 read in order.

Under the uniform model, this address requires 8×10^9 bits $= 10^9$ bytes $= 1$ GB to specify. Compression means finding a model under which the actual Q^* falls in a wider interval, so fewer bits suffice.

1.3 Non-Uniform Models: Narrowing Faster

Under a model $\mathcal{M}$ predicting $P_t(b) = P(b \mid b_1^{t-1})$, arithmetic coding narrows the interval not by $1/256$ each step but by $P_t(b_t)$. After n steps, the interval width is:

$$W = \prod_{t=1}^{n} P_t(b_t).$$

The code length is $-\log_2 W = \sum_t -\log_2 P_t(b_t) = n \cdot H_{\text{cross}}$ bits.

The model-weighted quotient Q^P lies in an interval of width W within $[0, 1)$. The better the model, the wider this interval, the fewer bits needed to specify Q^P's position within it.

Observation 2 (Model Quality = Interval Width)**.** *For enwik9 at 1.588 bpc (our v16): $W = 2^{-1.588\times10^9} \approx 10^{-4.78\times10^8}$. The model places the actual sequence in an interval of width 10^{-478M}, which takes 1.588×10^9 bits $\approx$ 189 MB to specify. The fx2-cmix record at 0.886 bpc gives $W = 2^{-8.86\times10^8}$, a wider interval requiring only 111 MB.*

2 The Arithmetic Coder's Sliding Window

2.1 Precision and Carried Entropy

A practical arithmetic coder does not maintain a multi-billion-digit interval. It works in a fixed-precision window (typically 32 or 64 bits), emitting high-order bits as they stabilize and shifting in new precision at the low end. At any moment, the coder's state consists of:

- The bits already emitted (the "frozen" high-order part of Q^P).
- The current window $[L, H)$, a subinterval of width $\sim 2^{-w}$ where w is the window width.
- The pending bits not yet resolved (renormalization carries).

The window is a *sliding view* into the quotient. It carries forward exactly w bits of entropy about the future of the sequence. Past bits have been emitted; future bits are unknown. The window is the coder's working memory.

2.2 The KN Sliding Window Is Exactly This

Kneser–Ney smoothing maintains a context window of k previous bytes (for KN-k). This window is the *model's* sliding view into the sequence. At each position, the model consults the last k bytes to predict the next.

Proposition 3 (Coder Window $\leftrightarrow$ Model Context)**.** *The arithmetic coder's precision window and the model's context window perform dual roles:*

	Coder Window	*Model Context*
Width	*w bits of interval precision*	*k bytes of history*
Carries	*Unresolved code bits*	*Predictive state*
Slides	*As high bits stabilize*	*As new bytes arrive*
Discards	*Emitted bits (frozen)*	*Old context (forgotten)*
Purpose	*Encode Q^P incrementally*	*Predict $P_t(b)$*

Both are finite-state approximations to the full sequence: the coder approximates Q^P with w bits of precision, the model approximates $P(\cdot \mid \text{all history})$ with k bytes of context.

The correspondence is not merely analogical. In KN-k, the context window determines which n-gram counts are consulted. The interpolation weights λ_i at each order are the model's way of distributing the predictive "entropy budget" across context lengths—exactly as the coder distributes its precision bits across the interval.

When KN drops a byte from context (backing off from order k to order $k-1$), it loses specific information but gains robustness (more counts, smoother estimates). This is the same trade-off the coder makes when emitting a bit: specificity is lost (the bit is frozen), but the window is refreshed for future precision.

2.3 Accumulated State in the Quotient

The quotient Q^P accumulates the full history of the sequence: every byte contributes to its value. But the coder's window only sees w bits of this accumulation at once.

The *remainder* beyond the window is the higher-order structure that the model has absorbed into its predictions. For KN-6, this remainder is exactly what the 6-byte context captures: patterns at distances 1–6 are explicit in the context; patterns at distances 7+ are lost.

In the quotient view, the KN-6 context window looks at 6 base-256 digits of Q^P (48 bits of the quotient). The remaining digits of Q^P are the higher-order structure that KN-6 cannot predict—the 0.70 bpc gap to fx2-cmix.

3 The λ-Quotient and Superposition

3.1 λ as the Counting Unit

The v2 paper's sum view $P(E) = \sum_t S_t$ places all positions into a single integer. Each term S_t is a point in event space; the sum is a *superposition* over time.

The quotient parameter λ determines what counts as one "time step." Two natural choices:

- $\lambda = 1$: one byte per step. Time counts in byte units. The sequence has $n = 10^9$ steps for enwik9. Each step narrows the arithmetic coding interval by the byte probability.
- $\lambda = 8$: one bit per step. Time counts in bit units. The sequence has $8n = 8 \times 10^9$ steps. Each step is a binary decision (bit-level arithmetic coding). The event space per step is $\{0, 1\}$ with $|E_{\text{step}}| = 2$.

At $\lambda = 8$, the base-256 encoding $e = \sum b_t \cdot 256^{t-1}$ unfolds to the binary representation $e = \sum_i \text{bit}_i \cdot 2^{i-1}$, which is just the bitwise content of the file. The quotient $Q = e/2^{8n}$ is the file interpreted as a binary fraction. Each bit of the file is literally a bit of Q.

Observation 4 (λ and Granularity)**.** *$\lambda = 1$ (byte) gives 256-ary arithmetic coding. $\lambda = 8$ (bit) gives binary arithmetic coding. $\lambda = \ell_w$ (word length) gives word-level coding. The quotient Q is the same number in all cases; only the granularity of the encoding/decoding steps changes.*

3.2 Superposition in the Quotient Space

The sum $P(E) = \sum_t S_t$ from v2 can now be interpreted as a superposition in the λ-quotient space. Each term S_t is a "pure state" at position t; the sum is a mixture.

At $\lambda = 1$, the superposition has n terms (bytes). At $\lambda = 8$, it has $8n$ terms (bits). The terms interfere constructively (when patterns align) or destructively (when patterns conflict).

The key insight: *different λ values decompose the same quotient Q at different scales.* Byte-level decomposition sees the coarse structure (character frequencies, bigrams). Bit-level decomposition sees the fine structure (bit correlations within bytes). Word-level decomposition sees the semantic structure (word frequencies, word bigrams).

All are projections of the same underlying Q onto different quotient spaces.

4 Words as a Quotient Space

4.1 Variable-Frequency Events

Words do not occur at a fixed byte rate. In English text, word lengths vary from 1 ("a", "I") to 20+ characters. The word-level quotient space Q_w has events at irregular intervals in the byte stream.

This irregularity complicates the Fourier perspective: a signal sampled at variable rate does not have a clean DFT.

4.2 Fixed Word Length: Padding and Reset

One solution: enforce fixed word length ℓ by padding short words and splitting long words. Define a "word clock" that ticks every ℓ bytes. Each tick is a word boundary, and the model resets its word-level state.

Definition 5 (Fixed-Rate Word Quotient)**.** *With fixed word length ℓ, the word-level quotient is:*

$$Q_w = \frac{w}{|W|} \in [0, 1)$$

where w is the word index in a vocabulary W of size $|W|$, and the word clock ticks every ℓ bytes. The byte sequence decomposes as:

$$Q = Q_{w,1} \cdot |W|^{-1} + Q_{w,2} \cdot |W|^{-2} + \cdots$$

where $Q_{w,j}$ is the j-th word's quotient position.

This is exactly analogous to reading the byte sequence in base $|W|$ instead of base 256, but at word granularity. The quotient is the same number $Q \in [0, 1)$; the base changes from 256 to $|W|$.

4.3 Rhythm as Fixed Rate

MJC observes: in speech, rhythm suggests that humans actually do enforce approximately fixed word length. Stressed syllables occur at roughly regular intervals ("stress-timed" languages like English), and word boundaries cluster near these regular beats.

This is the word clock: a periodic reset event that separates word-level processing from letter-level processing. Between beats, the model accumulates letter-by-letter state (spelling the word). At each beat, the accumulated state is captured into a word embedding, and the letter-level state resets.

Remark 6 (The Accumulator Reset)**.** *The embedding conjecture paper describes this cycle: causal capture (spelling → word identity), forward inference (word identity → predictions), reset (clear letter state). The fixed-rate word clock formalizes the reset as a periodic event. In the sat-RNN experiments, the hidden-state norm jumps by* $\|\Delta h\| = 5.31$ *at word boundaries—direct evidence of the reset event.*

The variable word lengths in written text are then a *residual*: the word's actual length deviates from the clock period, and this deviation must be encoded separately. In the arithmetic coding view, the deviation costs extra bits—which is exactly the spelling overhead from the embedding conjecture.

5 Fourier on the Unit Circle

5.1 Letter Phase and Word Phase

With both letter and word quotients mapped to $[0, 1)$, we define two angles on the unit circle:

$$\theta_b(t) = 2\pi \cdot \frac{b_t}{256} \qquad \text{(letter phase at byte position } t\text{)} \tag{1}$$

$$\theta_w(j) = 2\pi \cdot \frac{w_j}{|W|} \qquad \text{(word phase at word position } j\text{)} \tag{2}$$

The letter phase is sampled at the byte rate (n samples). The word phase is sampled at the word rate ($n/\bar{\ell}$ samples, where $\bar{\ell}$ is the mean word length). Under fixed word length ℓ, the word phase is sampled at rate n/ℓ.

5.2 The Joint Phase

The interesting object is not either phase alone but their relationship. At byte position t within the j-th word (position $r = t - j\ell$ within the word, $0 \leq r < \ell$):

- The letter phase $\theta_b(t)$ carries the character identity.
- The word phase $\theta_w(j)$ carries the word identity.
- The intra-word position r modulates the letter phase (different characters are expected at different positions within a word).

The joint phase on the circle is:

$$\Theta(t) = \theta_w(\lfloor t/\ell \rfloor) + \frac{r}{\ell} \cdot 2\pi + \epsilon_b(t)$$

where the first term is the word's base phase, the second is a uniform advance through the word (the "letter clock" within the word), and $\epsilon_b(t)$ is the residual—the letter-by-letter surprise.

5.3 Fourier Decomposition

The Fourier transform of the letter phase sequence $\theta_b(1), \ldots, \theta_b(n)$ decomposes into:

- **DC component:** the mean character value—related to the marginal byte frequency distribution.
- **Low frequencies ($f < 1/\bar{\ell}$):** word-level and sentence-level structure. Peaks at multiples of $1/\bar{\ell}$ correspond to word-periodic patterns.
- **Band around $1/\bar{\ell}$:** the word clock itself. A sharp peak here means regular word boundaries; a broad peak means variable word lengths.
- **High frequencies ($f > 1/\bar{\ell}$):** intra-word letter patterns. The spectrum of English spelling.

Proposition 7 (Word Clock as Carrier Frequency)**.** *In amplitude-modulated (AM) radio, a carrier frequency is modulated by a signal. Analogously, the word clock at frequency $1/\ell$ is a carrier, and the word identities are the modulating signal. The letter-level phases are the full modulated waveform. Demodulation (extracting the word signal from the letter signal) is exactly the word embedding operation.*

5.4 Log-Counting Under Phase

MJC suggests that log-counting under phase connects the Fourier view to compression. The key observation:

The log-probability $-\log_2 P_t(b_t)$ at each position is a *scalar attached to the phase point $\theta_b(t)$* on the circle. The total code length is $\sum_t -\log_2 P_t(b_t)$, which is a sum of scalars weighted by their phase positions.

This is a *phase-weighted sum*—exactly the object that Fourier analysis decomposes. Define:

$$C(f) = \sum_{t=1}^{n} \big(-\log_2 P_t(b_t)\big) \cdot e^{-2\pi i f t/n}$$

This is the DFT of the per-position code lengths. The DC component $C(0)$ is the total code length (the Hutter Prize score). Non-zero frequencies reveal *where in the spectrum the compression is coming from*:

- A peak at frequency $f_w = n/\bar{\ell}$ means the model is doing well at word boundaries (predictable word-level transitions).
- A peak at $f_w/2$ means bi-word patterns (word bigrams) contribute to compression.
- High-frequency structure means letter-level patterns (spelling) contribute.

This spectral decomposition of the code length is a new diagnostic tool: it answers not just "how much compression?" but "at what scales does the compression occur?"

6 The Hierarchy of Quotient Spaces

6.1 Bits, Bytes, Words, Phrases

The quotient $Q = e/|E|$ is a single number, but it admits a hierarchy of decompositions:

Level	Unit	λ	Base
Bits	1 bit	8	2
Bytes	1 byte	1	256
Words	$\bar{\ell}$ bytes	$1/\bar{\ell}$	$\lvert W\rvert$
Phrases	$\bar{\ell}_p$ bytes	$1/\bar{\ell}_p$	$\lvert P\rvert$

Each level reads the same quotient Q in a different base. The byte level reads it in base 256 and sees character patterns. The word level reads it in base $|W|$ and sees word patterns. The phrase level reads it in base $|P|$ and sees phrase patterns.

6.2 The KN Tower as Quotient Tower

KN-k interpolation forms a tower:

$$\text{KN-1} \to \text{KN-2} \to \cdots \to \text{KN-}k$$

where each level conditions on one more byte of context. The KN-quotient paper identified this as a tower of quotient projections.

In the sequence-as-integer view, each KN level reads a different number of base-256 digits of Q. KN-1 reads one digit (the previous byte). KN-6 reads six digits. The interpolation weights λ_i determine how much each digit contributes to the prediction.

The sliding window of KN-k is the model's view into Q: it sees the k most recent digits and must predict the next. Digits further back are "emitted" from the model's perspective —they have influenced the accumulated counts but are no longer directly visible.

6.3 Beyond the KN Window: Word-Level Quotient

The 0.70 bpc gap between our KN-6 and fx2-cmix comes from structure at scales beyond 6 bytes. This is precisely the word-level (and higher) quotient space.

A word like "encyclopedia" is 12 bytes—twice the KN-6 window. KN-6 can predict the end of the word from the last 6 characters ("...opedia" is highly constrained), but it cannot use the first 6 characters to predict the last 6. The word-level quotient space captures this: the word identity (a single event in W) summarizes all 12 characters and predicts what comes next.

The word-level quotient Q_w is higher-order information that KN's sliding window drops. It lives in a separate quotient space, accessible via a separate event space. This is exactly the "context event" from the kn-quotient commentary: a word embedding that conditions the LPP between the byte-level and higher-level event spaces.

7 Construction: from Q to Compression

7.1 The Direct Path

Given enwik9 as an integer e, the compression task is:

1. Compute $Q = e/256^n$ (the sequence as a unit-interval address).
2. Find a model $\mathcal{M}$ such that Q falls in a wide interval of width $W = 2^{-L}$ where L is small.
3. Output $\lceil L \rceil + 1$ bits specifying Q's position in that interval.

This is arithmetic coding, stated in quotient language.

7.2 The Model as a Lens

The model $\mathcal{M}$ is a "lens" that magnifies the relevant structure of Q. A better model gives a wider interval (more magnification, fewer bits).

Different models are lenses of different focal length:

- KN-6: focuses on the last 6 digits of Q. Resolves character n-grams up to order 6.
- Sparse contexts: focuses on digits at offsets 1, 2, 4, 8 (non-contiguous). Resolves patterns that KN misses.
- Match model: focuses on long exact repetitions (many consecutive digits matching a previous occurrence). Resolves copy events.
- Word model: focuses on the word-level rereading of Q in base $|W|$. Resolves word patterns.

Our v16 combination uses three lenses simultaneously, achieving 1.588 bpc. The gap to fx2-cmix is the magnification from additional lenses (bit-level models, longer contexts, word/phrase models, neural components) that our system does not yet include.

8 Conclusion

The sequence-as-integer view makes three connections explicit:

1. Q **is the file.** The quotient $Q = e/256^n$ is the byte sequence read as a base-256 fraction. Arithmetic coding is the process of specifying Q with minimum bits under a model. The coder's precision window and the model's context window are dual sliding views into the same quotient.
2. **Words are a rereading.** The same Q read in base $|W|$ at the word rate gives the word-level quotient Q_w. Enforcing fixed word length (the word clock) makes Q_w periodic and Fourier-analyzable. The word clock corresponds to the rhythmic structure of speech. Variable word lengths in text are a residual cost.
3. **Fourier decomposes compression by scale.** The DFT of per-position code lengths $-\log_2 P_t(b_t)$ separates compression into word-scale, letter-scale, and cross-scale contributions. The DC component is the total score; peaks at the word frequency reveal word-level predictability; peaks at harmonic frequencies reveal phrase-level structure.

The hierarchy of quotient spaces—bits, bytes, words, phrases—is a hierarchy of bases for reading the same number Q. Each base reveals structure invisible at other scales. The KN sliding window reads k digits at the byte scale; adding a word-level quotient space reads the same digits at the word scale. The gap between our current 1.588 bpc and the 0.886 bpc record is, precisely, the higher-scale structure that our models do not yet read.

References

[1] Claude and MJC, "Arithmetic Coding via Universal Models: Any UM is a Compressor," 2026. (v1.)

[2] Claude and MJC, "Arithmetic Coding, Quotients, and Prime Powers: The $E \to \mathbb{N} \to Q$ Bridge," 2026. (v2.)

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

[4] Claude and MJC, "The Embedding Conjecture," 2026.

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

[6] Claude and MJC, "Event Arithmetic: E onto $\mathbb{N}$," 2026.

[7] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Comm. ACM*, 1987.