The Path to the Lexicon

From the Bigram Model to Word Events

Claude and MJC — February 24, 2026

1. The Orthographic Model

We have a byte-level model — bigram, trigram, or deeper — that predicts bytes within word spans. The extended shift chain (WM paper) gives within-word context at multiple depths:

```
ES byte_output 256 OUTPUT
ES byte_prev   256
ES byte_prev2  256
SHIFT byte_output byte_prev
SHIFT byte_prev  byte_prev2
LPP byte_prev  byte_output
LPP byte_prev2 byte_output
```

On reset (space, newline), the deeper ESs clear. This model predicts bytes within words. We freeze it and treat it as a **decoder**: given a sequence of bytes, it assigns probabilities. Conjunction patterns and the timing resolution handle marginal dominance between depths.

2. The Memory Trace

The memory trace is the compressed residual: the information that the model doesn't predict. It is stored as an **arithmetic-coded bitstream**.

At each byte position, the model provides a probability distribution over possible next events. The actual event is encoded against that distribution via arithmetic coding. The file length in bits equals the total encoding cost.

The trace is **lossless**: the frozen model plus the trace file recovers the entire input. Under the null model (base 256, no predictions), the trace is the raw file. Under a bigram model at $\sim$2 bpc, the trace is $\sim$250 MB for enwik9.

2.1. The Trace as a Sequence of Events

The trace is logically a sequence of events, one per position. Every byte of the input is covered. At most positions, the event is a byte (from the byte ES, base 256). Later, when word events exist, a single word event can cover an entire word span — the decoder knows the spelling, so the individual bytes are determined.

The key constraint: every event in the trace must **unambiguously decode to output bytes**. A word event "the" decodes to `t`, `h`, `e`, space. A byte event decodes to itself. The trace is a flat list of events, some in the byte ES, some in a word ES. The decoder interprets each event and produces the output stream.

3. Tick-Tock

The input (enwik9) is read **once**. After that, everything is replay from trace files.

1. **Tick 0**: train the byte-level model on enwik9. Write the memory trace — an AC-encoded bitstream under the byte model's predictions. Freeze the model.
2. **Tock 1**: extend the model. Add a word-level ES, add LPPs from word events to byte output. The frozen byte-level weights are preserved; new structure is added on top.
3. **Tick 1**: replay the trace. The frozen byte model decodes the trace (AC decode $\rightarrow$ original bytes). The extended model processes those bytes and learns — discovering word events from the data. Then the extended model writes a **new trace**: re-encoding the same bytes under its (better) predictions. The new trace is shorter.

Each cycle: decode the old trace with the frozen model $\rightarrow$ bytes $\rightarrow$ encode under the extended model $\rightarrow$ new shorter trace. If the new trace is not shorter, we stop and reconsider.

4. Discovering the Lexicon

The lexicon is a set of word events — common enough to matter, projectable onto the orthographic decoder, that save encoding cost when used.

4.1. What a Word Event Is

A word event is a compound event: the conjunction of byte events observed during a word span, as seen through the orthographic model's shift chain. At the moment of reset after "the", the shift chain holds $E_{-3}[\texttt{t}] \wedge E_{-2}[\texttt{h}] \wedge E_{-1}[\texttt{e}]$. This conjunction IS the word "the" — its identity as seen through the model's structure.

4.2. How the Organism Discovers Words

During Tick 1 (replay), the extended model processes bytes and observes word boundaries (resets). At each reset, the shift chain holds a compound event. If the same compound event recurs frequently, threshold creation (§7 of the Spec) reifies it as a neuron in the word-level ES.

The organism does not count words externally. It processes bytes, maintains the shift chain, and discovers recurrent patterns. The word "the" emerges because $E_{-3}[\texttt{t}] \wedge E_{-2}[\texttt{h}] \wedge E_{-1}[\texttt{e}]$ at reset recurs $\sim$2 million times — far above any reasonable threshold.

4.3. The Shift Chain Depth Limit

A depth-k shift chain identifies words by their last k bytes. Depth 3 captures "the", "and", "for" exactly. For longer words, the suffix is often diagnostic: "tion" identifies a class of words even without the prefix.

As the model grows (deeper shift chains, more data), the lexicon grows to include longer words. But the most frequent short words — which carry the most predictive value for higher-level patterns — are captured first.

5. The Word-Level Trace

Once the lexicon exists, the trace changes character. At word boundaries where the compound event matches a known word event:

- Emit the **word event**. One event covers the entire word span. The decoder knows the spelling, so the per-byte costs within the word are zero or near-zero.

At word boundaries where the compound event is unknown:

- Emit an **escape** (no word event), then encode the word byte-by-byte using the orthographic model.

The word-level trace is shorter than the byte-level trace because common words are encoded as single events rather than spelled out byte by byte. The trace is still lossless: every event decodes unambiguously to bytes.

5.1. The Encoding Test

The word-level trace has cost:

$$\sum_{\text{words}} \begin{cases} -\log_2 p(\text{word event} \mid \text{context}) & \text{if known word} \\ -\log_2 p(\text{escape}) + \sum_i -\log_2 p(b_i \mid \text{byte context}) & \text{if unknown} \end{cases}$$

where the word-level probabilities $p(\text{word event} \mid \text{context})$ come from word-level LPPs (which word follows which). These LPPs are learned during Tick 1 replay.

If this total is less than the byte-only trace, the lexicon has captured real structure. This is the test. Compression falls out of understanding: the model that understands words produces a shorter trace.

6. What We Need to Build

1. **AC encoder/decoder** in the runner, so we can write and replay trace files.
2. **Extended shift chain** with word-boundary clearing (depth 3+ model, described as SN).
3. **Trace recording**: at each position, encode the actual event against the model's distribution. Write the AC bitstream to a file.
4. **Threshold creation at reset**: when the shift-chain conjunction at a word boundary recurs above threshold, create a neuron in the word ES.

5. **Word-level LPPs**: patterns from the word ES to the byte output (spelling) and from word ES to word ES (word sequences).
6. **Replay**: decode the old trace with the frozen model, re-encode with the extended model, measure whether the new trace is shorter.

The AC encoder is the critical infrastructure. Without it, we cannot write traces, and without traces, we cannot do Tick-Tock. This is the next concrete step.