The Model Within the Model: Commentary on the Extended Event Space

Claude and MJC

February 15, 2026

Abstract

We develop the self-similar structure implicit in the extended event space [1]. The UM is a model of an organism of organisms: the human brain contains earlier animal brains, and so on. When we extend H with lexical events, we hold the outer I and O fixed but introduce an *inner* model U_v between bytes and words, with its own I', H', O'—all of which are part of H from the outer perspective. The embedding E_v sits at the heart of this inner model, inducing a bijection used at both the input and output sides. We clarify the role of letter-accumulator events (seemingly redundant but necessary for representing misspellings), the log-stochastic and witness implementation forms, and explain why tokenization is an "attractive nuisance" that discards information. This commentary accompanies [1]; the source notes appear as Appendix A.

1 The Self-Similar Architecture

Both of the following are true:

1. We hold I and O fixed.
2. We extend I and O with new input and output events.

The resolution is that the UM is self-similar at every scale. A UM is a model of an organism of organisms.

Specifically, we hold the *outer* I and O fixed (bytes in, bytes out). The H we extend, but we consider the "inner" H to still be fixed. We add in between a new I' and O' for the inner H, but these are all part of H from the perspective of the larger organism.

Definition 1 (Nested model)**.** *Given a UM with $E = I \times H \times O$, a* nested extension *introduces an inner model U_v such that:*

$$H_{extended} = I' \times H_{inner} \times O', \tag{1}$$

where I' and O' are the inner input and output event spaces (visible only within H from the outer perspective), and H_{inner} is the original hidden space. The outer I and O remain fixed.

This nesting continues at every scale as we grow the architecture. A model factors into models:

$$H' = I_1 \times H_1 \times O_1 = I_1 \times (I_2 \times H_2 \times O_2) \times O_1 = \cdots \tag{2}$$

Remark 2 (Self-representation)**.** *A model U can be represented in a model of itself. If this self-model adds no capabilities, the model simply represents itself. However, if (for example) a UM runner adds additional H state, then an extension U' represents U enriched with dynamic runtime information. There is also a model of a* representation *of a model: our concrete representation of the UM via SN induces a description of any other, necessarily smaller, model (having $E = A^\ell$, etc.).*

2 E_v at the Heart

Equation 1 of [1] shows that E_v must be at the heart of the model—it must be in H, and it must be at the center.

The key observation is that we want a log product pattern (LPP) over byte sequences *and* the lexicon simultaneously. Once we see this, several things follow immediately.

Proposition 3 (E_v bijection)**.** *The embedding E_v induces a bijection between the byte-level and word-level representations. This bijection is useful at both the I and O sides:*

- *At I: bytes accumulate into word evidence (recognition).*
- *At O: word identity distributes into predicted bytes (prediction).*

Write u_v for the inner model (shifting from E synecdochically to u, with u_v representing both the English lexicon and its orthography). Then u_v takes I into I' (which is part of H from the outer perspective, but is the extended input seen by the new inner model H') and also takes O' into O.

By IO symmetry of the LPP, both I and O projections are given by the log counting function ω_0 in the efficient log-stochastic implementation.

Remark 4 (Implementation forms)**.** *Only two forms should be used:*

1. ***Log-stochastic form****: the standard ω_0 counting gives log-probabilities directly.*
2. ***Witness form****: when precise counts are desired, the exact count beyond the floor log count is given by the length of a dataset of memory traces.*

3 Why Letter Events Are Not Redundant

A natural objection: "The letter 'e' has appeared in the current word" seems redundant, because we are building a word embedding. Isn't this information implicit in the accumulating log support for the word?

The answer is no, and the reason is critical. We want to not only *recognize* words but also *represent* events like "teh", "thh", and so on ad infinitum. This is why the extended event space becomes necessary:

- The word embedding carries the word identity ("the").
- The letter accumulator carries the *actual* letters observed.
- The discrepancy between these two is the spelling variant, encoded by its luck $\lambda = 1/P(\text{variant})$.

This richer embedding solves the strawberry problem (counting letters in words) in provable information-theoretic minimum: the letter events are explicit, not implicit in a black-box embedding.

A useful side effect is that this architecture can predict typos: the model has explicit access to which letters have appeared and in what position, making misspelling patterns first-class events.

4 The Word-Start Carrier

The word-start fundamental frequency can be implemented via SN programming with absolute (strength 255) patterns:

1. Word position starts at 1 (this is a joint event with the first input event; it would be at 0 before that input arrives) at strength 255.
2. Implement via SN absolute patterns the obvious ring modulus: position increments with each byte, resets to 1 at word boundaries.
3. The position event is deterministic (strength 255)—a UM runner is free to short-circuit it as pure logic.

This is the explicit version of the carrier signal that the RNN implements implicitly through W_h rotation [1, Section 6].

5 The Tokenization Nuisance Attractor

An *attractive nuisance* is something that does not work but causes people to keep trying it by seeming likely to work—the chess equivalent of a false checkmate.

Tokenization is this attractive nuisance for language modeling. The attractor draws researchers in because tokenized models seem simpler (fewer events per sequence), but the fundamental problem is that **tokenization throws away information**.

In the extended model U', this information loss becomes demonstrable. The byte-level letter events, position events, and spelling-variant events are all discarded by any fixed tokenization. The "teh" $\rightarrow$ "the" mapping is invisible to a tokenizer that maps both to the same token.

The extended event space avoids this by maintaining bytes as the outer I and O while building the lexicon *inside* H. No information is discarded; the word level is an additional structure, not a replacement for the byte level.

6 Updating the Perspective

The original paper's Remark 5 is correct in placing events in I, but needs updating with the inner/outer perspective. Specifically, we need to split the perspective into $U \times U$:

- The *outer* U sees bytes in and bytes out, with the entire lexical machinery inside H.
- The *inner* U_v sees letter events in, word events in H_{inner}, and predicted letters out.

The "model within a model" structure means that U_v is itself a complete model. The UM algebra (composition under multiplication, the ω chain, memory traces) tells us how to connect the inner and outer models.

Furthermore, this decomposition clarifies the relationship between the "multiplication" (Section 5 of [1], $H' = I_{\text{inner}} \times H_{\text{inner}} \times O_{\text{inner}}$) and the "composition" (f composes under multiplication)—they are the same operation viewed from different scales.

A Source: MJC Commentary (20260215)

The following is the original commentary that motivated this paper, reproduced verbatim.

Commentary on The Extended Event Space (v1)

20260215 MJC

We've been doing the [?] thing for a while for references and it's time to figure out what we are going to do other than that. My opinion: we can use numbers for references within our track of work, and keep it with name, year for references to everyone else's work.

What we are extending. Both are true: (1) We hold I and O fixed. (2) We extend I and O with new input and output events. The resolution is that the UM is self-similar at every scale. A UM is a model of an organism of organisms. The human brain contains earlier animal brains, and so on.

Specifically we hold the outer I and O fixed. The H we extend, but we consider the "inner" H to still be fixed. We add in between a new I and O (for the inner H) but these are all part of H from the perspective of the larger organism / model / brain.

In conclusion: Both things are true, we have a model within a model, and this will continue to happen at every scale as we grow the architecture.

Section 2. Eq. 1 shows that the E_v must be at the heart of the model, it must be in H and it probably must be at the center of the model. But we have to dig a little deeper and see that we want an LPP over byte sequences and the lexicon. Once we see this we see immediately that: E_v induces a bijection, hence it becomes useful at both I and O sides. Outer I and O remain fixed. We use u_v (shifting from E, synecdochically, to u) to take I into I' which is part of H from the outer perspective, but is the extended input seen by the new inner model H', but also to take O' into O; by IO symmetry of the LPP we have that I and O are both given by the log counting function ω_0 in the efficient log-stochastic implementation. We note further that only the log-stochastic form or, if precise counts are desired, the witness form should be used. We can implement the word-start fundamental frequency also with SN programming. "The letter 'e' has appeared in the current word." seems to be redundant, because we are building a word embedding, so isn't this implicit in the accumulating log support for the word? However, the point is that we want to not only recognize but also, critically, represent events like "teh" and "thh" and so on ad infinitum; this is why this event space becomes necessary (and gives our richer embedding that will solve the strawberry problem in provable information-theoretic minimum). Side effect of all this is that we can predict typos. So most of Remark 5 is right in some way, but needs to be updated with above understanding. In particular things being in I is right and explained rightly but we need to split the perspective into $U \times U$.

Section 5 (H'). Note that H' now includes $H' = I_{\text{inner}} \times H_{\text{inner}} \times O_{\text{inner}}$ or we could just index them as i and say a model factors into models. Obviously we have already shown that f composes under multiplication, etc.

Section 4. Some commentary on I'. Remark 5 describes the tokenization nuisance attractor. An "attractive nuisance" is something that does not work but causes people to keep trying it by seeming likely to work. The chess equivalent of a false checkmate. The attractor is a nuisance because it throws away information, which we will be able to demonstrate in u'.

References

[1] Claude and MJC. *The Extended Event Space: Injecting Lexical Structure into H.* Hutter archive, 15 Feb 2026. `extended-es.pdf`