Lexemes as Binary Event Spaces: From Atomic Patterns to Bag-of-Letters Prediction

Claude and MJC

February 14, 2026

Abstract

We show how to introduce lexeme-level prediction into the isomorphic Universal Model without defining any new finite automata. The model's existing atomic patterns—the data-derived skip-bigrams over individual bytes—already carry the influence of each letter separately on the prediction of upcoming word boundaries. A word like "the" is not recognized by a dedicated detector but is *implicit* in the joint support from `t`, `h`, and `e` at their respective offsets. This gives a natural "bag of letters" model of words, which we note matches findings from human reading research (transposed-letter effects, etc.). We formalize each lexeme as a binary event space—log support for "the" vs. its negation (which includes "there", "them", "then", etc.)—and show that this binary ES is the correct shape: input bytes map to a *probability* of the lexeme, not a certainty. The negative event distributes over the rest of the lexeme's prefix class, making this a clean case study for negative event distribution. Finally, we observe that the map from input bytes to support over lexemes is itself a complete model U, and the UM algebra tells us how to compose it with the existing byte-level model.

1 Introduction

The tock step [2] discovers new event spaces from data statistics. The tock protocol [3] proposed injecting words one at a time via dedicated finite automata—word detectors that scan the byte stream and fire deterministically.

MJC's key observation: we don't need to define new FSAs. The model's *existing* atomic patterns already encode the influence of each letter. When the model has seen `t` at offset 3, `h` at offset 2, and `e` at offset 1, these three independent byte-level observations jointly support the prediction that the current position follows the word "the." No word detector is needed. The word is implicit in the bag of letters.

This paper develops this observation into a formal construction.

2 Atomic Patterns Already Match Letters

Definition 1 (Atomic byte patterns)**.** *An* atomic byte pattern *is a skip-bigram: a pair* (x_d, o) *where* x_d *is the byte at offset* d *from the current position and* o *is the output byte. The pattern's support is the log count ratio:*

$$T(o \mid x_d) = \log \frac{P(o \mid x_d)}{P(o)}, \tag{1}$$

the log-likelihood ratio of the output byte given the input byte at offset d*.*

The isomorphic UM already has these patterns for all offsets $d = 1, 2, \ldots, k$. The factor map [4] showed that the sat-rnn's 128 neurons implement conjunction detectors over pairs of offsets, with the dominant pair $(1, 7)$ used by 52/128 neurons.

Remark 2 (What the patterns already know about "the")**.** *Consider predicting the byte after "the " (the space after "the"). The model sees:*

Offset	*Byte*	*Pattern*
$d = 1$	`e` *(101)*	$T(\texttt{' '} \mid \texttt{e}_{d=1})$
$d = 2$	`h` *(104)*	$T(\texttt{' '} \mid \texttt{h}_{d=2})$
$d = 3$	`t` *(116)*	$T(\texttt{' '} \mid \texttt{t}_{d=3})$
$d = 4$	`' '` *(32)*	$T(\texttt{' '} \mid \texttt{' '}_{d=4})$

Each of these is an independent atomic pattern. None of them "knows" the word is "the"—each only knows what one byte at one offset contributes to the output prediction.

But jointly, $\texttt{t}_{d=3}$, $\texttt{h}_{d=2}$, $\texttt{e}_{d=1}$, *and* $\texttt{' '}_{d=4}$ *are exactly the signature of "the " occurring. The word is latent in the bag of letters.*

3 The Bag-of-Letters Model

Definition 3 (Bag-of-letters representation)**.** *For a word $w = c_1 c_2 \cdots c_\ell$ of length ℓ, its* bag-of-letters representation *at position t (where position t is the byte* after *w) is the multiset:*

$$B_w(t) = \{(c_1, d{=}\ell), (c_2, d{=}\ell{-}1), \ldots, (c_\ell, d{=}1)\}, \tag{2}$$

i.e., each letter of w paired with its offset from the current position.

Proposition 4 (Bag-of-letters is sufficient for short words)**.** *For short words ($\ell \leq 4$), the bag-of-letters representation is almost always sufficient to identify the word uniquely, given that position within the byte stream is known. The reason: English has very few short anagrams that also share letter positions (e.g., "the" has no rearrangement that is also a common word at offsets $3, 2, 1$).*

For longer words, the bag-of-letters becomes ambiguous ("conservation" vs. "conversation" share the same bag). But even in these cases, the individual letter contributions carry most of the predictive information.

Remark 5 (Human perception works this way)**.** *The psycholinguistic literature on transposed-letter effects shows that human word recognition is largely insensitive to internal letter order. The classic demonstration: "Aoccdrnig to rscheearch, the huamn mnid can raed txet wtih srcmabled ltetres." Readers can parse this because word recognition depends primarily on:*

1. *The first and last letters (position anchors),*
2. *The set of internal letters (the "bag"),*
3. *Word length,*
4. *Context (preceding words).*

The atomic byte patterns in the UM capture exactly (1)–(3): the offset encodes position, the byte value encodes letter identity, and the number of contributing offsets encodes length. It is notable that the same "bag-of-letters" structure falls out of both human perception and the RNN's learned patterns.

4 The Lexeme as Binary Event Space

The bag-of-letters representation gives us a *graded* support for each word. We now formalize this as a binary event space.

Definition 6 (Lexeme ES)**.** *For a lexeme w, define the binary event space $E_w = \{w^+, w^-\}$ where:*

- *w^+ = "w occurred" (the current position is at or just after an occurrence of w),*
- *w^- = "w did not occur" (the negation: everything that is* not *w, including "there", "them", "then", etc. for w = "the").*

Proposition 7 (Support from atomic patterns)**.** *The log support for w^+ given the current context is the sum of individual letter contributions:*

$$T(w^+) = \sum_{i=1}^{\ell} T(c_i \text{ at offset } \ell - i + 1) + T_{boundary}, \tag{3}$$

where $T(c_i$ at offset $d)$ is the log-likelihood that the byte at offset d is c_i (computable from the data), and $T_{boundary}$ accounts for word-boundary evidence (space before, space/punctuation after).

This is an approximation: it assumes independence between the letter observations. The actual support should account for correlations (e.g., `t` *at offset 3 and* `h` *at offset 2 are not independent—"th" is a common bigram). The conjunction patterns from the factor map [4] capture exactly these correlations.*

Remark 8 (Probability, not certainty)**.** *The support $T(w^+)$ is a* log probability*, not a binary decision. Even when all letters of "the" are present at the right offsets, $T(w^+)$ may not be overwhelming: the evidence is also consistent with "them", "then", "there", "these", "their", "they", etc.*

The correct shape is: input bytes (which happen to be deterministic at each position) map to a probability distribution *over lexemes, not a certainty. This is fundamental: the model should maintain uncertainty about which word is occurring until enough evidence accumulates.*

Definition 9 (Negative event distribution)**.** *The negative event w^- is not a single event but a* distribution *over alternative lexemes. For w = "the", the negative event decomposes as:*

$$P(w^-) = P(\text{"there"}) + P(\text{"them"}) + P(\text{"then"}) + P(\text{"these"}) + \cdots + P(\text{not "the"-prefixed}). \tag{4}$$

The "the"-prefixed portion of w^- distributes over the rest of the lexicon that shares the prefix "the". This makes "the" a particularly clean case study for negative event distribution: the negative event has a natural decomposition by prefix extension.

Example 10 (Support calculation for "the")**.** *At a position where the preceding bytes are "...* `of the` *" (i.e., we are predicting the byte after the space after "the"):*

$$T(\text{"the"}^+) = T(\text{' '}_{d=4}) + T(\texttt{t}_{d=3}) + T(\texttt{h}_{d=2}) + T(\texttt{e}_{d=1}) + T(\text{' '}_{d=0}) \tag{5}$$

$$+ \text{ conjunction corrections.} \tag{6}$$

The conjunction corrections come from the 2-offset patterns: e.g., $T(\texttt{t}_{d=3}, \texttt{h}_{d=2})$ contributes additional support from the "th" bigram at the right position.

The support for the negative event:

$$T(\text{"the"}^-) = \log\big(1 - \exp(T(\text{"the"}^+))\big) \tag{7}$$

as the standard negation of the binary ES [1].

5 The Lexeme Predictor as Atomic Input

A crucial observation: the "the" predictor does not wait until all letters are seen before producing output. It has positive support even after seeing just `t`.

Proposition 11 (Graded support at every position)**.** *The lexeme support $T(w^+)$ is well-defined at every byte position, not just at the end of the word. After seeing* `t`*:*

$$T(\text{“the”}^+ \mid \texttt{t}_{d=1}) > 0, \tag{8}$$

because `t` *is the first letter of "the" and observing it raises the probability that "the" is occurring. After seeing* `th`*:*

$$T(\text{“the”}^+ \mid \texttt{t}_{d=2}, \texttt{h}_{d=1}) > T(\text{“the”}^+ \mid \texttt{t}_{d=1}), \tag{9}$$

and so on: the support accumulates as more letters arrive, reaching its maximum when all letters plus a word boundary have been seen.

The support can also decrease: after "the" if the next byte is `r` *(heading toward "there"), $T(\text{“the”}^+)$ drops and $T(\text{“there”}^+)$ rises.*

Remark 12 (Just another atomic input)**.** *From the perspective of the rest of the model, the graded support $T(\text{“the”}^+)$ at each position is* just another atomic input *—exactly like a byte observation at a particular offset. The byte-level model sees:*

Input	*Type*
$\texttt{e}_{d=1}$	*byte at offset 1*
$\texttt{h}_{d=2}$	*byte at offset 2*
$\texttt{t}_{d=3}$	*byte at offset 3*
$T(\text{“the”}^+)$	*lexeme support (graded)*

The lexeme support is just another feature that the model can condition on. It happens to be computed from a conjunction of byte observations, but from the model's perspective it is an opaque scalar input, no different from any other.

This is the compositionality of the UM: the output of one model (U_{lex}: bytes → lexeme support) becomes an input to another (U_{byte}: bytes + lexeme support → output byte). The composed model's event space is enriched by the lexeme dimension, but its forward pass is unchanged: each input (whether a raw byte or a lexeme support value) contributes to the output through the same pattern-matching mechanism.

Proposition 13 (Recursive composability)**.** *If lexeme support is just another atomic input, then:*

1. *Multiple lexeme predictors can run in parallel, each producing a graded support signal. $T(\text{“the”}^+)$, $T(\text{“of”}^+)$, $T(\text{“and”}^+)$, ... all feed into the byte-level model as additional atomic inputs.*

2. *Higher-order predictors can condition on lexeme support: $T(\text{“of the”}^+)$ could be computed from $T(\text{“of”}^+)$ and $T(\text{“the”}^+)$ as inputs. The phrase is a conjunction of lexeme events, exactly as the lexeme was a conjunction of letter events.*

3. *The recursion bottoms out at bytes and tops out wherever the data stops supporting further conjunctions. The factorization tower [2] is built from the bottom up, each level providing atomic inputs to the next.*

6 Subtracting Lexeme Contributions

Proposition 14 (Marginals are downstream of lexemes)**.** *The marginal byte distributions—$P(\texttt{t})$, $P(\texttt{h})$, $P(\texttt{e})$, $P(\texttt{' '})$—are* causally downstream *of the marginal lexeme distribution. The reason: the bytes that appear in the data are there* because *certain words were used. The frequency of the bigram "th" is high partly because "the" is the most common English word.*

This causal direction means: to understand the byte-level Markov statistics, we should factor out the lexeme contributions. The "residual" byte statistics (after subtracting lexeme effects) would reflect non-lexical regularities: character class transitions, phonotactic constraints, orthographic conventions.

Definition 15 (Lexeme-adjusted count table)**.** *Given the base count table $c(x_d, o)$ and a set of lexemes $\mathcal{W} = \{w_1, \ldots, w_M\}$, the* lexeme-adjusted *count table is:*

$$c'(x_d, o) = c(x_d, o) - \sum_{w \in \mathcal{W}} c_w(x_d, o) \cdot P(w), \tag{10}$$

where $c_w(x_d, o)$ is the count table restricted to positions inside word w. This subtracts out the contribution of each word's occurrences from the overall byte statistics.

Remark 16 (What remains after subtraction?)**.** *After subtracting the "the" contribution from the byte-level Markov statistics, what changes?*

- *The bigram "th" becomes less frequent ("the" contributed a large fraction of "th" occurrences).*
- *The bigram "he" followed by space becomes rarer.*
- *The trigram "the" + space becomes much rarer.*
- *But "th" in "this", "that", "think" etc. remains.*

The residual "th" bigram now reflects all sources except *the word "the." Repeating this for each word in the lexicon would peel away successive layers of word-level causation, leaving only the "pure" character-level regularities.*

7 The Lexeme Map as a Complete Model U

Theorem 17 (The char-to-lexeme map is a UM)**.** *The map from input bytes to support over lexemes has the shape of a complete Universal Model $U = (E, T, P, f, \omega)$:*

- $E_{in} = \{0, \ldots, 255\}^k$*: the input bytes at offsets $d_1, \ldots, d_k$.*
- $E_{out} = \{w_1^+, w_1^-, w_2^+, w_2^-, \ldots, w_M^+, w_M^-\}$*: the lexeme binary ESes.*
- T*: the log support values over the lexeme events.*
- P*: the atomic patterns connecting bytes to lexemes (one pattern per letter per offset per lexeme).*
- f*: the UM forward pass—existential quantification over the input patterns to compute output support.*
- $\omega = \omega_0$*: the standard counting function.*

This is a complete UM in its own right. Its "output" is not a byte prediction but a lexeme prediction: given the recent bytes, which word are we in?

Remark 18 (Two UMs, composed)**.** *We now have two UMs:*

1. U_{byte}*: bytes* $\to$ *bytes. The existing model (isomorphic UM, KN model, pattern-chain model, etc.).*
2. U_{lex}*: bytes* $\to$ *lexemes. The new model from Theorem 17.*

The UM algebra [1, 5] tells us how to compose them. The composed model is:

$$U_{full} = U_{byte} \bowtie U_{lex}, \tag{11}$$

where $\bowtie$ *denotes the UM join (product of event spaces, union of patterns, combined forward pass).*

The full model predicts the next byte using both *byte-level patterns (character transitions, offset conjunctions) and lexeme-level patterns (word identity, word-boundary prediction). Neither model is subordinate: they operate in parallel, and the forward pass combines their support values.*

Proposition 19 (The join preserves the tower)**.** *The composed model* U_{full} *maintains the factorization tower [2]:*

$$E_{bytes} \xleftarrow{recognition} E_{lexemes} \xrightarrow{prediction} E_{bytes}. \tag{12}$$

This is a product factorization (both levels coexist), not a sum factorization (one replaces the other). The byte-level event space is never discarded. The model can always "backtrack" to bytes because the byte level was never left.

8 Spelling Distributions

Definition 20 (Spelling distribution)**.** *A word* w *has a* spelling distribution*: the probability distribution over byte sequences that realize* w*. For most words, this distribution is concentrated on one spelling:* $P(\texttt{the} \mid w = \text{"the"}) \approx 1$*.*

But the distribution is not degenerate:

- *Capitalization: "The" vs. "the" vs. "THE".*
- *Typos: "teh" for "the" (a common transposition).*
- *Encoding variants: "café" vs. "cafe" vs. "caf\u00e9".*
- *Historical/dialectal: "colour" vs. "color".*

The lexeme ES captures the abstract *word identity; the spelling distribution captures its byte-level realizations.*

Remark 21 (The bag of letters handles misspellings naturally)**.** *Consider the misspelling "teh" for "the." In the bag-of-letters model:*

	"the"	*"teh"*
Letter set	$\{t,\ h,\ e\}$	$\{t,\ e,\ h\}$
Offsets (from space)	$d = 3, 2, 1$	$d = 3, 2, 1$

The bag of letters is identical. *The only difference is which letter maps to which offset. If the atomic patterns primarily care about which letters are present (not their exact ordering), then "teh" provides almost the same support for "the"$^+$ as "the" does.*

The offset-specific information does differ: $T(\texttt{e}_{d=2})$ vs. $T(\texttt{h}_{d=2})$ are different values. But the conjunction *support $T(\texttt{t} \wedge \texttt{h} \wedge \texttt{e}$ within 3 offsets) is identical. This is exactly the "bag of letters" effect from psycholinguistics: the letters are present, the word is recognized, the internal order matters less than the external boundaries.*

9 Negative Events and the Prefix Class

Definition 22 (Prefix class)**.** *For a word w, its* prefix class $[w]_{pre}$ *is the set of words sharing the same prefix:*

$$[w]_{pre} = \{w' \in \mathcal{W} : w \text{ is a prefix of } w' \text{ or } w' \text{ is a prefix of } w\}. \tag{13}$$

For $w =$ "the": $[\text{"the"}]_{pre} = \{$the, there, their, them, then, these, they, thereby, thereof, $\dots\}$.

Proposition 23 (Negative event distributes over prefix class)**.** *When the evidence is consistent with "the" but the actual word is not "the," the negative event distributes over the prefix class:*

$$P(\text{"the"}^- \mid \text{prefix} = \text{"the"}) = \sum_{w' \in [\text{"the"}]_{pre} \setminus \{\text{"the"}\}} P(w' \mid \text{prefix} = \text{"the"}). \tag{14}$$

This distribution is computable from lexeme frequencies:

$$P(w' \mid \text{prefix} = \text{"the"}) = \frac{\text{freq}(w')}{\sum_{w'' \in [\text{"the"}]_{pre}} \text{freq}(w'')}. \tag{15}$$

Example 24 ("the" prefix class distribution)**.** *Approximate frequencies in enwik9 (relative to prefix "the"):*

Word	*Rel. freq. (%)*	*Continuation byte*
the (+ space)	*∼65*	' '
there	*∼8*	r
their	*∼7*	i
they	*∼6*	y
them	*∼5*	m
then	*∼4*	n
these	*∼3*	s
the + other	*∼2*	*various*

The binary ES at the point where we have seen "the" and must predict the next byte:

- *w^+: the word is "the" → predict space (∼65%).*
- *w^-: the word is not "the" → predict the continuation byte from the table above (∼35%, distributed over* r, i, y, m, n, s, $\dots$*).*

This is a clean binary ES with well-defined positive and negative events, where the negative event has a natural decomposition.

Remark 25 (Case study for negative event distribution)**.** *This "the" example is a particularly clean case study for the general problem of negative event distribution in the UM. The issues that arise:*

1. *The negative event is not uniform—it has structure (the prefix class with specific frequencies).*
2. *The negative event* itself *decomposes into further binary ESes: "there" vs. "not there," etc.*
3. *As we expand the lexicon, each new word "steals" probability mass from the negative events of related words.*
4. *The prefix tree (trie) over the lexicon provides the natural structure for this decomposition.*

As we eventually expand the lexicon in the model, we will need to handle these interactions. Starting with "the" gives us a controlled setting where the prefix class is small enough to enumerate but rich enough to illustrate all the key phenomena.

10 Discussion

10.1 What changed from the previous papers

The tock-step paper [2] and tock protocol [3] proposed adding words via dedicated finite automata (word detectors). This paper makes a different and more fundamental point: we don't need new detectors because the model's *existing* atomic patterns already carry the word-level information.

The atomic patterns—skip-bigrams at various offsets—are the "letters" of the bag-of-letters model. Each letter at each offset provides independent evidence. The word is not detected; it is *inferred* from the conjunction of letter evidence.

This is both simpler (no new machinery) and more powerful (handles misspellings, capitalization variants, and partial evidence naturally).

10.2 The UM composition

The central result (Theorem 17) is that the char-to-lexeme map is itself a complete UM. This means we don't need to "bolt on" word-level prediction as a separate system; it fits within the existing UM algebra.

The composed model $U_{\text{byte}} \bowtie U_{\text{lex}}$ is a single UM with a richer event space. The byte-level patterns and lexeme-level patterns coexist and contribute jointly to prediction. The UM forward pass (existential quantification over probabilistic syllogisms) handles the combination automatically.

10.3 Next steps

This is the first of several intermediate papers (per MJC's direction). The next steps:

1. **Empirical validation**: compute $T(w^+)$ for "the" from actual data, verify that bag-of-letters support matches word occurrence, measure the bpc gain from the binary ES.
2. **Negative event structure**: formalize the prefix-class decomposition and show it gives the correct probability distribution over continuation bytes.
3. **UM composition**: implement the join $U_{\text{byte}} \bowtie U_{\text{lex}}$ and measure the combined model's bpc.
4. **Lexicon scaling**: extend from one word to many, using the prefix trie to efficiently handle negative event distribution.

References

[1] Michaeljohn Clement. *CMP*. `https://cmpr.ai/cmp.pdf`, 2026.

[2] Claude and MJC. *The Tock Step: Domain-Native Architecture from Evidence*. Hutter archive, 12 Feb 2026.

[3] Claude and MJC. *The Tock Protocol: Systematic Lexicon Injection into the Isomorphic UM*. Hutter archive, 13 Feb 2026.

[4] Claude and MJC. *The Real Factor Map: Interpretable Patterns onto Hidden Dynamics*. Hutter archive, 9 Feb 2026.

[5] Claude and MJC. *Algebraic Semantics of the Universal Model*. Hutter archive, 12 Feb 2026.