Multi-Frequency Recurrence in the Universal Model

v3: Positive and Negative Results

Claude and MJC

18 February 2026

1 Introduction

The KN-6 model in the UM runner uses six orders of n-gram context, each looking at a contiguous window of 1–6 bytes. This is a **single-frequency** architecture: every LPP operates at the byte clock. But natural language contains structure at multiple timescales: characters, morphemes, words, phrases, sentences, paragraphs, topics.

This paper develops **multi-frequency recurrence**: extending the UM forward pass so that different groups of event spaces and LPPs can operate at different clock rates, capturing structure across timescales. We validate the approach with six experiments on 1M bytes of enwik9: two positive results (**+0.184 bpc** from word + tag LPPs) and four negative results that reveal the structure of information redundancy between frequency groups.

2 Frequency in the UM

2.1 The byte clock

Currently, the UM processes one byte per tick. At each tick:

1. Input event: the observed byte enters byte_input ES.
2. Forward pass: KN-6 propagates through 6 LPPs.
3. Learning: ω_0 updates counts.
4. Output: prediction distribution over byte_output ES.

2.2 The quotient perspective

From the um-arithmetic framework (archive 20260216), the quotient $Q = e/|E|$ maps events to the unit interval. Different quotient bases give different "clocks":

- $\lambda = 1$: byte clock (256 events per tick)
- $\lambda = 8$: bit clock (2 events per tick)
- $\lambda = 1/\ell$: word clock ($\sim$5000 events per tick, where ℓ is mean word length)

3 The Shift-Register Connection

The weight construction results (archive 20260211) showed that the sat-rnn's hidden state acts as a shift register with 16 groups of 8 neurons. Different groups respond to inputs at different time offsets.

The skip-pattern analysis (archive 20260208) found that non-contiguous offsets (e.g., [1,8,20,3]) capture nearly as much information as contiguous order-12 contexts (0.069 vs 0.067 bpc). The offset 8 was chosen before offset 2 because it provides complementary information.

These findings directly support multi-frequency architecture: the shift-register groups *are* multi-frequency detectors, skip patterns capture information at specific frequencies, and the optimal offset selection is a form of frequency tuning.

4 Positive Results

4.1 Word-onset LPP

The first multi-frequency P-program adds a word-onset LPP: at each byte position, the context is (first byte of current word, position in word capped at 15), predicting the output byte. A separate 1M hash table stores joint counts; a learned sigmoid mixing weight α combines with KN-6.

Result: KN-6 2.398 $\rightarrow$ 2.279 bpc (**+0.118**). Learned weight $w = 0.091$ (9.1% word-onset). HT: 30.6K entries (2.9%).

4.2 Three-frequency model: word + tag

The tag-onset LPP adds a second frequency: context is (in_tag flag, previous byte). A 3-way softmax combines KN-6, word-onset, and tag-onset.

Results on 1M enwik9:

Model	bpc	Gain
KN-6 alone	2.398	—
KN-6 + word	2.279	+0.118
KN-6 + word + tag	2.214	+0.184

Learned weights: KN-6 = 85.3%, tag = 11.0%, word = 3.6%. Tag HT: 6.2K entries (0.6%). Total structural memory: 36.8K entries (3.5%).

4.3 Analysis of positive results

1. **Tag dominates word.** The tag LPP gets 11% vs word's 3.6%, matching the surprise analysis where in_tag explains 86.8% of mean surprise.

2. **Marginal gains are additive.** Tag adds +0.066 on top of word's +0.118, with no destructive interference.

3. **Tiny memory.** 36.8K structural entries vs millions of byte-level KN entries. Structure is sparse.

4. **Exceeds v1 predictions.** Predicted $\sim$0.03 bpc; achieved +0.184 (6$\times$ more), because the LPP captures byte-conditional patterns within each regime.

5 Negative Results

Four experiments tested whether further frequency groups add value beyond word + tag. All are negative.

5.1 OS-conditional LPP (+0.005, marginal)

Context: (oversupport bucket, in_tag, prev_byte). The oversupport bucket discretizes the ring pattern metric into low/medium/high.

Result: only +0.005 bpc over 3-way model. The OS-conditional LPP (10.4% weight) *subsumes* the tag LPP (drops to 1.4%), since its context is a superset of the tag context.

Lesson: surprise detection does not directly improve prediction. Oversupport is useful for *attribution* (identifying where the model is confused), not for *prediction* (making better forecasts). This confirms MJC's insight: "surprise is attribution, not interpretation."

5.2 Line-position LPP (+0.004, marginal)

Context: (position in line capped at 31, prev_byte). Five-frequency model with word + tag + line-pos + after-period.

Result: +0.004 bpc over 3-way. Line-position gets 9.4% weight but *steals* from tag (drops from 11% to 1.5%). After-period: 0.7%.

Lesson: line position and tag status are highly correlated in enwik9 (lines starting with `<` are tags). Adding correlated features does not compound—they redistribute weight but don't add new information.

5.3 Skip-bigram LPPs (+0.002, marginal)

Context: (data[$t - d$], prev_byte) for $d \in \{8, 16, 32\}$.

Solo performance (each skip with KN-6 only):

Offset	bpc	Gain
$d = 8$	2.269	+0.129
$d = 16$	2.272	+0.126
$d = 32$	2.275	+0.123

Combined with word + tag: only +0.002 bpc marginal.

Key finding: solo skip-bigrams give $\sim$+0.13 bpc (comparable to word-onset at +0.118), but are *completely redundant* with structural LPPs. The word-onset and tag-onset LPPs capture the same non-contiguous information that skip-bigrams provide—but more efficiently (less memory, more interpretable). **Structural features are the efficient representation of what skip-patterns capture statistically.**

6 The Complementarity Principle

The experiments reveal a **complementarity principle**: adding LPPs improves prediction only when they capture genuinely independent information. Redundant features (correlated with existing LPPs) just redistribute weight without improving total compression.

The effective features are those that partition the data into regimes with different distributions:

- **in_tag**: tag vs text regime (86.8% of variance)
- **word position**: boundary vs interior (1.93× harder)

Features that fail are those correlated with these two:

- Line position ≈ tag status (both reflect XML structure)
- Skip-bigrams ≈ word/tag structure (same information, different encoding)
- Oversupport ≈ tag status (confused where tag status changes)

To gain beyond +0.184 bpc, we need features *orthogonal* to both tag and word structure: perhaps content-specific features (article topic, entity type), syntactic features (part of speech), or cross-article features (template detection).

7 Predicted Gains at Scale

At 1M bytes, two structural LPPs capture +0.184 bpc (7.7% of baseline).

At scale (1B bytes, KN-6 ≈ 1.78 bpc), the absolute gains will shrink as KN-6 itself captures more structure: $\text{gain}_{1B} \approx 0.184 \times 1.784/2.398 \approx 0.137$ bpc (∼16 MB on the Hutter Prize).

But the real value is demonstrating the architecture: once the UM runner supports arbitrary P-programs as LPPs, gains from truly orthogonal features can compound.

8 Conclusion

Multi-frequency recurrence extends the UM from a single-clock system to a bank of matched filters. On 1M enwik9:

- **Positive**: word-onset + tag-onset = +0.184 bpc (85/11/4%)
- **Negative**: OS-conditional, line-position, skip-bigrams, after-period all add <0.005 bpc marginal
- **Principle**: gains require complementary information, not just more features. Correlated features redistribute weight without improving compression.
- **Key insight**: structural LPPs (word, tag) are the efficient representation of what skip-bigrams capture statistically.