# The Representation Bottleneck

Claude and MJC

March 2026

**Abstract**

Three experiments in a single session converge on one conclusion: the UM's combination problem is a *representation* problem, not a scoring-rule problem. (1) Retroactive training gives 0.5 bpc at 100K, proving that data coverage matters. (2) Adding 4-gram/6-gram LPPs degrades performance under *all three* scoring rules tested. (3) External KN-6 with H3 scoring works because KN distributions are probability-calibrated. The UM's log-stochastic weights encode observation magnitude but not distributional confidence, making it impossible to distinguish reliable predictions from spurious ones when combining multiple LPPs.

## 1 Three Experiments, One Conclusion

### 1.1 Retroactive Training

The 3-pass pipeline (online → retroactive → frozen) achieves:

| $N$ | Online | Frozen |
|---|---|---|
| 100K | 4.485 | 3.846 |
| 1M | 4.078 | 3.744 |

The 0.64 bpc gain at 100K (14%) is the largest single-technique improvement. But it operates *within* the trigram model. It makes existing neurons better, not the combination rule smarter.

### 1.2 Order Degradation

Adding higher-order LPPs via `wm-ngram` degrades frozen bpc:

| Order | Sharpest | H3 | Count |
|---|---|---|---|
| 3 | 3.846 | **3.826** | 4.087 |
| 4 | 4.056 | 4.056 | 4.055 |
| 6 | 4.074 | 4.074 | 4.070 |

The degradation is robust to all three scoring rules.

### 1.3 KN-6 H3

The external KN-6 system uses the same H3 rule ($2^{g/H}$) and *improves* from order 4 to 8 (2.484 → 2.189 bpc at 1M). KN distributions are probability-calibrated via the recursive discount formula $p_k = (c - D)/n + \lambda \cdot p_{k-1}$.

## 2 The Bottleneck

A UM LPP entry has weight $w \in \{0, \ldots, 255\}$. The distribution for a source event is $p(b) = 2^{w_b} / \sum_b 2^{w_b}$. Two contexts can produce the same distribution:

- Context A: 500 observations, $w_{\text{top}} = 9$, $w_{\text{second}} = 7$. Gap = 2. Reliable: the top byte is consistently most common.
- Context B: 3 observations, $w_{\text{top}} = 2$, $w_{\text{second}} = 0$. Gap = 2. Unreliable: only 3 data points.

The gap is the same. The entropy is different (lower for B), but not because B is more reliable—it's because B has seen fewer distinct outcomes. Under any gap-based or H3-based rule, B looks at least as good as A.

In KN, the same two contexts have $n_A = 500$ and $n_B = 3$. The KN weight $D \cdot n/(n + D)$ is 0.998 for A and 0.75 for B. The 500-observation context dominates. This is why KN works and the UM doesn't: KN has access to the raw count, not just the log-stochastic shadow.

## 3 Path Forward

Two approaches:

1. **Add count metadata**: store total observation count per (source, LPP) pair alongside the log-stochastic weights. Use this count in the combination rule. This is mechanically simple but philosophically uncomfortable—it adds non-UM state to the scoring.
2. **Probability calibration**: convert log-stochastic weights to calibrated probabilities before combination. This requires knowing the effective sample size, which is what the count metadata would provide.
3. **KN-native pipeline**: use KN interpolation inside the UM framework. Each "LPP" would compute KN-discounted probabilities rather than log-stochastic distributions. This is the pragmatic path: proven to work, and the order-scaling paper shows H3 on top of KN beats KN alone at high orders.

The KN-native path is the most direct route to competitive compression. The pure-UM path requires solving the count problem first.

## 4 Conclusion

The combination problem is not "which LPP should win?" but "how well-calibrated are each LPP's predictions?" Log-stochastic weights compress the count information needed for calibration. Any scoring rule operating on uncalibrated distributions will face the same degradation when combining sparse and dense sources. The path to competitive compression goes through probability calibration, whether via KN discount, explicit count metadata, or a future UM-native mechanism.