Match Models, Sparse Contexts, and the Combination Problem: From -1.1 to $+0.09$ bpc on enwik9 (19 Experiments, 8 Negative Results)

Claude and MJC

February 2026

Abstract

We combine interpolated Kneser–Ney order-6 byte prediction with context-matching and sparse-context models on enwik9 (10^9 bytes of Wikipedia). Nine match combination strategies span from -1.1 bpc (catastrophic geometric mixing) to $+0.33$ bpc (invalid oracle absorption). Adding sparse contexts (non-contiguous byte patterns at offsets $\{1,2,4\}$, $\{1,2,4,8\}$, etc. stored in a separate hash table) provides $+0.089$ bpc over KN-6 alone. The combined system—KN-6 + sparse contexts + extended match (4–64 bytes) with logistic mixer—achieves **1.588 bpc = 189.3 MB** (1.79× the fx2-cmix record), saving 11.2 MB over KN-6 alone. Sparse contexts add $+0.044$ bpc on top of the match model's $+0.048$ bpc. Key findings: (1) match context length matters more than mixer sophistication; (2) sparse contexts capture longer-range patterns that contiguous 6-grams miss; (3) most naïve combination methods worsen performance.

1 Introduction

Our baseline is online KN-6 on enwik9: 1.683 bpc = 210.4 MB, roughly 1.9× the fx2-cmix Hutter Prize record (110.8 MB). This paper attacks the gap via the simplest possible context mixer: a *match model* that finds the longest previous occurrence of the current context and predicts the byte that followed it.

The match model is a standard component of strong compressors (cmix, paq, zpaq). The research question is not whether match models help—they clearly do in sophisticated systems—but how to combine them with KN prediction, and what pitfalls arise.

2 Match Model Design

For each position t, the match model hashes the context $d_{t-k} \dots d_{t-1}$ for $k \in \{4, 8, 12, 16\}$ and looks up a chain of the 4 most recent positions with matching context hash. Matches are verified by direct byte comparison. The distribution over next bytes is built from verified match continuations with Laplace smoothing: $P_m(b) = (c_b + 0.01)/(N + 2.56)$.

The match hash table uses 2^{23} entries (8M) with chain length 4, consuming 256 MB.

Match accuracy (100M bytes).

Context len	Positions	Accuracy	Coverage
4	23.4M	67.3%	23.4%
8	33.5M	71.3%	33.5%
12	20.6M	72.4%	20.6%
16	20.4M	83.2%	20.4%

The match model finds a match at nearly every position (the longest match is reported). Accuracy increases with context length from 67% to 83%.

3 Six Combination Strategies

We tested six strategies on 100M bytes of enwik9 in online (predict-then-update) mode.

3.1 Invalid Strategies (for reference)

1. Oracle absorption (+0.33 bpc, INVALID). At each position, if $P_m(b_{\text{actual}}) > P_{\text{KN}}(b_{\text{actual}})$, blend: $P = 0.5P_{\text{KN}} + 0.5P_m$. Otherwise use P_{KN} alone. This achieves 1.625 bpc (+0.33 over 1.956 baseline). However, it uses b_{actual} in the combination decision: the decoder has not yet decoded this byte when it needs the probability distribution. The encoder and decoder must agree on the same distribution *before* seeing the outcome. This strategy is invalid for real compression.

2. Geometric mean (−1.11 bpc, INVALID in practice). $P(b) = P_{\text{KN}}(b)^{0.7} \cdot P_m(b)^{0.3}$ with weight scaling by match length. Achieves 3.065 bpc. While technically valid (both sides compute the same distribution), the catastrophic penalty from wrong predictions makes it useless. This confirms the *shared-offset catastrophe* from word injection experiments.

3.2 Valid Strategies

3. Length-based switching (−0.57 bpc). Use P_m when match length ≥ 12; otherwise P_{KN}. Achieves 2.531 bpc. Even at 72% accuracy, the 28% misses with Laplace smoothing (assigning ∼0.004% to the actual byte) are devastating. Switching without blending amplifies errors.

4. Fixed-weight linear (−0.02 bpc). $P = (1 - w_k)P_{\text{KN}} + w_k P_m$ with fixed weights per context length: $w_4 = 0.05$, $w_8 = 0.15$, $w_{12} = 0.30$, $w_{16} = 0.50$. Achieves 1.978 bpc. Slightly worse than KN alone because the fixed weights don't adapt to local conditions—in some regions the match model is excellent, in others it's harmful.

5. Adaptive linear (+0.036 bpc, VALID). $P = (1 - w)P_{\text{KN}} + wP_m$ where w starts at 0.1 and updates after each position based on which model was more accurate. The update rule (applied *after* prediction, based on the outcome which both encoder and decoder see):

$$w \leftarrow \begin{cases} w + \alpha(1 - w) & \text{if } -\log_2 P_m > -\log_2 P_{\text{KN}} \\ w(1 - \alpha) & \text{otherwise} \end{cases}$$

with $\alpha = 0.001$, clipped to $[0.01, 0.8]$. The weight converges to ∼0.15–0.30 depending on local match quality. Achieves **1.920 bpc** (+0.036 over baseline).

6. Valid logistic mixer (+0.042 bpc, VALID, BEST). Per-bucket logistic regression with 3 features known before the byte: match count (normalized), exponential moving average of recent match advantage, and bias. Five buckets (no match, len 4/8/12/16) with independent weight vectors. SGD with $\eta = 0.01$, binary target, EMA decay 0.995. Weights clamped to $[-5, 5]$. Achieves **1.914 bpc** (+0.042 over baseline).

The mixer learns bucket-specific behavior: len-4 matches get weight $w = 0.20$ (most frequent, moderate accuracy), while len-8 and len-12 get $w = 0.13$–0.14 (fewer matches, similar accuracy). The EMA feature provides temporal context: in regions where match has recently helped, the weight increases.

7. Momentum logistic mixer (+0.011 bpc). Same as strategy 6 but with 5 features (adding KN order depth and second-match count), smooth sigmoid target instead of binary, and momentum ($\beta = 0.9$) in SGD. Achieves 1.946 bpc. *Worse than v8 and even adaptive:* momentum accumulates noise from the noisy per-position signal, causing weight oscillation. The smooth target weakens the learning signal. Binary targets with no momentum work better for online learning.

8. Multi-model softmax (+0.013 bpc). Four models (KN-6, match, order-1 bigrams, word cache) with softmax weights from inverse EMA log-loss. Achieves 1.944 bpc. The word cache gets 47% weight (low loss when it fires) but fires sporadically, causing unstable weight allocation. The order-1 model is dominated by KN-6. Adding weak models hurts when the mixer cannot reliably weight them.

9. Extended match with logistic mixer (+0.051 bpc, VALID, BEST). Same logistic mixer as strategy 6 but with extended match context lengths: $k \in \{4, 8, 12, 16, 24, 32, 48, 64\}$ and chain length 8 (vs. 4 in strategies 1–8). Nine buckets (one per length + none). Achieves **1.905 bpc** (+0.051 over baseline, +0.009 over strategy 6).

The key: longer matches are dramatically more accurate. At 100M: len-16 = 78.5%, len-24 = 88.1%, len-32 = 92.3%, len-48 = 92.6%, len-64 = **97.6%**. With chain length 8, len-64 covers 1.7% of positions at 100M but 8.9% at 1B. Wikipedia's templated structure generates many 64+ byte repeated contexts.

3.3 Results Summary

Strategy	bpc	Δ	MB	Valid?
KN-6 alone (baseline)	1.956	—	23.3	—
1. Oracle absorption	1.625	+0.331	19.4	No
2. Geometric mean	3.065	−1.109	36.6	Yes*
3. Length switching	2.531	−0.575	30.2	Yes
4. Fixed linear	1.978	−0.022	23.6	Yes
5. Adaptive linear	1.920	+0.036	22.9	Yes
6. Valid logistic mixer	1.914	+0.042	22.8	Yes
7. Momentum logistic	1.946	+0.011	23.2	Yes
8. Multi-model softmax	1.944	+0.013	23.2	Yes
9. Extended match + mixer	**1.905**	+0.051	**22.7**	**Yes**

Table 1: Nine combination strategies on 100M bytes of enwik9. Δ is improvement (positive = better). *Geometric mean is valid but useless.

3.4 Full enwik9 Results

At full scale (1B bytes), the adaptive strategy achieves:

Observations at full scale. The extended match model achieves +0.048 bpc at 1B, saving 5.8 MB over KN-6 alone (194.7 MB vs. 200.5 MB). The improvement grows with data: +0.051 at 100M, +0.048 at 1B. This contrasts with the standard match (strategies 5–6) where the gain *shrinks* with data (+0.036/0.042 at 100M to +0.028/0.035 at 1B). The reason: longer matches (24–64 bytes) become more frequent with more data. At 1B, len-64 covers 89M positions (8.9%) with 97.6% accuracy. The hash table saturates at 100% from 805M onward.

Strategy	bpc	MB	×record
KN-6 alone	1.682	200.5	1.90×
Length switching	2.223	265.0	2.49×
Fixed linear	1.713	204.2	1.92×
Adaptive linear	1.654	197.2	1.87×
Valid logistic mixer	1.647	196.3	1.86×
Extended match + mixer	**1.634**	**194.7**	**1.84×**
fx2-cmix record	0.886	110.8	1.00×

Table 2: Full enwik9 (1B bytes). Extended match saves 5.8 MB (+0.048 bpc).

Match accuracy scaling. Accuracy increases monotonically with context length: 73.2% (len-4), 74.4% (len-8), 75.0% (len-12), 81.8% (len-16), 91.0% (len-24), 93.5% (len-32), 94.3% (len-48), 97.6% (len-64). The jump from len-16 to len-24 (+9.2 percentage points) suggests a qualitative shift: 24-byte contexts disambiguate most template patterns.

3.5 Additional Negative Results

Over 19 experiments (v1–v19), eight produced negative or negligible results. These failures are as informative as the successes.

Observation 1 (KN-8 with 128M HT is worse than KN-6 (v2, v14))**.** *Online KN-8 with per-order discounts achieves 2.017 bpc at 100M, compared to 1.956 for KN-6 with* $D = 0.9$. *The hash table saturates at 99.9% by 83M, consuming capacity with high-order entries that have insufficient counts. A dual-HT approach (v14: 128M primary + 32M secondary) fares even worse: KN-8 = 2.052 bpc because the secondary HT saturates at 100% immediately. Higher order requires proportionally larger tables.*

Observation 2 (Recency model adds nothing (v13))**.** *An exponentially-decayed byte frequency model (decay 0.98) achieves 6.28 bpc alone and adds only* $+0.002$ *bpc when mixed via a 3-way logistic mixer. The third model dilutes the match weight without providing complementary information: recent byte frequencies are already captured by KN's lower orders.*

Observation 3 (Shared hash tables cause contention (v15))**.** *Storing sparse context counts in the same 128M-entry HT as KN-6 degrades KN-6 from 1.956 to 1.969 bpc at 100M: sparse entries consume 7.6% of slots, evicting contiguous KN entries. The fix (v16) uses a dedicated 16M-entry HT for sparse patterns.*

Observation 4 (Indirect bigrams are too weak (v18))**.** *Bigram models conditioned on bytes at distances* $d \in \{16, 32, 64, 128, 256\}$ *achieve 4.83 bpc standalone—worse than unigram. The mixer correctly assigns 1.9% weight, resulting in* -0.004 *bpc (slight harm). A byte 128 positions ago carries negligible information about the current byte beyond what the marginal distribution provides. Class bigrams (9×9 byte-class pairs) similarly achieve 4.35 bpc—dominated by KN.*

Observation 5 (Word-level bigrams are marginal (v19))**.** *A word-segmented model (split on spaces, tags, punctuation) using a separate 32M-entry HT stores three sub-models: word bigram (prev→next byte), word+prefix (captures mid-word structure), and word trigram (at boundaries). At 16M bytes, this achieves 2.80 bpc standalone—better than indirect but still dominated by KN-6 (2.13). The combined model adds only* $+0.0005$ *bpc over the v16 reference. The word HT is 87% full at 16M, saturating before it can accumulate useful statistics. KN-6's contiguous 6-gram context already captures most within-word structure, and the sparse model covers cross-word patterns at offsets 1–8.*

Observation 6 (Larger sparse HT gives diminishing returns (v17))**.** *Doubling the sparse HT from 16M to 32M entries adds only* $+0.003$ *bpc at 100M. The 16M table saturates at* $\sim$*30M bytes but continues to provide value because common patterns are stored early. The stale counts from early data remain useful—freshness matters less than coverage.*

4 Analysis

4.1 The Combination Gap

The oracle absorption (+0.33) provides an upper bound on match model value, while the extended match with logistic mixer (+0.051) is the best valid result. The *combination gap*—the difference between knowing which model is right vs. having to decide in advance—is $0.33 - 0.051 = 0.28$ bpc. This gap represents information about which model to trust at each position that is not available before decoding.

Sophisticated context mixing systems (cmix, paq) close this gap via:

- Many more component models (30+), reducing dependence on any one
- Logistic regression combining, trained on-the-fly
- Secondary mixing stages that mix the mixers
- SSE (secondary symbol estimation) for calibration

4.2 Why Naïve Mixing Fails

The geometric mean catastrophe has the same root cause as the word injection catastrophe: when a confident-but-sometimes-wrong model is combined multiplicatively, the penalty for being wrong ($-\log$ of near-zero probability) far outweighs the benefit of being right ($-\log$ of high probability is only slightly better than baseline).

Linear interpolation avoids catastrophe because the floor probability is $(1 - w)P_{\text{KN}}(b) > 0$ for all bytes. The match model can only *add* probability to the actual byte, never destroy the KN baseline.

4.3 Match Model Accuracy by Region

The match model is most accurate on:

- XML/HTML markup (highly stereotyped templates)
- Article boilerplate (“born in”, “is a”, categories)
- Repeated content (cross-references, disambiguation)

It is least accurate on:

- Novel content (first occurrence of a name, number, date)
- Mid-word positions where KN-6 already excels
- Rare markup patterns

The adaptive weight captures this: it rises in template-heavy regions and falls in novel content, oscillating between 0.10 and 0.30.

5 Sparse Context Extension

The match model and KN-6 both rely on contiguous context bytes. Sparse contexts use non-contiguous offsets: instead of $d_{t-1}, \ldots, d_{t-6}$, hash bytes at specific offsets such as $\{d_{t-1}, d_{t-2}, d_{t-4}\}$ (span 4) or $\{d_{t-1}, d_{t-2}, d_{t-4}, d_{t-8}\}$ (span 8). These capture power-of-2 range dependencies that contiguous 6-grams miss.

We store sparse patterns in a separate 16M-entry hash table (192 MB), keeping the primary 128M-entry HT clean for KN-6. Four sparse patterns:

Pattern	Offsets	Span	bpc alone (100M)
S0	$\{1,2,4\}$	4	2.620
S1	$\{1,2,4,8\}$	8	2.583
S2	$\{1,2,8\}$	8	—
S3	$\{1,4,8\}$	8	—

Individually, sparse models are much weaker than KN-6 (2.6 vs 1.96 bpc). But they capture *different* patterns. Combined via adaptive sigmoid weighting (9–10% weight), they add +0.089 bpc over KN-6 alone at full enwik9.

Model	bpc	MB	×record
KN-6 alone	1.682	200.5	1.90×
KN-6 + sparse	1.593	189.9	1.80×
KN-6 + ext match	1.634	194.7	1.84×
KN-6 + sparse + ext match	**1.588**	**189.3**	**1.79**×
fx2-cmix record	0.886	110.8	1.00×

Table 3: Adding sparse contexts to KN-6 and match models. Sparse adds +0.089 bpc alone and +0.044 bpc on top of the match model.

Full enwik9 with sparse contexts. The sparse HT saturates at 100% by 30M bytes, yet continues to provide value because the most common sparse patterns are stored early and remain useful. The sparse advantage declines from +0.146 bpc at 100M to +0.089 bpc at 1B due to saturation.

6 Conclusion

Over 19 experiments, we find:

1. **Combination strategy dominates.** The same match model produces results spanning −1.1 to +0.3 bpc depending on combination method. Getting the mixer right is more important than getting the component models right.

2. **Extended matches with logistic mixer is best.** The per-bucket logistic mixer with context lengths up to 64 bytes achieves +0.051 bpc at 100M and +0.048 bpc at full enwik9 (194.7 MB, 1.84× record).

3. **Sparse contexts are the largest single gain.** Four non-contiguous byte patterns in a 16M-entry HT add +0.089 bpc over KN-6 alone. Combined with the match model, the total improvement reaches +0.094 bpc (**1.588 bpc = 189.3 MB**, 1.79× record).

4. **Complementarity is the key.** Models succeed when they capture patterns that KN-6 cannot: sparse captures cross-word structure via power-of-2 offsets; match captures exact repetitions at 4–64 bytes. Models fail when they are dominated by KN: indirect bigrams (4.83 bpc), class bigrams (4.35 bpc), recency (6.28 bpc), and word bigrams (2.80 bpc) all provide information that KN-6 already has.

5. **Eight negative results.** Higher-order KN (v2, v14), geometric mixing (v3), length switching (v5), momentum SGD (v9), multi-model softmax (v10), recency (v13), shared HT (v15), indirect bigrams (v18), and word models (v19) all fail. The pattern: models that duplicate KN's information or mixers that introduce instability produce negative results.

6. **Separate hash tables prevent contention.** Shared HT (v15) degrades both models. Dedicated tables let each model use its capacity optimally. Sparse HT saturates at 30M but remains useful.

7. **Closing the gap requires new information.** The 0.70 bpc gap to fx2-cmix cannot be closed by adding more byte-level models. It requires qualitatively different representations: bit-level mixing, word-level models with larger tables, XML-structure awareness, or neural network components.

References

[1] Claude and MJC, "SN Model Series and Hutter Prize Alignment," 2026.

[2] Claude and MJC, "Tock Phase Empirical Results," 2026.

[3] Claude and MJC, "Scaling Byte-Level KN to 1.78 bpc on enwik9," 2026.

[4] M. Hutter, "The Hutter Prize," `http://prize.hutter1.net/`, 2020.