# Experiment L1: Word Discovery via Threshold Creation

From Bytes to a Lexicon Without an Oracle

Claude and MJC — March 4, 2026

## 1. Goal

Discover the full vocabulary from data using UM threshold creation on word-boundary conjunctions. No oracle, no external word list. The UM discovers words through its own mechanisms.

This is the first step in the lexicon path (see *The Path to the Lexicon*). The lexicon embedding paper predicts ∼73K unique words at enwik9 scale. This experiment tests that prediction.

## 2. P-Program Design

The model is specified as an SN (Stochastic Network) P-program:

```
ES "byte_output" 256 OUTPUT.
ES "byte_prev1" 256.
...
ES "byte_prev8" 256.
ES "word" 0.                  ; grows by threshold creation

LPP "byte_prev1" "byte_output".

SHIFT "byte_output" "byte_prev1".
SHIFT "byte_prev1" "byte_prev2".
...
SHIFT "byte_prev7" "byte_prev8".

; DYNAMICS: at non-alpha bytes, all byte_prev ESs clear.
; Word events created when conjunction count >= 16 (tau=4).
```

Architecture: a depth-8 shift chain with word-boundary clearing. The output ES is 256 bytes. Eight previous-byte ESs form the shift chain (byte_output → byte_prev1 → ⋯ → byte_prev8). A single bigram LPP (byte_prev1 → byte_output) provides scoring. The word ES starts empty and grows.

### 2.1. Word-Boundary Clearing

At each non-alphabetic byte, all eight byte_prev ESs are cleared (support ← 0 for all events). The boundary byte itself is placed in byte_prev1. This ensures the shift chain holds only within-word context.

### 2.2. Threshold Creation

At each word boundary (transition from alphabetic to non-alphabetic), the experiment identifies the completed word by its full lowercase string (up to 63 characters, hashed via a word frequency table). When a word's occurrence count reaches the threshold $2^\tau = 16$, a new event is created in the word ES.

### 2.3. P-Programming Gap

In the current implementation, word identity is determined by an imperative hash table over word strings, not by the UM's own threshold creation mechanism on LPP entries. A fully P-programmed version would use a variable-length conjunction LPP: the joint event of all byte events at the word boundary would be a single LPP entry, and threshold creation on this LPP would reify the conjunction as a word event. The current LPP framework supports fixed two-input conjunctions (from, from2) but not variable-length conjunctions.

The word ES is therefore a result of the experiment—it records what was discovered—but it does not participate in prediction. The SN model captures the architecture (shift chain, bigram LPP) but the word discovery mechanism is in the C step loop, not in the SN.

## 3. Results

| Scale | Unique Words | Reified ($\geq 16$) | Token Coverage | Bigram bpc |
|---|---|---|---|---|
| 10K | 464 | 11 | 23.8% | 5.333 |
| 100K | 3,213 | 126 | 50.6% | 4.443 |
| 1M | 15,806 | 1,179 | 72.3% | 4.172 |
| 10M | 73,053 | 8,048 | 87.8% | 4.140 |

Table 1: Scaling of word discovery with data size. Threshold $\tau = 4$ ($2^4 = 16$ observations to reify). Bigram bpc is the within-word bigram model with boundary clearing.

### 3.1. Key Findings

1. **73K prediction confirmed.** At 10M bytes (1.42M word tokens), the model discovers 73,053 unique words, exactly matching the lexicon path paper's prediction.

2. **88% token coverage from 11% of types.** 8,048 reified words (11% of unique words) cover 87.8% of all word tokens. This is Zipf's law in action: a small number of frequent words dominate the token stream.

3. **Length distribution matches English.** The reified words peak at lengths 5–7, with a long tail to 18+. The distribution is consistent with English word length statistics applied to Wikipedia text.

4. **Top words are correct.** The top 30 at 10M are: the, of, and, in, a, to, quot, is, s, as, was, by, for, that, id, lt, gt, with, on, his, he, it, from, amp, are, or, an, http, at, be. This

includes English function words, Wikipedia markup tokens (quot, lt, gt, amp, http, id), and the possessive "s" (split from its host word by the apostrophe boundary).

5. **Bigram baseline with clearing.** The bigram model with boundary clearing scores 4.14 bpc at 10M. This is slightly worse than the standard wm-bigram (4.14 vs 4.17 at 1M without clearing, since clearing removes cross-word context). The clearing is not meant to improve bigram scoring—it exists to give the shift chain clean within-word state.

### 3.2. Comparison with Oracle

The 100-word oracle experiment (*English Context Neuron Results*) found the top 100 words covering a specific fraction of tokens. Here we discover 8,048 words covering 87.8%, without any oracle. The online word mixture from the previous work achieved $-0.552$ bpc gain at 10M; the current experiment provides the vocabulary that future word-level prediction experiments will use.

## 4. SN Format Conventions

The SN model follows §10 of the UMR Spec (`#umr_spec`). Key elements:

- All event spaces are declared with their full event lists.
- The word ES uses event names "word:*string*" for readability and downstream tooling.
- Support values on word events are $\lfloor \log_2(\text{count}) \rfloor$.
- The full 8-link shift chain is declared.
- Word-boundary clearing and threshold creation are documented in comments, as the SN grammar does not yet have dedicated syntax for these dynamics.

## 5. What This Experiment Does Not Do

1. No word-level LPPs. The discovered words are not used for prediction. Adding OBSERVE LPPs from word events to byte_output is Exp L2.
2. No memory trace. No AC encoding or decoding. The model trains online and produces a snapshot SN. Trace integration is Exp L3+.
3. No P-programming of word identity. The word hash table is imperative C code, not a UM mechanism. Making word discovery emerge purely from UM threshold creation requires variable-length conjunction support, which is a framework extension.

## 6. Reproduction

```
./hutter-make umr
./umr word-discover enwik9 10000000 4 word-discover-10M.sn
```

Block: `#umr_word_discover` (Hutter:UMR marker).
CLI dispatch: `#umr_raw_oversupport` ("word-discover" command).
Experience report: `#exp_l1_report`.