RNN to Universal Model Mapping

Working Document

Version: 20260130-v2

IMPORTANT: T values are LOG support, not probabilities. Support values do NOT sum to any constant within an ES. To get probabilities, apply softmax to T within each ES. See Section 5.

1 Key Insight: Event Spaces and Softmax

Every softmax indicates an event space. In the UM, **only positive support can exist**. The only downward pressure comes from competition *within* event spaces (via softmax).

The output layer is an ES: 256 bytes. We predict an output *distribution* $\in [0, 255]^{256}$ (support values), then softmax for probabilities. The internal event spaces remain to be discovered—this is the main experimental problem.

2 What We Are Actually Doing

1. **Compile** patterns into the RNN by training (opaque weights)
2. **Extract** patterns back out as events
3. **Discover** intermediate event spaces (partition of hidden state)
4. **Name** the events (interpretable features)

The input event names are trivial: "The input is 'a' " etc. The output events likewise. The intermediate ones we discover inductively.

We're building a **lexicon** and **grammar** as we go. What the RNN discovers as a black box, we rediscover in interpretable form.

The lexicon comes first—single-character Markov patterns are what the RNN learns most easily.

3 The Doubled-E Trick

Doubling the hidden state ($h_j \to \{h_j, \bar{h}_j\}$) creates a binary ES for each neuron. This is a **mechanical trick** to handle negative weights, not the natural factorization.

With 128 hidden neurons, we get 128 binary event spaces. This is the maximally fine-grained partition.

The goal: find **coarser partitions** that are semantically meaningful. Group hidden activations into ESs where:

- Activations within each ES compete via softmax
- Model accuracy is preserved
- ESs have interpretable meanings (lexical features, grammatical features)

4 RNN Architecture

Single-layer Elman RNN for byte-level prediction.

Spaces:

$x_t \in \{0, \ldots, 255\}$	(input byte—an ES)
$h_t \in \mathbb{R}^{128}$	(hidden state—ESs to be discovered)
$y_t \in \mathbb{R}^{256}$	(output logits—an ES)

Forward pass (one timestep):

$$h_t = \tanh(W_{xh}[:, x_t] + W_{hh} \cdot h_{t-1} + b_h)$$
$$y_t = \text{softmax}(W_{hy} \cdot h_t + b_y)$$

Parameters:

$W_{xh} \in \mathbb{R}^{128 \times 256}$	(input to hidden)
$W_{hh} \in \mathbb{R}^{128 \times 128}$	(hidden to hidden)
$W_{hy} \in \mathbb{R}^{256 \times 128}$	(hidden to output)
$b_h \in \mathbb{R}^{128}, \quad b_y \in \mathbb{R}^{256}$	(biases)

5 Universal Model Formalism

From CMP paper: $u = (E, T, P, f, \omega)$

Components:

- E = Event space (factored as $\prod_i E_i$)
- $T : E \to [0, 255]$ = Total thought (**LOG support** for each event)
- $P \subseteq E \times E \times [0, 255]$ = Pattern space (positive only!)
- $f_p : T \to T$ = Update function
- ω = Learning function

CRITICAL: T values are log support, not probabilities.

- $t = 20$ for all events in an ES $\neq$ $t = 1$ for all events (same probabilities after softmax, but different support levels)
- Support values do NOT sum to any constant
- To get probabilities: $p_i = \text{softmax}(t)_i = e^{t_i} / \sum_j e^{t_j}$

Standard update function:

$$(f_p(t))_j := \max_i \min(t_i, p_{ij})$$

Key constraint: Only positive patterns exist. Downward pressure comes exclusively from competition within event spaces (via softmax when converting to probabilities).

6 ESs Are The Only Non-Linearity

Critical insight: Without knowing the ESs, we cannot do a forward pass.

The UM forward pass structure:

1. Apply patterns (this is linear: max-min over pattern strengths)
2. Apply softmax within each ES (this is the non-linearity)

The softmax step is where competition happens. Without knowing which events belong to which ES, we don't know where to apply softmax.

The RNN's implicit ESs:

The RNN implements ESs implicitly through:

- tanh non-linearity (squashes to $[-1, 1]$)
- Negative weights (create downward pressure)
- Biases (shift decision boundaries)

Our task: find ESs such that softmax-within-ES produces **equivalent downward pressure** to what the RNN computes.

7 Remapping Negative Weights

Key insight: $\bar{h}_j$ ("not h_j") is a placeholder, not a real event.

In the doubled-E trick, negative weight $w_{ij} < 0$ becomes pattern $(e_i \to \bar{h}_j)$. But $\bar{h}_j$ just means "support for h_j being inactive."

With real ESs:

If h_j belongs to ES $= \{h_j, h_k, h_m\}$, then:

- Negative weight TO h_j should become positive weight TO h_k or h_m
- Supporting "not h_j" = supporting the competitors of h_j in its ES

Structure in weights reveals ESs:

If neurons j, k, m share an ES, we expect:

- Negative weights TO j correlate with positive weights TO k, m
- They play similar "roles"—same sources tend to activate one OR another
- They are mutually exclusive in activation patterns

This correlation structure in the weight matrices is a signal for discovering ESs.

The search:

Find ESs such that:

$$\text{softmax within ES} \approx \text{RNN's tanh + negative weights + biases}$$

When we find the right partition, the negative weight patterns "collapse" into positive patterns to ES-mates.

8 Doubled-E Baseline Implementation

Status: COMPLETE — Exact equivalence verified.

8.1 The Key Mathematical Insight

The tanh function and binary softmax are related:

$$\tanh(x) = 2 \cdot \sigma(2x) - 1$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid function.

For a binary ES $\{h_j, \bar{h}_j\}$ with support values t_h and $t_{\bar{h}}$:

$$p(h_j) = \text{softmax}([t_h, t_{\bar{h}}])_0 = \frac{e^{t_h}}{e^{t_h} + e^{t_{\bar{h}}}} = \sigma(t_h - t_{\bar{h}})$$

To achieve $\tanh(\text{pre}_h) = 2 \cdot p(h_j) - 1$, we need:

$$p(h_j) = \sigma(2 \cdot \text{pre}_h) \quad \Rightarrow \quad t_h - t_{\bar{h}} = 2 \cdot \text{pre}_h$$

8.2 Support Value Mapping

Given the RNN pre-activation $\text{pre}_h[j] = W_{xh}[j, x_t] + W_{hh}[j, :] \cdot h_{t-1} + b_h[j]$:

$$t(h_j) = 2 \cdot \max(0, \text{pre}_h[j]), \qquad t(\bar{h}_j) = 2 \cdot \max(0, -\text{pre}_h[j])$$

This ensures:

- If $\text{pre}_h > 0$: $t_h - t_{\bar{h}} = 2 \cdot \text{pre}_h - 0 = 2 \cdot \text{pre}_h$ ✓
- If $\text{pre}_h < 0$: $t_h - t_{\bar{h}} = 0 - 2|\text{pre}_h| = 2 \cdot \text{pre}_h$ ✓

8.3 Forward Pass Algorithm

1. **Compute pre-activation** (same as RNN):

$$\text{pre}_h[j] = b_h[j] + W_{xh}[j, x_t] + \sum_k W_{hh}[j, k] \cdot h_{t-1}[k]$$

2. **Convert to doubled-E support**:

$$t(h_j) = 2 \cdot \max(0, \text{pre}_h[j]), \quad t(\bar{h}_j) = 2 \cdot \max(0, -\text{pre}_h[j])$$

3. **Apply binary softmax**:

$$p(h_j) = \frac{e^{t(h_j)}}{e^{t(h_j)} + e^{t(\bar{h}_j)}}$$

4. **Map to hidden output** (for propagation to next layer):

$$h_{\text{out}}[j] = 2 \cdot p(h_j) - 1 \in [-1, 1]$$

5. **Compute output logits**:

$$y[o] = b_y[o] + \sum_j W_{hy}[o, j] \cdot h_{\text{out}}[j]$$

6. **Apply output softmax** for prediction distribution.

8.4 Why the Doubling Doesn't Propagate

A key insight: the doubling is **local to each layer's non-linearity**.

After softmax competition within each binary ES $\{h_j, \bar{h}_j\}$, we extract $p(h_j)$ and map it back to $[-1, 1]$ via $h_{\text{out}} = 2p - 1$. The $\bar{h}_j$ events have served their purpose (enabling competition) and are discarded.

The next layer sees only $h_{\text{out}}[j] \in [-1, 1]$, exactly as the RNN would produce. This is why we can defer natural ES discovery—the doubled-E trick gives us a working UM immediately.

8.5 Verification Results

Tested on two datasets with a trained model:

Dataset	RNN bpc	UM bpc	Difference
hutter.c (90 KB)	7.1887	7.1887	0.00%
enwik9 sample (100 KB)	6.3810	6.3810	0.00%

Step-by-step predictions also match exactly (same top prediction, same probability).

8.6 What This Enables

1. **Direct isomorphism**: RNN → UM without discovering natural ESs first
2. **Interpretable representation**: Patterns are explicit (source, destination, strength)
3. **Foundation for ES discovery**: Can now refine binary ESs into natural ones while preserving accuracy
4. **Validation baseline**: Any proposed ES partition must match or beat this bpc

9 Pattern Extraction (Doubled-E Baseline)

For the mechanical baseline using 128 binary ESs:

9.1 Event Spaces

$$\begin{aligned} E_I &= \{e_0, e_1, \ldots, e_{255}\} && |E_I| = 256 \text{ (input bytes)} \\ E_H &= \{h_0, \bar{h}_0, h_1, \bar{h}_1, \ldots\} && |E_H| = 256 \text{ (128 binary ESs)} \\ E_O &= \{o_0, o_1, \ldots, o_{255}\} && |E_O| = 256 \text{ (output bytes)} \end{aligned}$$

Each binary ES $\{h_j, \bar{h}_j\}$ competes via softmax. Both can have arbitrary support; softmax determines which "wins."

9.2 Pattern Sets

P_{xh}**: Input → Hidden** (from W_{xh})

For each $W_{xh}[h][i]$:

$$\begin{cases} (e_i \to h_h) \text{ strength } \propto W_{xh}[h][i] & \text{if } W_{xh}[h][i] > 0 \\ (e_i \to \bar{h}_h) \text{ strength } \propto |W_{xh}[h][i]| & \text{if } W_{xh}[h][i] < 0 \end{cases}$$

Negative weight means: "input i supports h being *inactive*." Competition within $\{h_h, \bar{h}_h\}$ via softmax resolves which dominates.

P_{hh}**: Hidden → Hidden** (from W_{hh}) — same logic.

P_{hy}**: Hidden → Output** (from W_{hy})

For each $W_{hy}[o][h]$:

$$\begin{cases} (h_h \to o_o) & \text{if } W_{hy}[o][h] > 0 \\ (\bar{h}_h \to o_o) & \text{if } W_{hy}[o][h] < 0 \end{cases}$$

Negative weight means: "h being *inactive* supports output o."

10 Discovering Natural Event Spaces

10.1 Observation Method (Activation Probing)

Run specific queries against the data (see `activation-probing.tex`). At matching positions, record hidden activations $h_t \in \mathbb{R}^{128}$.

Look for groups of neurons where:

- One neuron in the group tends to dominate (winner-take-all behavior)
- Other neurons in the group are suppressed when one is active
- The group has interpretable meaning

10.2 Weight Correlation Method

Analyze weight matrices for correlation structure:

- For each neuron j, find which other neurons have correlated incoming weights
- Specifically: neurons where negative weights from source i to j correlate with positive weights from i to k
- These (j, k) pairs are candidates for sharing an ES

10.3 Expected Discoveries (Lexicon First)

First-order patterns (single-char Markov) should appear as:

- ES for "previous char class" (vowel, consonant, digit, space, etc.)
- ES for "current position in word" (start, middle, end)
- ES for character-specific features

These form the **lexicon**—the vocabulary of events the model uses internally.

Higher-order patterns (grammar) build on the lexicon:

- ES for "in XML tag"
- ES for "in quoted string"
- ES for syntactic structures

11 Validation

11.1 Accuracy Metric

Bits-per-character is the primary metric. This measures cross-entropy loss: how well the predicted distribution matches actual next byte.

Same bpc $\Rightarrow$ same compression performance $\Rightarrow$ successful translation.

11.2 Interpretability Metrics

1. Can we name each ES?
2. Do the names make linguistic/structural sense?
3. Can we predict ES behavior on new inputs?

12 Implementation Status

Component	Status	Notes
Doubled-E pattern extraction	**Complete**	Exact RNN match verified
RNN vs UM comparison mode	**Complete**	`./hutter um` command
Activation probing framework	**Complete**	`./hutter probe` command
Weight correlation analysis	Not implemented	Next priority
ES discovery tooling	Not implemented	Depends on above

13 Next Steps

1. **Weight correlation analysis**: Find ES candidates by analyzing which neurons have anti-correlated incoming weights (negative to one $\Leftrightarrow$ positive to another).
2. **Activation probing experiments**: Use the probe tool to find contexts where specific neurons compete. Example queries:
   - After vowels vs. after consonants
   - Inside XML tags vs. outside
   - After space vs. mid-word
3. **Propose candidate ESs**: Based on correlation and probing data, propose groups of neurons that might share an ES.
4. **Test ES partitions**: Replace binary ESs with proposed groupings, verify bpc is preserved.
5. **Name the events**: Give interpretable names to discovered ESs, building the lexicon.