Dynamic Context-Overlap Patterns in Predictive-Coding Transformers: A Unified Architecture Using Bidirectional Energy Flow and Emergent Template Discovery

Ben Goertzel

September 14, 2025

Abstract

This speculative draft presents an architecture where linguistic patterns emerge dynamically from context overlaps rather than being fixed in advance. Building on Freeman's insight that words can be represented by the company they keep, we develop a system where: (i) state and attention unify into dynamic context patterns that evolve through bidirectional energy flow; (ii) association vectors are continuously recomputed from discovered overlaps rather than pre-stored; (iii) a backward disinhibition mechanism using noise enables novel pattern discovery; and (iv) hierarchical structure emerges through context-parameterized substitutions that preserve closure in a single comparison space. The architecture combines predictive-coding (PC) principles with lateral spreading dynamics, where each position maintains word-context sequential pairs with forward association weights and backward disinhibition weights. Templates and roles are not predefined but discovered through iterative energy minimization that finds overlapping contexts. Alternatively, template discovery can be delegated to symbolic pattern mining systems like Hyperon's Pattern Miner or Probabilistic Logic Networks (PLN), creating a hybrid neural-symbolic architecture where symbolic reasoning extracts structural patterns from the neural system's dynamic context representations. This creates a system capable of generating genuinely novel structures beyond its training data.

Running Example: Throughout this document, we'll use the sentence *"The chef that the critics praised cooks delicious pasta daily"* and systematic variants to illustrate how dynamic patterns emerge from context overlaps.

Contents

1 Introduction

Language models can benefit from recognizing that patterns are not fixed entities but emerge dynamically from context overlaps. Freeman [1] observed that words can be represented by their association neighborhoods, and that these representations can be composed so phrases remain in the same space as words. We extend this insight: rather than pre-computing these associations, they should emerge through bidirectional energy flow that discovers overlaps between contexts. This "dynamic closure" means that novel word combinations can generate new patterns not present in training data, while still maintaining comparability in a single space.

The architecture we propose unifies what are traditionally separate concepts—state and attention—into dynamic context patterns. Each position in the sequence maintains not a fixed hidden state, but an evolving pattern of word-context pairs with forward and backward weights. Through iterative energy minimization with noise-driven disinhibition, these patterns can discover novel associations by finding unexpected overlaps between contexts.

A key architectural choice is how templates and roles are discovered. While the core system can discover these through neural energy minimization, we also explore a hybrid approach where symbolic pattern mining systems—such as Hyperon's Pattern Miner operating on graph or metagraph representations, or Probabilistic Logic Networks (PLN) reasoning over discovered patterns—extract and generalize structural templates. This neural-symbolic integration allows the fluid dynamics of context overlap to be complemented by the precision and interpretability of symbolic pattern extraction.

Running Example

Consider our example sentence: *"The chef that the critics praised cooks delicious pasta daily"* This sentence illustrates how dynamic patterns emerge:

- **Novel association discovery**: Even if "chef" and "cooks" never co-occurred at this distance in training, their context overlaps (both associate with "kitchen", "food", "restaurant") allow the system to dynamically discover their connection
- **Bidirectional flow**: Forward energy from "chef" meets backward energy (with noise) from "cooks", creating a coherent pattern despite the intervening relative clause
- **Context-parameterized substitution**: "chef" $\leftrightarrow$ "baker" works because their context overlaps are similar, but the exact substitution mapping depends on the discovered patterns
- **Emergent templates**: The subject-verb-object structure isn't predefined but emerges from the overlap patterns—either through neural discovery or symbolic pattern mining

Minimal pairs we'll track:

- *"The critics that the chef praised cooks delicious pasta daily"* (role flip)
- *"The chef the critics praised cooks..."* (garden-path without "that")
- *"The chef that the critics praised is cooked for daily"* (passivization)

1.1 Technical overview

The core innovation is representing each position not as a vector but as a dynamic pattern $\Psi_t = \{(w_i, c_i, f_i, b_i)\}$, where each element consists of a word, its context, a forward association weight, and a backward disinhibition weight. The system discovers associations through:

Dynamic association discovery:

$$\phi_k(w, t) = \sum_{\text{contexts}} \text{overlap}(\text{Con}(w), \text{discovered\_contexts}[t]) \cdot \text{strength}[\text{context}]$$

where discovered_contexts accumulates during processing rather than being pre-computed.

Bidirectional energy flow:

$$E_{\text{forward}}[t] = \sum_i \text{MI}(w_t, w_i) \cdot \phi_i(\text{context}), \quad E_{\text{backward}}[t] = \sum_j \eta \cdot \text{noise}[j] \cdot \text{RCMI}(w_t, w_j) \cdot \phi_j(\text{future})$$

Pattern evolution:

$$\Psi_t^{(n+1)} = \Psi_t^{(n)} + \alpha(E_{\text{forward}} - E_{\text{backward}}) - \beta \nabla E_{\text{overlap}}$$

The composition operator becomes context-dependent:

$$\phi_k(t_1 \circ t_2; \Psi) = \sum_{i,j} B_\Psi(i, j)\, \phi_i(t_1; \Psi)\, \phi_j(t_2; \Psi)\, \phi_k\big(\gamma_\Psi(i, j)\big),$$

where B_Ψ and γ_Ψ are parameterized by the current pattern state Ψ rather than fixed.

Running Example

For "cooks pasta", the system dynamically discovers:

- $\phi(\text{"cooks"}; \Psi)$ emerges from overlaps: kitchen contexts + action contexts + food preparation contexts
- $\phi(\text{"pasta"}; \Psi)$ emerges from: Italian contexts + food contexts + dinner contexts
- The composition $\phi(\text{"cooks pasta"}; \Psi)$ creates new overlaps not present in either word alone
- Different pattern states Ψ yield different associations (context-dependent meaning)
- A symbolic pattern miner could extract the general template: ACTION(agent, patient) from multiple such instances

Note: All numerical values in examples are illustrative/hypothetical.

2 Dynamic Patterns from Context Overlaps

Traditional approaches pre-compute word associations from corpus statistics. We propose that associations should emerge dynamically as the system processes text, discovering overlaps between contexts that may not have been observed during training.

2.1 Context patterns as unified state-attention

Define a position's state not as a vector but as a pattern:

$$\Psi_t = \{(w_1, c_1, f_1, b_1), (w_2, c_2, f_2, b_2), \ldots\}$$

where:

- w_i is a word or word sequence
- c_i is its observed context (a set of surrounding patterns)
- f_i is the forward association strength (frequency-based)
- b_i is the backward disinhibition weight (noise-modulated)

This representation unifies state and attention: the "state" is the current pattern, and "attention" emerges from the overlap strengths between patterns.

Running Example

At position "cooks" in our sentence:

- Pattern includes: ("chef", {restaurant, kitchen}, 0.7, 0.3)
- Also includes: ("prepares", {food, dinner}, 0.6, 0.4)
- Overlap with "pasta" pattern: shared {food, dinner} contexts
- This overlap wasn't pre-computed but discovered during processing

The overlap strength becomes the "attention" weight naturally. *Note: Values are illustrative.*

2.2 Bidirectional energy flow with disinhibition

Energy flows both forward (standard association) and backward (through noise-driven disinhibition):

$$
\begin{aligned}
E_{\text{total}}[t] &= E_{\text{forward}}[t] + E_{\text{backward}}[t] + E_{\text{overlap}}[t] \\
E_{\text{forward}}[t] &= \sum_{i<t} f_i \cdot \text{overlap}(\Psi_t, \Psi_i) \\
E_{\text{backward}}[t] &= -\sum_{j>t} b_j \cdot \eta_j \cdot \text{overlap}(\Psi_t, \Psi_j) \\
E_{\text{overlap}}[t] &= -\lambda \sum_{i,j} \text{coherence}(\Psi_i \cap \Psi_j \cap \Psi_t)
\end{aligned}
$$

where η_j is noise applied at position j, enabling the discovery of novel associations.

Running Example

Processing "The chef that the critics praised cooks":

- **Forward flow**: "chef" $\rightarrow$ 0.8 energy (illustrative)
- **Backward flow**: "cooks" $\leftarrow$ 0.6 energy + 0.2 noise (illustrative)
- **Overlap discovery**: Despite separation, overlap in {kitchen, food} contexts creates connection
- **Novel pattern**: The bidirectional flow creates a pattern not in training: "chef-praised-cooks"

Note: Energy values are hypothetical.

2.3 Iterative pattern formation

Patterns evolve through iterative energy minimization:

for iteration = 1 to K **do**
  Compute context overlaps: $O_t = \text{find\_overlaps}(\Psi_t, \Psi_{1:T})$
  Forward propagation: $F_t = \sum_i f_i \cdot O_{ti}$

Backward propagation with noise: $B_t = \sum_j b_j \cdot (\eta_j + O_{tj})$
Pattern update: $\Psi_t \leftarrow \Psi_t - \alpha\nabla(F_t - B_t) + \beta \cdot \text{novel}(O_t)$
Discover new associations: if coherent(O_t) then add_to_patterns(O_t)
end for

The term novel(O_t) rewards discovering patterns not in the initial set.

3 Freeman's Dynamic Hierarchy Through Context-Parameterized Composition

Freeman's key insight was that phrases can stay in the same space as words through non-associative composition. We extend this: the composition itself should be parameterized by discovered context patterns.

3.1 Dynamic association vectors

Instead of static $\phi(w)$, associations are computed based on current discoveries:

$$\phi_k(w; \Psi) = \sum_{p \in \text{discovered}(\Psi)} \sum_{s \in S_p} A_\Psi(p, s) \cdot \text{RSMI}_\Psi(p, s; w, w_k),$$

where discovered(Ψ) are patterns found through overlap analysis, not predefined templates.

Running Example

Dynamic association for "chef":

- Initial: ϕ("chef") based on training contexts
- After discovering "critics praised" pattern: associations update to include evaluation contexts
- After discovering "cooks" connection: associations strengthen for action verbs
- Final ϕ("chef"; Ψ_{final}) differs from initial, includes novel associations

Note: This demonstrates true dynamism beyond fixed vectors.

3.2 Context-parameterized substitution

The combination code becomes dynamic:

$$\gamma_\Psi(i, j) = \sum_{\text{overlaps}} \text{similarity}(\text{overlap}, \text{Con}(i) \cap \text{Con}(j)) \cdot \text{novelty}[\text{overlap}]$$

This allows novel combinations based on discovered patterns:

Running Example

Two possible bracketings with dynamic scoring:

1. [[cooks[delicious pasta]]daily]: - Initial score: 0.7 (illustrative) - After overlap discovery: 0.85 (improved by finding cooking-food patterns)
2. [[cooks delicious][pasta daily]]: - Initial score: 0.3 (illustrative) - After overlap discovery: 0.25 (further penalized as no coherent overlap found)

Note: Scores are hypothetical but show dynamic adjustment.

3.3 Hierarchical emergence through cross-products

Hierarchy emerges by discovering that word pairs can substitute for singles when their context overlaps match:

$$\text{substitute}(w_i \circ w_j, w_k) \text{ if } \text{overlap}(\Psi_{ij}, \Psi_k) > \theta$$

But crucially, Ψ_{ij} is discovered dynamically, not pre-computed.

Running Example

Dynamic hierarchy discovery:

- "cooks pasta" develops pattern $\Psi_{\text{cooks-pasta}}$ through processing
- System discovers this overlaps with $\Psi_{\text{prepares-meal}}$ (not pre-defined)
- Creates dynamic substitution: [cooks pasta] $\leftrightarrow$ [prepares meal]
- This substitution wasn't in training but emerged from context overlaps

4 Predictive-Coding with Lateral Pattern Spreading

The predictive-coding framework provides the iterative machinery for pattern discovery. Instead of separate PC and Hopfield modules, we have unified lateral spreading dynamics.

4.1 Unified PC-lateral architecture

Each layer performs pattern evolution through energy minimization:

$$E^{(\ell)} = E^{(\ell)}_{\text{prediction}} + E^{(\ell)}_{\text{overlap}} + E^{(\ell)}_{\text{novelty}}$$

where:

$$E^{(\ell)}_{\text{prediction}} = \|\Psi^{(\ell-1)} - \text{predict}(\Psi^{(\ell)})\|^2 \tag{1}$$

$$E^{(\ell)}_{\text{overlap}} = -\sum_{t,s} \text{coherence}(\Psi^{(\ell)}_t \cap \Psi^{(\ell)}_s) \tag{2}$$

$$E^{(\ell)}_{\text{novelty}} = -\nu \sum_{t} \text{novel}(\Psi^{(\ell)}_t) \tag{3}$$

Running Example

Processing "The chef that the critics praised cooks":

- **Layer** ℓ: Pattern predicts object after "praised"
- **Layer** $\ell - 1$: Sees "cooks" (unexpected!)
- **Error**: $E_{\text{prediction}} = 3.2$ (illustrative)
- **Overlap search**: Discovers "chef"-"cooks" overlap via {kitchen, food}
- **Energy after discovery**: $E_{\text{total}} = 0.8$ (illustrative)

Note: Energy values are hypothetical.

4.2 Lateral spreading with word-context pairs

The lateral structure uses Freeman's suggested word-context sequential pairs:

$$\text{LateralPattern}_t = \{(w_i, c_i), (w_j, c_j), \ldots\}$$

with propagation:

$$\text{spread}(E, \eta) = E \cdot F + (E + \eta) \cdot B$$

where F are forward weights and B are backward (disinhibition) weights.

Running Example

Lateral spreading at "cooks":

- Forward spread from "chef": 0.7 via {kitchen} context (illustrative)
- Backward spread to "pasta": 0.6 + 0.2 noise via {food} context (illustrative)
- Creates coherent pattern: chef-cooks-pasta despite intervening words
- Pattern includes novel associations discovered through spreading

Note: Values are illustrative.

5 Template and Role Discovery Through Context Overlaps

Rather than predefining templates and roles, they emerge from discovered patterns. We explore two complementary approaches: neural discovery through energy minimization, and symbolic extraction through pattern mining.

5.1 Neural template emergence

Templates form when the system repeatedly discovers similar overlap patterns:

$$\text{Template}_p = \text{abstract}(\{\Psi_1, \Psi_2, \ldots, \Psi_n\})$$

where abstraction finds common structure across pattern instances.

Running Example

Template emergence in our sentence:

- Processing discovers: chef-cooks-pasta pattern
- Also discovers: critics-praised-chef pattern
- Abstraction finds common structure: agent-action-patient
- This becomes a template (not predefined!)
- Future processing can use this discovered template

5.2 Symbolic pattern mining alternative

Instead of or in addition to neural discovery, templates can be extracted by symbolic pattern mining systems that operate on the dynamic context representations:

Hyperon Pattern Miner approach: The Pattern Miner can analyze the graph or metagraph structure formed by context overlaps Ψ_t, using information-theoretic measures to extract frequently occurring patterns:

$$\text{Pattern}(G) = \text{argmax}_p \left[\text{freq}(p, G) \cdot \log \frac{\text{freq}(p, G)}{\prod_i \text{freq}(p_i, G)} \right]$$

where G is the graph representation of discovered context patterns.

PLN reasoning for pattern generalization: Probabilistic Logic Networks can then generalize discovered patterns through probabilistic inference:

$$P(\text{Template}|\text{Instances}) = \frac{P(\text{Instances}|\text{Template}) \cdot P(\text{Template})}{P(\text{Instances})}$$

This allows abstract templates to be inferred from specific pattern instances with associated confidence levels.

Running Example

Hybrid discovery process:

- Neural system discovers: chef-cooks-pasta, critics-praised-chef patterns
- Pattern Miner extracts: Common graph structure with 3 nodes and 2 directed edges
- PLN generalizes: TRANSITIVE(X, Y, Z) with confidence 0.85
- Template fed back to neural system for future use
- Symbolic system provides interpretable explanation of discovered structure

Note: This shows neural-symbolic cooperation.

5.3 Dynamic role assignment

Roles aren't labeled in advance but emerge from position in overlap patterns:

$$\text{role}(w, \Psi) = \text{position}(w, \text{abstract}(\Psi))$$

In the hybrid approach, the symbolic system can assign semantic labels to discovered roles based on their properties.

Running Example

Role discovery (illustrative probabilities):

- "chef" in chef-cooks pattern: position 1 → "initiator" role (0.8)
- "chef" in praised-chef pattern: position 2 → "target" role (0.7)
- Pattern Miner identifies: position 1 typically has agency features
- PLN infers: position 1 = AGENT with confidence 0.75
- Symbolic labels make roles interpretable

Note: Probabilities are hypothetical.

5.4 Self-organizing template library

Templates self-organize through repeated discovery:

while processing text **do**
  Discover patterns through neural overlap
  Extract patterns to graph/metagraph representation
  Apply Pattern Miner to find frequent structures
  Use PLN to generalize and assign confidence
  If structure confidence ¿ threshold: add to template library
  Feed templates back to neural system
  Prune templates that stop generating matches
end while

Running Example

Self-organization progression:

- **Early**: Only simple adjacency patterns
- **After 100 sentences**: Pattern Miner finds transitive structures (illustrative)
- **After 1000 sentences**: PLN generalizes to abstract templates (illustrative)
- **Convergence**: Library of 50 templates with confidence scores (illustrative)
- **Continuous**: Can still discover novel patterns not in library
- **Interpretable**: Symbolic system provides human-readable template descriptions

Note: Numbers are hypothetical examples.

6 Implementation Architecture

The complete architecture integrates dynamic neural patterns with optional symbolic reasoning:

6.1 Core pattern evolution loop

```
class DynamicPatternLayer:
    def forward(self, input_patterns):
        patterns = input_patterns

        for iteration in range(self.K):
            # Find overlaps (not pre-computed)
            overlaps = self.discover_overlaps(patterns)

            # Bidirectional energy
            forward = self.forward_flow(patterns, overlaps)
            backward = self.backward_flow_with_noise(patterns, overlaps)

            # Update patterns
            patterns = self.evolve_patterns(patterns, forward, backward)

            # Discover novel associations
            novel = self.extract_novel_patterns(patterns, overlaps)
            if self.is_coherent(novel):
                self.pattern_memory.add(novel)

            # Optional: symbolic pattern extraction
            if self.use_symbolic:
                templates = self.symbolic_miner.extract(patterns)
                patterns = self.integrate_templates(patterns, templates)
```

```
            # Dynamic hierarchy through substitution
            patterns = self.apply_dynamic_substitutions(patterns)

        return patterns
```

6.2 Dynamic association computation

```
def compute_association(word, patterns):
    # Not a lookup but a computation
    overlaps = find_all_overlaps(word.contexts, patterns)

    association = zeros(vocab_size)
    for overlap in overlaps:
        strength = overlap.coherence * overlap.novelty
        for word_k in overlap.words:
            association[word_k] += strength

    return association
```

6.3 Neural-symbolic interface

```
class NeuralSymbolicBridge:
    def __init__(self, pattern_miner, pln_reasoner):
        self.miner = pattern_miner
        self.reasoner = pln_reasoner

    def extract_templates(self, neural_patterns):
        # Convert neural patterns to graph
        graph = self.patterns_to_graph(neural_patterns)

        # Mine frequent patterns
        mined = self.miner.mine_patterns(graph)

        # Generalize with PLN
        templates = self.reasoner.generalize(mined)

        return templates

    def patterns_to_graph(self, patterns):
        # Convert dynamic patterns to graph representation
        nodes = [p.word for p in patterns]
        edges = [(p1, p2, overlap_strength(p1, p2))
                 for p1, p2 in pattern_pairs]
        return Graph(nodes, edges)
```

Running Example

Processing our sentence with hybrid architecture:

- **Neural phase**: Discovers chef-cooks-pasta patterns dynamically
- **Symbolic extraction**: Pattern Miner finds TRANS(X,Y,Z) structure
- **PLN generalization**: Infers general transitive template with 0.8 confidence
- **Feedback**: Template guides future neural processing
- **Novel discovery**: System still finds patterns beyond extracted templates

Note: Shows neural-symbolic cooperation.

7 Theoretical Analysis: Why Dynamic Patterns Help

The dynamic approach with optional symbolic reasoning offers advantages over fixed representations:

7.1 Information-theoretic perspective

Let Ψ_t be dynamic patterns, T_s be symbolically extracted templates, and Z_t be fixed features. The information about next token Y_t given context C_t satisfies:

$$I(Y_t; (\Psi_t, T_s)|C_t) \geq I(Y_t; \Psi_t|C_t) \geq I(Y_t; Z_t|C_t)$$

The symbolic templates T_s provide additional structured information that complements the fluid neural patterns.

Running Example

Information gain from hybrid approach (illustrative):

- Fixed features: $I(Y_t; Z_t|C_t) = 0.6$ bits
- Dynamic patterns: $I(Y_t; \Psi_t|C_t) = 0.9$ bits
- With symbolic templates: $I(Y_t; (\Psi_t, T_s)|C_t) = 1.1$ bits
- Gain from symbolic structure: 0.2 additional bits
- This translates to 30% total perplexity reduction (hypothetical)

Note: Values are illustrative.

7.2 Capacity and generalization

Dynamic patterns constrain the hypothesis space adaptively, with symbolic templates providing additional structure:

Proposition (Hybrid capacity control). Let $\mathcal{H}_{\text{neural}}$ be the neural pattern space and $\mathcal{H}_{\text{symbolic}}$ be the template space. The hybrid system operates in:

$$\mathcal{H}_{\text{hybrid}} = \mathcal{H}_{\text{neural}} \cap \text{consistent}(\mathcal{H}_{\text{symbolic}})$$

This intersection reduces capacity while maintaining expressiveness through the dynamic neural component.

7.3 Interpretability advantage

The symbolic component provides interpretable explanations:

$$\text{Explanation}(\Psi) = \text{PLN\_proof}(\text{Pattern\_Miner}(\Psi))$$

This allows the system to provide human-understandable justifications for its pattern discoveries.

8 Experimental Validation Approach

To validate these ideas, key experiments would include:

8.1 Measuring dynamic pattern formation

Track how patterns evolve during processing:

- Count novel associations discovered per sentence
- Measure overlap coherence over iterations
- Compare neural-only vs hybrid neural-symbolic
- Evaluate interpretability of symbolic templates

Running Example

Expected measurements on our sentence (illustrative):

- Novel associations discovered: 12 (neural), 15 (hybrid)
- Templates extracted: 3 by Pattern Miner
- PLN confidence in templates: 0.75-0.85
- Human interpretability score: 0.4 (neural) vs 0.8 (hybrid)

Note: Values are hypothetical targets.

8.2 Testing symbolic extraction quality

Evaluate the symbolic pattern mining:

- Precision/recall of extracted templates vs human annotations
- Generalization of PLN-inferred patterns to new sentences
- Consistency between neural discoveries and symbolic extractions

8.3 Validating bidirectional flow

Compare with and without backward disinhibition:

- Measure long-distance dependency resolution
- Track novel pattern discovery rate
- Evaluate on garden-path sentences
- Compare neural vs symbolic template usage

Running Example

Garden-path resolution "The chef the critics praised cooks":

- Neural only: 70% correct parse (illustrative)
- With backward flow: 85% correct parse (illustrative)
- With symbolic templates: 90% correct parse (illustrative)
- Novel patterns found: 5 neural, 2 additional symbolic (illustrative)

Note: Percentages are hypothetical predictions.

9 Conclusion

This architecture represents a fundamental shift from fixed to dynamic representations, with an optional but powerful integration of symbolic reasoning. By allowing patterns to emerge from context overlaps through bidirectional energy flow, the system can discover novel associations beyond its training data. The unification of state and attention into dynamic context patterns, combined with noise-driven disinhibition for backward flow, creates a system capable of true generalization.

The hybrid neural-symbolic approach offers additional advantages: symbolic pattern mining systems like Hyperon's Pattern Miner can extract interpretable templates from the neural system's discoveries, while PLN reasoning can generalize these patterns with associated confidence levels. This creates a best-of-both-worlds architecture where neural flexibility combines with symbolic precision and interpretability.

The key insights are: (1) patterns should be discovered, not predefined; (2) associations should be computed from overlaps, not stored; (3) bidirectional flow with noise enables novel discovery; (4) hierarchy emerges through dynamic substitution based on context overlap; (5) templates and roles can self-organize neurally or be extracted symbolically; and (6) neural-symbolic integration provides both flexibility and interpretability.

Running Example

Implementation Roadmap:
To implement this system, the team should:

1. Start with basic overlap discovery between adjacent words
2. Add bidirectional energy flow with simple noise
3. Implement pattern evolution through iterative updates
4. Enable dynamic association computation from overlaps
5. Add context-parameterized substitution for hierarchy
6. Implement neural template discovery through abstraction
7. Optionally integrate Pattern Miner for symbolic extraction
8. Optionally add PLN for template generalization
9. Test on novel combinations to verify true generalization

Success metrics:

- Novel patterns discovered per sentence ¿ 5
- Correct parse without pre-training templates ¿ 70%
- With symbolic extraction: ¿ 80%
- Perplexity reduction vs fixed: ¿ 20% (neural), ¿ 25% (hybrid)
- Template interpretability: ¿ 0.7 (human evaluation)
- All metrics are hypothetical targets

The result is a system that doesn't just select from pre-existing patterns but actively discovers new ones, achieving the dynamic flexibility that Freeman envisioned while maintaining the mathematical rigor needed for implementation. The optional symbolic component adds interpretability and structured reasoning without sacrificing the neural system's adaptability.

References

[1] R. J. Freeman. "Parsing using a grammar of word association vectors." *arXiv preprint* arXiv:1403.2152, 2014. Available at `https://arxiv.org/abs/1403.2152`.