# From Word-Association Grammars to Predictive-Coding Transformers: A Practical Bridge Using Hopfield-Style Lateral Dynamics and Pattern-Mined Templates

Ben Goertzel

September 13, 2025

**Abstract**

This speculative draft connects two lines of work: (i) Freeman's idea of representing words by the company they keep and *composing* those representations so that phrases and sentences stay in the same comparison space; and (ii) predictive-coding (PC) Transformers augmented with Hopfield-style lateral memory that search for low-energy, self-consistent internal states. We further generalize Freeman's notion of context from local adjacency to a library of *template patterns* with *roles* (e.g., subj/head/obj). Words can be near-substitutes when they fill the same role across templates, and near-complements when they co-fill different roles in the same template instance. This role-aware extension captures nonlocal grammatical structure while preserving Freeman's closure property and meshes naturally with PC inference and Hopfield retrieval.

**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 each component works in practice.

## Contents

# 1 Introduction

Language models can benefit from various complementary inductive biases. One interesting bias we explore here, expressed crisply by Freeman [1], says: represent a word by its association neighborhood and combine such representations so that phrases remain in the same space as words. This "closure" makes entire sentences directly comparable and encourages a notion of phrase structure because different merge orders produce different composite vectors. The second bias we explore here comes from predictive coding: each layer predicts what the layer below should look like, compares prediction to reality, and adjusts to reduce the error. If, in addition, the model can *retrieve* stable association patterns from memory, its internal states settle into coherent, grammar-like configurations.

**Running Example**

Consider our example sentence: *"The chef that the critics praised cooks delicious pasta daily"* This sentence illustrates several key challenges:

- **Non-local dependency**: "chef" (subject) and "cooks" (main verb) are separated by a relative clause
- **Multiple plausible associations**: "praised" could associate with "chef" or "critics"
- **Substitutability**: "chef" ↔ "baker", "pasta" ↔ "pizza"
- **Complementarity**: "cooks" naturally pairs with "pasta" as verb-object

**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)

We put these two biases/approaches together and then push the idea beyond adjacent context. Instead of only bigram or trigram cues, we let a library of *templates* with *roles* define what it means

for two words to "fit together" (complementarity) or to be "interchangeable" (substitutability). This captures nonlocal dependencies like subject-verb agreement and complements in a controlled way while keeping everything in a single comparison space. Practically, the Transformer supplies top-down predictions; the Hopfield lateral module pulls states toward plausible assemblies; and a light self-bootstrapping loop lets the model help discover the very templates it then uses.

The numbers given in various examples regarding this sentence are all made up and illustrative, we have not run the large-scale experiments that would be needed to actually make these calculations (yet).

### 1.1 Technical overview

To presage some of the details: Freeman [1] represents a word $w$ by a vector $\phi(w)$ whose components reflect association to vocabulary items via context statistics; a non-associative product extends $\phi$ from words to binary trees $t = (t_1, t_2)$:

$$\phi_k(t) \;=\; \sum_{i,j} \mathrm{MI}(i,j)\, \phi_i(t_1)\, \phi_j(t_2)\, \phi_k\big(\delta(i,j)\big),$$

where $\delta(i,j)$ indexes an observed combination code and MI weights pair salience. Closure holds because $\phi(t)$ lives in the same space as $\phi(w)$; non-associativity induces phrase structure.

**Running Example**

For "cooks pasta", we'd compute:

- $\phi$("cooks") = vector of associations (high for "eat", "prepare", "chef")
- $\phi$("pasta") = vector of associations (high for "Italian", "food", "delicious")
- $\phi$("cooks pasta") = composition using MI(cooks, pasta) and combination code

Different bracketings like "[cooks [delicious pasta]]" vs "[[cooks delicious] pasta]" yield different vectors, capturing phrase structure.

**Quantitative comparison:**

- [cooks[delicious pasta]]: $\mathbf{1}^\top \phi(t) = 0.82$
- [[cooks delicious]pasta]: $\mathbf{1}^\top \phi(t) = 0.31$

*Implementation note.* In our PC formulation we project hidden states into the same association space used by Freeman's composition and penalize their mismatch (Section 4.1). This makes the closure property an explicit part of the layer energy.

A PC Transformer layer predicts $h^{(\ell-1)}$ from $h^{(\ell)}$, penalizing prediction error as an energy term. A lateral Hopfield block retrieves pattern values $r_t = \sum_m \alpha_{t,m} v_m$ with softmax weights over keys; the state update $h_t \leftarrow h_t + U r_t$ reduces a Hopfield energy. To "Freemanize" this a bit, we can add: (i) an association head that predicts a sparse $\phi(w_t \mid \text{context})$; (ii) attention biases $b_{ij}$ derived from MI or learned association scores; (iii) a low-rank tri-linear composer that mirrors Freeman's product in the lateral path; and (iv) a Hopfield memory over combination codes $\delta(i,j)$.

We then generalize this approach to a *template-based* setting with a template library $P$ and role sets $S_p$. Define role-substitution statistics $\mathrm{RSMI}(p,s;w,x)$ for same-role co-usage and role-complement statistics $\mathrm{RCMI}(p,s,t;w,x)$ for cross-role co-usage. Build role-aware association vec-

tors

$$\phi_k(w) = \sum_{p \in P} \Big[ \sum_{s \in S_p} A_{\text{same}} \ \text{RSMI}(p, s; w, w_k) \ + \sum_{\substack{s,t \in S_p \\ s \neq t}} A_{\text{cross}} \ \text{RCMI}(p, s, t; w, w_k) \Big],$$

and a role-aware composition

$$\phi_k(t) = \sum_{p \in P} \sum_{\substack{s,t \in S_p \\ s \neq t}} \sum_{i,j} B_{p,s,t}(i,j) \ \phi_i(t_1) \ \phi_j(t_2) \ \phi_k\big(\gamma_{p,s,t}(i,j)\big),$$

with $B_{p,s,t}$ a role-conditioned association prior and $\gamma_{p,s,t}$ a role-aware combination code. Add role and co-filler heads, role-aware attention biases, and a small EM-like loop that alternates between using the model to propose soft template instances and re-estimating $(P, \text{RSMI}, \text{RCMI})$. Training combines LM loss with association, composition-consistency, parsing/assembly, role, and co-filler losses; efficiency comes from top-$K$ sparsity and low-rank factorization.

### 1.2 Motivating examples: why roles and templates help

**Nontechnical intuition.** Two different phenomena matter for language structure: (1) *substitutability*: words that can stand in the same slot ("dog" and "cat" as subjects); (2) *complementarity*: words that naturally co-occur in different slots of the same pattern ("eat" as a predicate with "pizza" as its object). Adjacency captures some of this, but it often misses long-distance links (e.g., subject–verb agreement across modifiers).

**Concrete mini-examples.**

1. **Non-local subject–verb.** "*The chef that the critics praised cooks* daily." Here, `chef` is the subject of `cooks`, but they are not adjacent. A role-aware template `(subj, head, obj)` captures the `subj(head)` link even when separated by a relative clause.
2. **Attachment ambiguity.** "*I saw the man with a telescope.*" The PP `with a telescope` can attach to the verb `saw` (instrument) or the noun `man` (modifier). Role-aware templates distinguish these roles and accumulate evidence for the correct structure, whereas adjacency bigrams alone are often ambiguous.

Running Example

In our sentence, templates help resolve:

- **Main clause template**: (subj="chef", head="cooks", obj="pasta")
- **Relative clause template**: (subj="critics", head="praised", obj="chef")
- These templates capture the correct dependencies despite word order mixing them

**Attachment ambiguity test:**

- "I saw the chef with a telescope" - RCMI(instrument, "saw", "telescope") = 0.73
- "I saw the chef with a moustache" - RCMI(modifier, "chef", "moustache") = 0.81

Bracketing energies: telescope-as-instrument = -2.1, moustache-as-modifier = -2.3 (lower is better).

**Takeaway.** We will keep Freeman's core advantage — *closure in one comparison space* — but define association using template *roles*, so both substitutability (same role) and complementarity (cross roles) are learned, including at a distance.

## 2 Freeman's Grammar of Word-Association Vectors

Freeman's key idea in [1] is to represent a word by the company it keeps. Two words are similar if they tend to appear in similar immediate contexts. Using these similarities, each word is embedded as a vector whose coordinates reflect how strongly it is associated with other words in the vocabulary. Phrases are then built by *combining* these vectors in a way that keeps the result in the same space as single words. Because the combination is not associative, different ways of bracketing a sentence produce different vectors, naturally encouraging a notion of phrase structure. A good parse is the one whose combined vector looks most like many valid words or word-groups, hence concentrating probability mass on plausible compositions. A practical lesson is the emphasis on *closure*: words and whole sentences live in the same comparison space, which makes scoring and search simpler.

### 2.1 Technical details

Let the vocabulary be $V = \{w_1, \ldots, w_N\}$. Define the context of a token $w$ as the set of adjacent word-pairs

$$\mathrm{Con}(w) := \{(w', w'') : \ \exists i \ \ X_i = w', \ X_{i+1} = w, \ X_{i+2} = w''\}.$$

Word similarity may be measured with an information-theoretic score, for example Lin-style similarity:

$$\mathrm{sim}(w, u) := \frac{2\, I(\mathrm{Con}(w) \cap \mathrm{Con}(u))}{I(\mathrm{Con}(w)) + I(\mathrm{Con}(u))},$$

and a word is represented by a vector of similarities

$$\phi(w) := (\phi_1(w), \ldots, \phi_N(w)) \quad \text{with} \quad \phi_i(w) = \mathrm{sim}(w, w_i).$$

**Running Example**

For our example words:

- $\phi$("chef") has high values for "cook", "restaurant", "kitchen", "food"
- $\phi$("cooks") has high values for "prepares", "chef", "kitchen", "food"
- $\phi$("pasta") has high values for "Italian", "sauce", "delicious", "food"

Note how "chef" and "cooks" share associations ("kitchen", "food"), indicating their semantic relationship even when not adjacent in our sentence.
**Role-preserving substitution invariance:**

- "chef" $\to$ "baker": $||\phi(\text{chef}) - \phi(\text{baker})||_2 = 0.23$
- "pasta" $\to$ "pizza": $||\phi(\text{pasta}) - \phi(\text{pizza})||_2 = 0.19$
- "cooks" $\to$ "prepares": $||\phi(\text{cooks}) - \phi(\text{prepares})||_2 = 0.21$

Small distances confirm substitutability within roles.

Extend $\phi$ from words to binary trees $t = (t_1, t_2)$ by a product that remains in the *same* space:

$$\phi_k(t) \;=\; \sum_{i,j} \mathrm{MI}(w_i, w_j)\; \phi_i(t_1)\; \phi_j(t_2)\; \phi_k(\delta(i,j)).$$

Here $\mathrm{MI}(\cdot,\cdot)$ weighs the salience of adjacent pairs, and $\delta(i,j)$ maps a pair $(w_i, w_j)$ to the index of their observed combination (or to 0 if unobserved). Because the product depends on bracketing, different trees yield different $\phi(t)$. A parse score can be defined by the total mass

$$\mathrm{score}(t) \;=\; \sum_{k=1}^{N} \phi_k(t),$$

and the best bracketing is the one that maximizes this score. Freeman notes that static similarity is context-insensitive and suggests moving toward *vectors of contexts* (context-conditioned association) as an improvement, while preserving the closure property.

**Running Example**

Two possible bracketings of "cooks delicious pasta daily":

1. [[cooks[delicious pasta]]daily] - treats "delicious pasta" as a unit
2. [[cooks delicious][pasta daily]] - unnatural grouping

The first bracketing scores higher because:

- MI("delicious", "pasta") = 0.72 (high, common bigram)
- MI("cooks", ["delicious pasta"]) = 0.68 (verb-object relation)
- MI("pasta", "daily") = 0.11 (low, uncommon pairing)

Total scores: Option 1 = 2.84, Option 2 = 1.23

# 3 Predictive-Coding Energy-Based Transformer with Lateral Hopfield Connections

Predictive coding treats each layer of a model as trying to predict the activity of the layer below; errors between predictions and observations are pushed upward to refine beliefs. An energy-based view says the whole network searches for low-energy states where predictions match inputs and internal associations agree. In a PC Transformer, standard self-attention and feedforward blocks make top-down predictions, while *lateral* Hopfield modules let tokens and spans recall patterns from memory and settle into consistent configurations. The result is a decoder that combines statistical context with retrieved structural patterns, improving stability and long-range coherence.

## Rough Formalization

Let $x_{1:T}$ be token embeddings, and let $h^{(\ell)}_{1:T}$ be hidden states at layer $\ell$. A predictive-coding layer produces a top-down prediction $\hat{h}^{(\ell-1)}$ of the layer below and minimizes a local energy

$$E^{(\ell)}_{\mathrm{PC}} \;=\; \frac{\lambda_e}{2} \left\| h^{(\ell-1)} - \hat{h}^{(\ell-1)}(h^{(\ell)}) \right\|_2^2 \;+\; R^{(\ell)}(h^{(\ell)}),$$

where $R^{(\ell)}$ collects regularizers or priors. The Transformer parameterization is used for $\hat{h}^{(\ell-1)}(\cdot)$ via attention and MLPs.

**Running Example**

Processing "The chef that the critics praised cooks":

- **Layer** $\ell$: Predicts that after "praised" we might see an object
- **Layer** $\ell - 1$: Actually sees "cooks" (unexpected!)
- **Error signal**: High prediction error $E_{\text{PC}} = 3.2$
- **Correction**: Model realizes "chef" (not "critics") is the subject of "cooks"
- **After correction**: Error drops to $E_{\text{PC}} = 0.8$

**Garden-path variant** "The chef the critics praised cooks":

- **Iteration 0**: Incorrectly brackets [[chef critics] praised], $E_{\text{PC}} = 4.1$
- **Iteration 1**: Closure energy signals mismatch, $E_{\text{clos}} = 2.3$
- **Iteration 2**: Role biases correct to [chef [critics praised]], $E_{\text{total}} = 1.2$

A lateral Hopfield module augments each layer with associative retrieval. Given queries $q_t = W_Q h_t^{(\ell)}$ and a memory of $M$ patterns $(k_m, v_m)$,

$$r_t \;=\; \sum_{m=1}^{M} \alpha_{t,m}\, v_m, \quad \alpha_{t,m} \propto \exp\left(\frac{q_t^\top k_m}{\tau}\right),$$

and $h_t^{(\ell)}$ is updated by a residual $h_t^{(\ell)} \leftarrow h_t^{(\ell)} + U r_t$. An energy for modern Hopfield retrieval can be written as

$$E_{\text{Hop}} \;=\; -\sum_t \log \sum_m \exp\left(\frac{q_t^\top k_m}{\tau}\right) \;+\; \frac{\lambda_v}{2} \sum_t \|h_t^{(\ell)} - (h_t^{(\ell)} + U r_t)\|_2^2,$$

and the layer-level energy is $E^{(\ell)} = E_{\text{PC}}^{(\ell)} + E_{\text{Hop}}$. Training uses next-token cross-entropy plus energy-based or denoising auxiliaries; inference performs a small number of iterative corrections (error feedback and lateral retrieval) before emitting logits.

Running Example

Hopfield retrieval for "cooks":

- **Query**: Hidden state at "cooks" position
- **Retrieved patterns**: - Pattern 1: (chef, cooks, food) - score = 0.91 - Pattern 2: (critics, cooks, ...) - score = 0.12 - Pattern 3: (chef, prepares, dish) - score = 0.67
- **Update**: Reinforces "chef" as subject interpretation

**Center-embedding stress test:** "The chef that the critics that the editors admired praised cooks..."

- Depth 1: Role saturation = 0.75, conflicts = 0.1
- Depth 2: Role saturation = 0.82, conflicts = 0.08
- Depth 3: Role saturation = 0.88, conflicts = 0.05

Hopfield retrieval successfully recovers deeper template fragments.

# 4 Integrating Freeman's Ideas into a PC Transformer

We import three practical ideas. First, give the model an extra head that predicts a word's *association neighborhood* rather than only a single next token. This helps it understand what else would make sense in the same context. Second, nudge attention and memory toward word pairs that corpora show are tightly linked, so the model naturally prefers plausible compositions. Third, when combining spans, keep their representation inside a single common space so sentences can be compared directly. All three fit predictive coding: top-down predictions become softer and more realistic, lateral Hopfield retrieval stabilizes plausible phrases, and the whole network searches for low-energy states that match both the data and association structure.

### Technical proposals

**A. Association-space auxiliary head.** Add a head that predicts $\phi(w_t \mid \text{context})$, a sparse distribution over words that are contextually substitutable at position $t$. Supervise it by constructing targets from corpus statistics or a learned teacher. Use the loss

$$\mathcal{L}_{\text{assoc}} \;=\; \sum_t \mathrm{KL}\Big(\tilde{\phi}_t \;\big\|\; \mathrm{softmax}(W_\phi h_t)\Big)\,,$$

combined with the standard language modeling loss.

Running Example

At position "cooks", the association head predictions differ by variant:
**Original:** "The chef that the critics praised cooks"

- High scores: "prepares" (0.31), "makes" (0.28), "serves" (0.22)
- Low scores: "criticized" (0.02), "praised" (0.01)

**Role-flip:** "The critics that the chef praised cooks"

- High scores: "write" (0.19), "review" (0.17), "complain" (0.15)
- Still present: "cooks" (0.08) - but much lower!

The association head correctly identifies that "critics" are unlikely to "cook".

**B. MI-biased attention.** In self-attention, add a learned or table-driven bias $b_{ij}$ to the attention logits favoring high mutual-information neighbors:

$$\alpha_{ij} \propto \exp\left(\frac{q_i^\top k_j}{\sqrt{d}} + \beta\, b_{ij}\right), \quad b_{ij} \approx \mathrm{MI}(\mathrm{token}_i, \mathrm{token}_j).$$

Use a sparse top-$K$ bias to keep computation stable; learn $\beta$.

Running Example

MI biases in our sentence variants:

- High MI: ("chef", "cooks") = 0.76, ("cooks", "pasta") = 0.71, ("delicious", "pasta") = 0.69
- Medium MI: ("critics", "praised") = 0.54, ("pasta", "daily") = 0.22
- Low MI: ("critics", "pasta") = 0.08, ("chef", "daily") = 0.11

**Impact on attention:** With $\beta = 0.5$, attention from "cooks" to "chef" increases from 0.18 to 0.42, while attention to "critics" decreases from 0.15 to 0.07.

**C. Closure-preserving composition in the lateral path.** Insert a bilinear or low-rank tri-factor module that mirrors Freeman's composition while staying in one space:

$$c_k = \sum_{i,j} \underbrace{\tilde{M}_{ij}}_{\text{learned or MI prior}} \underbrace{a_i}_{\phi_i(t_1)} \underbrace{b_j}_{\phi_j(t_2)} \underbrace{P_{k,\delta(i,j)}}_{\text{combination code}},$$

implemented with factorization (CP or Tucker) so it is efficient. Feed $c$ into the Hopfield queries or as a residual into $h$ to bias span fusion.

**D. Hopfield memory over combination codes.** Treat $\delta(i,j)$ as pattern codes stored in lateral memory. Keys are functions of the pair $(i,j)$ or of their embeddings; values are the embeddings of their observed combination. Retrieval weight is proportional to $a_i b_j \tilde{M}_{ij}$. This encourages the network to settle into attractors that correspond to plausible phrase combinations.

**E. Soft bracketing preference via energy.** For a short window, enumerate or sample a few bracketings. Let the composer produce $\phi(t)$ for each; define a local energy

$$E_{\text{brack}} \;=\; -\log \sum_{\text{trees } t} \exp\big(\gamma\, \mathbf{1}^\top \phi(t)\big),$$

so higher total mass receives lower energy. Backpropagate through a differentiable relaxation; at inference, a few refinement steps bias the internal state toward good bracketings.

Running Example

Evaluating bracketing for "cooks delicious pasta":

- Option 1: [cooks[delicious pasta]] - Score: 0.82, Energy: -1.89
- Option 2: [[cooks delicious]pasta] - Score: 0.31, Energy: -0.71

**Passivization test:** "The chef that the critics praised is cooked for daily"

- Bracketing: [chef [critics praised]] [is [cooked for]] daily
- Role reassignment: chef = patient (not agent), Energy: -1.65
- System correctly handles morphological cues

**F. Aggressive substitution as augmentation.** During training, replace a token with a sample from its association vector $\tilde{\phi}_t$ at rate $p$. Apply a consistency loss so span-level association vectors remain stable when meaning should be preserved, and a contrastive loss so the model detects when substitution changes meaning.

Running Example

**Quantified substitution trials:**
Original: "The chef that the critics praised cooks delicious pasta daily"
Substitutions and their effects:

- "chef" → "baker": - $\mathbf{1}^\top\phi$(sentence) change: 0.84 → 0.82 (-2.4%) - Template saturation: 0.91 → 0.89 (-2.2%) - Cross-entropy at "pasta": 2.31 → 2.34 (+1.3%)
- "pasta" → "pizza": - $\mathbf{1}^\top\phi$(sentence) change: 0.84 → 0.81 (-3.6%) - Template saturation: 0.91 → 0.88 (-3.3%) - Cross-entropy at next position: 2.45 → 2.49 (+1.6%)
- "cooks" → "prepares": - $\mathbf{1}^\top\phi$(sentence) change: 0.84 → 0.83 (-1.2%) - Template saturation: 0.91 → 0.90 (-1.1%) - Cross-entropy at "pasta": 2.31 → 2.32 (+0.4%)

All substitutions preserve structure (low variance), confirming role-based invariance.

**G. Sentence-level comparability for memory.** Pool association-space vectors to form a single sentence trace $s = \text{Pool}(\phi_{1:T})$. Train $s$ to reconstruct token distributions and span associations. Use $s$ as a retrieval key in the Hopfield module to improve long-range prediction.

**H. Context-conditioned associations.** Replace static $\phi(w)$ with $\phi(w \mid \text{context})$ by conditioning the association head on local features (position, neighbors, provisional parse). A small hypernetwork $H$ can modulate $W_\phi$: $W_\phi \leftarrow H(\text{context}) \cdot W_\phi$.

**I. Multi-task objective.** The overall loss combines

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda_1 \mathcal{L}_{\text{assoc}} + \lambda_2 \mathcal{L}_{\text{comp}} + \lambda_3 \mathcal{L}_{\text{brack}} + \lambda_4 \mathcal{L}_{\text{denoise}},$$

where $\mathcal{L}_{\text{comp}}$ aligns composed $\phi(t_1 \circ t_2)$ with Hopfield-retrieved combination codes, and $\mathcal{L}_{\text{denoise}}$ supports iterative PC corrections.

**J. Practicalities.** Use sparse targets (top-$K$) for $\phi$, approximate MI with learned embeddings, and factorize tensors for efficiency. Non-associativity emerges from iterative span fusion rather than requiring a hard parser.

### 4.1 Making the PC–closure link explicit

**Nontechnical summary.** Freeman's closure says: compose in such a way that words, phrases, and sentences are comparable in the same space. To exploit this with predictive coding (PC), we must *measure prediction errors in that same space*. Concretely, each layer produces a top-down prediction not only for tokens but also for their association vectors. The PC update then reduces a single energy that includes an *association-space mismatch* term, nudging internal states so that predicted and composed association vectors agree.

**Technical details.** Let $h^{(\ell)}$ be hidden states at layer $\ell$, and let $\Pi = W_\phi$ be a learned projection from hidden states into the association space. Define the predicted association vector $\hat{\phi}^{(\ell)} = \Pi\, h^{(\ell)}$. Let $\phi_{\text{comp}}^{(\ell-1)}$ be the composition (Freeman or template-based) computed from the layer below on the current bracketings/spans. We add an explicit *closure energy*

$$E_{\text{clos}}^{(\ell)} \;=\; \frac{\lambda_\phi}{2} \,\big\|\, \hat{\phi}^{(\ell)} \;-\; \phi_{\text{comp}}^{(\ell-1)} \,\big\|_2^2.$$

The full layer energy is

$$E^{(\ell)} \;=\; E_{\text{PC}}^{(\ell)} \;+\; E_{\text{Hop}}^{(\ell)} \;+\; E_{\text{clos}}^{(\ell)} \;+\; E_{\text{tmpl}}^{(\ell)}\,,$$

where $E_{\text{PC}}^{(\ell)}$ penalizes prediction error in hidden space, $E_{\text{Hop}}^{(\ell)}$ is the Hopfield retrieval energy, and $E_{\text{tmpl}}^{(\ell)}$ encourages role/template consistency when templates are used.

Running Example

PC correction for "cooks pasta":

- **Predicted** $\hat{\phi}$: Based on hidden states, expects verb-object pattern
- **Composed** $\phi_{\text{comp}}$: Freeman composition of "cooks" and "pasta"
- **Mismatch**: $||\hat{\phi} - \phi_{\text{comp}}||_2 = 0.42$ initially
- **Correction**: Hidden states adjust, mismatch drops to 0.11
- **Energy**: $E_{\text{clos}} = 0.09$ after convergence

**Perplexity correlation with structure:** For each variant, measuring $I(Y_t; Z_t|C_t)$ where $Z_t$ = (association features, role posteriors):

- Original sentence: $I = 0.82$ bits, PPL reduction = 22%
- Role-flip variant: $I = 0.31$ bits, PPL reduction = 8%
- Garden-path: $I = 0.45$ bits after correction, PPL reduction = 14%

**Mechanics of one PC correction step.** Given $h^{(\ell)}$, compute $\hat{\phi}^{(\ell)} = \Pi h^{(\ell)}$ and $\phi_{\text{comp}}^{(\ell-1)}$ from the current spans. Form the association error $e_\phi = \hat{\phi}^{(\ell)} - \phi_{\text{comp}}^{(\ell-1)}$ and update

$$h^{(\ell)} \leftarrow h^{(\ell)} - \eta\, \Pi^\top e_\phi \;+\; (\text{PC hidden error term}) \;+\; (\text{Hopfield retrieval term}).$$

Because the gradient flows through $\Pi^\top$, this step *directly* moves hidden states so that top-down predictions for words and phrases agree in the *same* association space. Thus, minimizing the combined PC energy enforces Freeman-style closure operationally, not just conceptually.

## 5 Generalizing From Adjacency to Role-Aware Template Patterns

We now explore a more adventurous "neural-symbolic" extension of the above approach.

Freeman's original approach represents each word by the company it keeps and combines word vectors so that phrases stay in the same comparison space as single words. This section generalizes the notion of "context" beyond adjacent neighbors. Instead of looking only at bigrams or trigrams, we consider a library of *template patterns* with *argument roles* (for example, a simple transitive pattern with roles `subj`, `head`, `obj`). Two words are then associated if they *co-fill* roles in the same template instance (complementarity) or if they tend to fill the *same* role across many instances (substitutability). This preserves Freeman's key advantages—closure in one space, aggressive substitution, and non-associative composition—while capturing long-range, non-adjacent structure such as subject–verb links at a distance. It also aligns naturally with the predictive-coding Transformer architecture discussed earlier: top-down predictions can propose role assignments, lateral Hopfield memory can retrieve plausible template completions, and the whole system can refine itself by searching for low-energy, role-consistent configurations. See Freeman (2014) for the original closure-preserving composition idea.[1]

[1] R. J. Freeman, "Parsing using a grammar of word association vectors," arXiv:1403.2152 (2014). URL: `https://arxiv.org/abs/1403.2152`.

Running Example

Templates for our sentence variants:
**Original:** "The chef that the critics praised cooks delicious pasta daily"

- Main: Trans(subj="chef", head="cooks", obj="pasta", mod="daily")
- Relative: Trans(subj="critics", head="praised", obj="chef")
- Assembly energy: -3.2 (low conflict)

**Role-flip:** "The critics that the chef praised cooks delicious pasta daily"

- Main: Trans(subj="critics", head="cooks", obj="pasta")
- Relative: Trans(subj="chef", head="praised", obj="critics")
- Assembly energy: -1.1 (semantic mismatch: critics rarely cook)

**Passivized:** "The chef that the critics praised is cooked for daily"

- Main: Passive(patient="chef", aux="is", pred="cooked for")
- Relative: Trans(subj="critics", head="praised", obj="chef")
- Assembly energy: -2.8 (morphology correctly handled)

### 5.1 Template statistics: a simple view before symbols

**Simple counts first.** Let $P$ be a small library of patterns (e.g., `(subj, head, obj)`). For a word $w$ and a role $s$, maintain a count $C_{\text{role}}(p, s; w)$: how often $w$ fills role $s$ in template $p$. For two words $w, x$:

- Same-role co-usage: $C_{\text{same}}(p, s; w, x)$ counts how often $w$ and $x$ are both observed in role $s$ of template $p$ (substitutability).
- Cross-role co-usage: $C_{\text{cross}}(p, s, t; w, x)$ counts how often $w$ fills role $s$ and $x$ fills role $t \neq s$ in the *same* instance of template $p$ (complementarity).

Running Example

Template statistics from corpus:

- $C_{\text{role}}(\text{Trans}, \text{subj}; \text{“chef”}) = 450$ (chef as subject)
- $C_{\text{role}}(\text{Trans}, \text{subj}; \text{“baker”}) = 380$ (baker as subject)
- $C_{\text{role}}(\text{Trans}, \text{subj}; \text{“critics”}) = 120$ (critics as subject)
- $C_{\text{same}}(\text{Trans}, \text{subj}; \text{“chef”}, \text{“baker”}) = 85$ (both as subjects)
- $C_{\text{cross}}(\text{Trans}, \text{subj}, \text{head}; \text{“chef”}, \text{“cooks”}) = 120$
- $C_{\text{cross}}(\text{Trans}, \text{subj}, \text{head}; \text{“critics”}, \text{“cooks”}) = 3$ (rare!)
- $C_{\text{cross}}(\text{Trans}, \text{head}, \text{obj}; \text{“cooks”}, \text{“pasta”}) = 95$

These capture semantic relationships that adjacency misses.

**Why adjacency is not enough.** Adjacency is a good *special case* (next paragraph), but it cannot, by itself, prefer the correct subject for a distant verb or resolve PP attachment. The template counts above accumulate precisely the missing long-range evidence.

**From counts to scores (now the symbols).** Define smoothed, pointwise log-odds:

$$\text{RSMI}(p, s; w, x) \;=\; \log \frac{C_{\text{same}}(p, s; w, x) + \alpha}{C_{\text{role}}(p, s; w)\, C_{\text{role}}(p, s; x)}, \quad \text{RCMI}(p, s, t; w, x) \;=\; \log \frac{C_{\text{cross}}(p, s, t; w, x) + \alpha}{C_{\text{role}}(p, s; w)\, C_{\text{role}}(p, t; x)}.$$

A word's association vector remains Freeman-style:

$$\phi_k(w) \;=\; \sum_{p \in P} \left[ \sum_{s \in S_p} A_{\text{same}}\, \text{RSMI}(p, s; w, w_k) \;+\; \sum_{\substack{s,t \in S_p \\ s \neq t}} A_{\text{cross}}\, \text{RCMI}(p, s, t; w, w_k) \right].$$

This preserves *closure*: words, phrases, and sentences all live in the same space.

**Lemma (Adjacency MI as a template special case).** Let $p_{\text{Adj}}$ be a 2-role template with roles $L$ (left) and $R$ (right). If we identify bigram counts with $C_{\text{cross}}(p_{\text{Adj}}, L, R; w_i, w_j)$, then standard adjacency MI is proportional to $\text{RCMI}(p_{\text{Adj}}, L, R; w_i, w_j)$. Hence RCMI strictly generalizes adjacency MI: choosing $P = \{p_{\text{Adj}}\}$ recovers Freeman's adjacency prior; enlarging $P$ adds non-local and typed relations without breaking closure.

**Toy numbers (what RSMI/RCMI capture).** Consider a transitive template with roles $(\texttt{subj}, \texttt{head}, \texttt{obj})$. Counts: $C_{\text{role}}(\texttt{subj}; \texttt{dog}) = 120$, $C_{\text{role}}(\texttt{subj}; \texttt{cat}) = 110$, $C_{\text{role}}(\texttt{head}; \texttt{eat}) = 300$, $C_{\text{role}}(\texttt{obj}; \texttt{pizza}) = 150$. Co-usage: $C_{\text{same}}(\texttt{subj}; \texttt{dog}, \texttt{cat}) = 40$ (high substitutability); $C_{\text{cross}}(\texttt{head}, \texttt{obj}; \texttt{eat}, \texttt{pizza}) = 90$ (strong complementarity). Even if `dog` and `eat` are rarely adjacent due to modifiers, the cross-role signal remains strong, whereas adjacency alone would understate their relation.

### 5.2 Technical formulation

Let $P = \{p\}$ be a set of templates. Each template $p$ has a set of roles $S_p$ (e.g., $\{\texttt{subj}, \texttt{head}, \texttt{obj}\}$). A *template instance* is a tuple $I = (p, \{s \mapsto w\})$ that binds tokens to roles.

**Association vectors that preserve closure.** Generalizing Freeman's $\phi(w)$, represent $w$ by a vector over the vocabulary $V$ whose $k$-th component aggregates role-aware associations:

$$\phi_k(w) \;=\; \sum_{p \in P} \left[ \sum_{s \in S_p} A_{\text{same}}\, \text{RSMI}(p, s; w, w_k) \;+\; \sum_{\substack{s,t \in S_p \\ s \neq t}} A_{\text{cross}}\, \text{RCMI}(p, s, t; w, w_k) \right],$$

with weights $A_{\text{same}}, A_{\text{cross}} \geq 0$. As in Freeman, $\phi(\cdot)$ for words, phrases, and whole sentences all live in the *same* space, preserving closure and comparability.

**Running Example**

Role-aware association vectors:

- $\phi$("chef"): High for "baker" (0.72), "cook" (0.68) (same-role subjects), "cooks" (0.61), "prepares" (0.58) (cross-role complements)
- $\phi$("cooks"): High for "prepares" (0.74), "makes" (0.71) (same-role verbs), "chef" (0.61), "pasta" (0.59) (cross-role complements)
- $\phi$("critics"): High for "reviewers" (0.77), "judges" (0.73) (same-role), "praised" (0.52), "reviewed" (0.49) (cross-role)
- $\phi$("pasta"): High for "pizza" (0.79), "bread" (0.71) (same-role objects), "cooks" (0.59), "delicious" (0.55) (cross-role complements)

**Role-aware composition operator.** Freeman's binary, non-associative composition used a pairwise mutual-information prior and a "combination code" $\delta(i, j)$. We lift both to be role-aware. Let $B_{p,s,t}(i, j)$ score the plausibility that vocabulary items $w_i$ and $w_j$ fill roles $(s, t)$ of template $p$; a simple choice is an exponentiated, normalized version of RCMI. Let $\gamma_{p,s,t}(i, j)$ map $(i, j)$ into a *role-aware* combination code (analogous to $\delta$). For a binary tree $t = (t_1, t_2)$,

$$\phi_k(t) \;=\; \sum_{p \in P} \sum_{\substack{s,t \in S_p \\ s \neq t}} \sum_{i,j} B_{p,s,t}(i, j)\; \phi_i(t_1)\; \phi_j(t_2)\; \phi_k\big(\gamma_{p,s,t}(i, j)\big).$$

Different merge orders generally yield different vectors (non-associativity), which naturally induces phrase structure.

**Local assembly score for parsing.** For a span represented by $t$, define a score that rewards high mass and role consistency:

$$\text{score}(t) \;=\; \sum_{k} \phi_k(t) \;+\; \lambda_{\text{sat}}\, \text{Saturation}(t) \;-\; \lambda_{\text{conf}}\, \text{Conflict}(t),$$

where Saturation measures filled, compatible roles of some $p \in P$ and Conflict penalizes incompatible role assignments (e.g., two heads). A differentiable softmax over a small set of candidate merges per span can be used during training.

Running Example

Scoring the complete parse for variants:
**Original:**

- Main template saturation: subj(chef) + head(cooks) + obj(pasta) = 0.91
- Relative template saturation: subj(critics) + head(praised) + obj(chef) = 0.88
- Conflict: 0.02 (minimal)
- Total score: 3.42

**Role-flip:**

- Main template saturation: subj(critics) + head(cooks) + obj(pasta) = 0.41 (poor fit!)
- Relative template saturation: subj(chef) + head(praised) + obj(critics) = 0.82
- Conflict: 0.18 (semantic clash)
- Total score: 1.87

The original parse scores much higher due to semantic coherence.

### 5.3 Integration into a PC Transformer with lateral Hopfield memory

We add heads that predict role labels and association neighborhoods, bias attention toward pairs likely to co-fill roles, and use Hopfield memory to retrieve template fragments that stabilize plausible assemblies. Predictive-coding iterations compare top-down role and association predictions with bottom-up evidence and update the hidden states to reduce residuals, moving the network toward low-energy, role-consistent states.

**Technical proposals.**

1. **Association head (relational).** Predict a sparse $\phi(w_t \mid \text{context})$ at each position $t$ and train with a KL loss against corpus-derived targets.
2. **Role head.** Predict a distribution over $(p, s)$ for each token. Optionally predict co-fillers for other roles of $p$ using a second head trained against RCMI-based targets.

Running Example

Role head predictions across variants:
**Original:**

- "chef": P(Trans, subj) = 0.71, P(Trans, obj) = 0.19 (in relative)
- "critics": P(Trans, subj) = 0.82
- "cooks": P(Trans, head) = 0.91
- "pasta": P(Trans, obj) = 0.87

**Role-flip:**

- "critics": P(Trans, subj) = 0.43, P(Trans, obj) = 0.41 (confused!)
- "chef": P(Trans, subj) = 0.78
- "cooks": P(Trans, head) = 0.52 (uncertain due to subject mismatch)

3. **Role-aware attention bias.** Add an additive bias to attention logits,

$$\text{bias}(i \to j) \;=\; \beta_1 \,\text{prior}_{\text{same}}(i,j) \;+\; \beta_2 \,\text{prior}_{\text{cross}}(i,j),$$

where the priors are sparse top-$K$ summaries of RSMI and RCMI; learn $\beta_1, \beta_2$.

4. **Lateral Hopfield memory over templates.** Store template prototypes and partially filled frames as patterns. Keys encode $(p, \text{role pattern})$ with masks; values encode combination codes $\gamma_{p,s,t}(i,j)$ and partial assemblies. Queries concatenate the current hidden state, role posteriors, and association vector. Retrieval nudges states toward role-consistent, saturated frames.

Running Example

Hopfield template retrieval:

- **Query**: State at "cooks" + role distribution
- **Retrieved**: Transitive template with subj=chef, head=cooks, obj=?
- **Prediction**: Likely obj fillers (pasta=0.31, food=0.22, meal=0.18)
- **Update**: Reinforces transitive interpretation

**Center-embedding stress test:** "The chef that the critics that the editors admired praised cooks..."

- Depth 1: Role saturation = 0.75, conflicts = 0.10, retrieval precision = 0.82
- Depth 2: Role saturation = 0.82, conflicts = 0.08, retrieval precision = 0.87
- Depth 3: Role saturation = 0.88, conflicts = 0.05, retrieval precision = 0.91

Hopfield successfully maintains structure even with deep embedding.

5. **Energy terms.** Add a template energy

$$E_{\text{tmpl}} \;=\; \sum_{\text{spans}} \left[-\tau_1 \,\text{role-posteriors} \;-\; \tau_2 \,\text{RCMI consistency} \;+\; \tau_3 \,\text{role-conflicts}\right],$$

and minimize $E = E_{\text{PC}} + E_{\text{Hop}} + E_{\text{tmpl}}$ with a few predictive-coding correction steps per layer.

6. **Losses.** Train with

$$\mathcal{L} \;=\; \mathcal{L}_{\text{LM}} + \lambda_a \,\mathcal{L}_{\text{assoc}} + \lambda_r \,\mathcal{L}_{\text{role}} + \lambda_c \,\mathcal{L}_{\text{cofill}} + \lambda_{\text{comp}} \,\mathcal{L}_{\text{composition}} + \lambda_{\text{parse}} \,\mathcal{L}_{\text{assembly}}.$$

### 5.4 Self-bootstrapping templates

A simple EM-like loop lets the model help induce the templates it uses:

1. **Warm start.** Initialize with adjacency-based priors (as in Freeman) to build initial $\phi$ and weak role estimates.
2. **E-like step.** Run the model and collect soft template instances (which $p$, which roles, which co-fillers) from role heads, attention patterns, and Hopfield retrievals.
3. **M-like step.** Re-estimate $P$, prune low-support templates, split overly broad ones, and refresh RSMI/RCMI tables.
4. **Iterate.** Update biases and association targets and continue training.

**Running Example**

Bootstrapping progression:

- **Iteration 1**: Discovers basic transitive template from adjacency
- **Iteration 2**: Refines roles, separates relative clauses
- **Iteration 3**: Learns modifier attachments, temporal adverbs
- **Iteration 4**: Discovers passive construction from "is cooked for"
- **Convergence**: Stable template library with Trans, Relative, Passive, PP-attachment

### 5.5 Role-aware augmentation and invariances

- **Role-preserving substitution.** Replace a token with a top-$K$ same-role substitute (high RSMI) and enforce stability of span-level associations.
- **Role-complement perturbation.** Swap a co-filler in another role with a high-RCMI alternative and reward frame saturation with low conflict.
- **Contrastive guards.** When substitutions should change meaning, impose a margin between original and perturbed association vectors.

Running Example

Training augmentations with measured impacts (illustrative values):

- Replace "chef" → "baker" (same role): $\Delta\text{PPL} = +1.3\%$, structure preserved (hypothetical)
- Replace "pasta" → "pizza" (same role): $\Delta\text{PPL} = +1.6\%$, structure preserved (hypothetical)
- Replace "cooks" → "criticizes" (different role): $\Delta\text{PPL} = +43\%$, structure broken (contrastive loss applied) (hypothetical)
- Replace "daily" → "weekly" (same role): $\Delta\text{PPL} = +0.8\%$, minor change (hypothetical)

These help the model learn robust role representations while detecting meaningful changes. *Note: All PPL changes are illustrative/hypothetical.*

### 5.6 Practical notes and evaluation

Keep association targets sparse (top-$K$ per token), factorize multi-role composition with low-rank tensors, and store only high-support combination codes in memory. Evaluate on perplexity, robustness to role-preserving substitutions, unsupervised constituency or UD-like probes, frame saturation vs. conflict, and retrieval precision for co-fillers. Ablate role-aware biases and Hopfield modules to isolate gains.

### 5.7 Why this helps

Role-aware templates separate *substitutability* (same role) from *complementarity* (different roles), capturing long-range dependencies while preserving Freeman's closure. The predictive-coding Transformer supplies the iterative inference machinery, and Hopfield memory stabilizes template assemblies. Together they yield a practical, stronger generalization of Freeman's original program.

Running Example

**Final parse representation and perplexity gains:**
Our complete sentence variants with baseline Transformer PPL = 42.3:
**Original:** "The chef that the critics praised cooks delicious pasta daily"

- Composed $\phi$(sentence) with high culinary + evaluation associations
- Freeman+PC PPL = 33.1 (22% reduction)
- $I(Y_t; Z_t|C_t) = 0.82$ bits average

**Role-flip:** "The critics that the chef praised cooks delicious pasta daily"

- Semantic mismatch detected through low RCMI(critics, cooks)
- Freeman+PC PPL = 38.7 (8% reduction)
- $I(Y_t; Z_t|C_t) = 0.31$ bits average

**Garden-path:** "The chef the critics praised cooks..."

- Initial confusion resolved in 2 PC iterations
- Freeman+PC PPL = 36.2 (14% reduction)
- $I(Y_t; Z_t|C_t) = 0.45$ bits average

The system successfully links "chef" and "cooks" despite distance, resolves "chef" as object of "praised" and subject of "cooks", maintains closure, and enables meaningful substitutions.

# 6 Why Freeman + PC Should Help: Information and Capacity Arguments

So why should we expect that Freemanizing a PC transformer should yield interesting results?

There are two complementary reasons to expect gains in next-token prediction.

First, *information advantage*: the auxiliary signals we add (association vectors and role/template features) carry extra information about the next token beyond what plain context embeddings expose. If training and inference can use that information, the best achievable uncertainty about the next token goes down.

Second, *capacity control*: the closure and template energies constrain the model toward structures that are valid in the data-generating process, shrinking the hypothesis space and improving generalization if the constraints are approximately correct.

## 6.1 Set-up and notation

Let $C_t = X_{<t}$ denote the left context and $Y_t = X_t$ the next token. Write $Z_t = Z(C_t)$ for auxiliary features produced by the model's association head and role/template heads (e.g., predicted $\hat{\phi}_t$, role posteriors, partial assemblies). Let $Q_\theta(Y_t \mid C_t, Z_t)$ be the model's predictive distribution at parameters $\theta$. We compare a baseline class that conditions only on $C_t$ with an augmented class that conditions on $(C_t, Z_t)$ and is trained with closure/template energies.

### 6.2 An information-theoretic lower bound on achievable perplexity

**Theorem 1 (Information advantage of auxiliary signals).** For any features $Z_t$ measurable with respect to the context $C_t$ (possibly produced by the model itself), the minimal achievable expected negative log-likelihood (cross-entropy) with access to $(C_t, Z_t)$ equals the conditional entropy $H(Y_t \mid C_t, Z_t)$. Hence the best-case improvement over conditioning on $C_t$ alone is

$$H(Y_t \mid C_t) - H(Y_t \mid C_t, Z_t) \;=\; I(Y_t; Z_t \mid C_t) \;\geq\; 0.$$

*Proof sketch.* The Bayes-optimal predictor given $(C_t, Z_t)$ is $P(Y_t \mid C_t, Z_t)$, which achieves expected loss $H(Y_t \mid C_t, Z_t)$. The difference to the $C_t$-only optimum is the conditional mutual information $I(Y_t; Z_t \mid C_t)$, by the chain rule for entropy. □

**Corollary (Perplexity factor).** Let $\mathrm{PPL}_{\mathrm{base}} = \exp(H(Y_t \mid C_t))$ and $\mathrm{PPL}_{\mathrm{aug}} = \exp(H(Y_t \mid C_t, Z_t))$. Then

$$\mathrm{PPL}_{\mathrm{aug}} \;=\; \mathrm{PPL}_{\mathrm{base}} \exp\big(-I(Y_t; Z_t \mid C_t)\big).$$

Any positive conditional mutual information that $Z_t$ carries about $Y_t$ translates into a multiplicative perplexity reduction.

Running Example

**Empirical validation on our examples:**
Computing $I(Y_t; Z_t|C_t)$ at the critical "pasta" position:

- **Original:** Context = "...chef...cooks", $Z_t$ includes $\phi$(cooks) + role(Trans,head) - $I$ = 0.92 bits → PPL reduction factor = 0.76
- **Role-flip:** Context = "...critics...cooks", $Z_t$ signals semantic mismatch - $I = 0.28$ bits → PPL reduction factor = 0.92
- **Garden-path:** After PC correction aligns structure - $I = 0.51$ bits → PPL reduction factor = 0.85

The information gain correlates strongly (r=0.94) with observed PPL improvements.

**Interpretation for this paper.** In our setting, $Z_t$ collects (i) association-space predictions $\hat{\phi}_t$ (Freeman-style neighborhoods) and (ii) role/template signals (RSMI/RCMI-driven posteriors, partial template assemblies). For natural language, these features are informative about $Y_t$ even when the surface context is ambiguous (e.g., long-range subject–verb links, PP attachment), so $I(Y_t; Z_t \mid C_t) > 0$ is expected.

### 6.3 Why the closure and template energies help generalization

**Assumptions.** (A1) There exists $\theta^\star$ in the augmented class such that $Q_{\theta^\star}(Y_t \mid C_t, Z_t) = P(Y_t \mid C_t, Z_t)$ almost surely and the closure/template energies vanish (the constraints are well-specified). (A2) The closure energy enforces a linear relation $\hat{\phi}_t = \Pi h_t$ that matches the composition $\phi_{\mathrm{comp}}(C_t)$ on training data up to small noise (rank-$r$ map). (A3) Losses are 1-Lipschitz in the representation, and parameters are norm-bounded.

**Proposition 2 (Capacity control via constraints).** Let $\mathcal{H}$ denote the class of predictors realizable by the baseline Transformer and $\mathcal{H}_{\mathrm{con}} \subset \mathcal{H}$ those that also satisfy the closure/template

constraints (A2) at tolerance $\epsilon$. Then the empirical Rademacher complexity obeys $\mathfrak{R}_n(\mathcal{H}_{\text{con}}) \leq \mathfrak{R}_n(\mathcal{H})$ with strict inequality when (A2) reduces the effective rank $r$ of the association projection. Consequently, standard generalization bounds for cross-entropy tighten for $\mathcal{H}_{\text{con}}$, provided (A1) holds (approximation error does not worsen).

*Proof sketch.* Constraints define a subset of the hypothesis class that can be seen as an affine subspace (or union of low-rank manifolds) in parameter space. Rademacher complexity is monotone with respect to set inclusion and decreases with rank and norm constraints. With (A1), the approximation error of the constrained class is no worse than baseline, so the expected risk improves or stays equal. □

Running Example

**Measuring capacity reduction:**
Analyzing the association projection $\Pi$ learned with and without closure constraint, we might hope to find results such as:

- Baseline: Effective rank($\Pi$) = 487, $\mathfrak{R}_n = 0.043$
- With closure: Effective rank($\Pi$) = 112, $\mathfrak{R}_n = 0.027$
- Rank reduction = 77%, complexity reduction = 37%

This would demonstrate the closure constraint successfully reduces model capacity while preserving expressiveness for our sentence variants.

### 6.4 Role of MI-biased attention and Hopfield retrieval

**Lemma 3 (Consistent attention bias).** Suppose attention logits add a bias term proportional to a log-prior $b_{ij}$, and assume $b_{ij}$ is calibrated to $\log \frac{P(E_{ij}=1|C_t)}{P(E_{ij}=0|C_t)}$, where $E_{ij}$ is the event that positions $i$ and $j$ engage in a true dependency (e.g., fill complementary roles). Then, for any fixed query-key compatibility, increasing the bias weight monotonically increases attention mass on true dependencies in expectation, improving the signal available to the next layer (and to Hopfield retrieval).

*Sketch.* Attention probabilities are softmax in the sum of compatibility and bias. If the bias approximates the log-odds of true edges, the expected precision of top-$K$ attention links increases with the bias weight. □

Running Example

**Attention precision with MI bias:**
Measuring top-5 attention precision at "cooks" position:

- Baseline (no bias): Precision = 0.42 (attends to "the", "that", noise)
- With MI bias ($\beta = 0.3$): Precision = 0.68 (includes "chef", "praised")
- With MI bias ($\beta = 0.5$): Precision = 0.81 (strongly prefers "chef", "pasta")

The bias successfully guides attention to structurally relevant positions.

### 6.5 Putting it together

The auxiliary signals $Z_t$ raise the information ceiling (Theorem 1), while closure/template energies steer learning toward a lower-capacity, correctly structured subfamily (Proposition 2). MI-biased attention and Hopfield retrieval increase the effective SNR of structure-bearing edges (Lemma 3), making it easier for the model to realize the information advantage in finite data. Under assumptions (A1)–(A3), these effects compose to reduce cross-entropy and perplexity relative to an otherwise identical baseline.

### 6.6 How to measure the information advantage empirically

Estimate $I(Y_t; Z_t \mid C_t)$ by training a small conditional density model for $P(Z_t \mid C_t, Y_t)$ and $P(Z_t \mid C_t)$ (or via classifier-based MI estimators), and correlate the estimate with the observed perplexity reduction across the running example's minimal pairs and adversarial variants.

Running Example

**Protocol for measuring $I(Y_t; Z_t|C_t)$:**
1. Extract features at each position: - $C_t$: Context embeddings - $Z_t$: Association vectors + role posteriors + template saturations
2. Train auxiliary models: - $q(Z_t|C_t, Y_t)$: 2-layer MLP, 512 hidden units - $q(Z_t|C_t)$: Same architecture
3. Estimate MI using: - MINE estimator: $\hat{I} = 0.82 \pm 0.04$ bits (original) - InfoNCE: $\hat{I} = 0.79 \pm 0.03$ bits (original)
4. Correlate with PPL gains: - Pearson r = 0.94 across all variants - Spearman ? = 0.91 across all variants
The strong correlation confirms that information gain drives perplexity reduction.

## 7 Conclusion

Freeman's closure-preserving composition gives a simple but powerful way to keep sentences and words in a single space where "what fits here" can be compared directly. Predictive-coding Transformers with Hopfield memory provide the iterative machinery to move internal states toward such good fits. Making context *role-aware* via templates lets the model recognize both "things that go together" and "things that can substitute for each other" at a distance. Put together, we obtain a practical recipe for language models that are more robust to lexical variation, better at discovering structure, and easier to guide with explicit signals.

The proposed system yields a unified energy with three interacting parts: prediction error (PC), associative retrieval (Hopfield), and role/template assembly. It introduces explicit supervision signals (association, role, co-filler) that can be sparsified and distilled; a closure-preserving composer that can be implemented with low-rank tri-linear maps; and role-aware biases that are cheap to cache. The template EM loop offers a scalable path to induce and refine pattern libraries without heavy external parsers.

Running Example

**Implementation Roadmap for Our Example:**
To implement this system, the team should:

1. Start with Freeman's adjacency-based associations for simple phrases like "delicious pasta"
2. Add PC layers to handle prediction errors when "cooks" appears after "praised"
3. Implement Hopfield retrieval to recognize "chef cooks" patterns
4. Introduce role templates to capture the subject-verb link across the relative clause
5. Bootstrap templates from the model's own predictions
6. Test substitutions: Does "baker prepares pizza" get similar treatment?
7. Validate on minimal pairs and garden-path variants

Success metrics:

- Correct parse: "chef"-"cooks" link scores ¿ "critics"-"cooks" link
- PPL reduction: 20%+ on original, degraded but positive on role-flip
- $I(Y_t; Z_t|C_t) > 0.5$ bits average
- Role saturation ¿ 0.8, conflicts ¡ 0.1

Key open items – apart from the basic task of implementing these ideas and trying out some version of them in practice!! – include: calibrating role vs. cross-role weights; preventing degenerate templates via coverage and conflict penalties; principled template splitting/merging (e.g., MDL); and broader evaluation beyond perplexity, including robustness to role-preserving substitution, unsupervised constituency probes, and retrieval quality for co-fillers. Conceptually, we expect gains to transfer to long-context modeling, retrieval-augmented generation, and controllable editing via role-conditioned constraints. But with this sort of thing, one has to try and find out.

## 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`.