OSN3P: An Optimized Mixture-of-Experts Routing Framework

Kerode Mume

Abstract

OSN3P is a sparse mixture-of-experts routing framework built around a deterministic tokenization scheme and a one-shot graph-routing core. Each character is encoded by three attributes: its position in the alphabet, its position inside the word, and a binary capitalization flag. These lexical tokens are pooled into word representations, assembled into a 12-word semantic search window, and then routed through a sparse expert lattice based on Neuron Contact, Next-Neuron, and Next-Neighborhood One-shot Pathfinding networks. The result is a controllable and interpretable architecture that emphasizes explicit routing, low-overhead preprocessing, and modular domain specialization. This paper formalizes the tokenization rule, the routing mechanism, and the N^3P expert computation underlying OSN3P.

Keywords. mixture-of-experts, sparse routing, deterministic tokenization, graph experts, one-shot pathfinding

1 Introduction

Large neural systems often trade interpretability for scale. OSN3P is motivated by a different design goal: preserve routing transparency while keeping the representation pipeline lightweight and easy to inspect. Instead of relying on opaque subword segmentation alone, OSN3P begins with a deterministic lexical decomposition of every character. Those character tuples are converted into compact word embeddings, grouped into fixed-width semantic windows, and then processed by a sparse mixture-of-experts (MoE) router.

The core computational block is a family of N^3P experts, where each expert reasons over three local graph relations: neuron contact, next-neuron adjacency, and next-neighborhood context. The router activates only the most relevant experts for a given semantic window, which reduces unnecessary computation and yields a route trace that can be inspected after inference.

This manuscript presents OSN3P as a system paper and reference formulation. The main contributions are:

- a deterministic tokenization rule based on alphabet index, intra-word position, and capitalization;
- a 12-word semantic window representation for keyword extraction and routing;
- a sparse MoE controller over domain-specialized N^3P experts for interpretable one-shot inference.

2 Framework Overview

Figure 1 summarizes the proposed pipeline. Input text is decomposed at the character level, lifted into semantic nodes, pooled into a 12-word search window, and finally routed to a sparse subset of

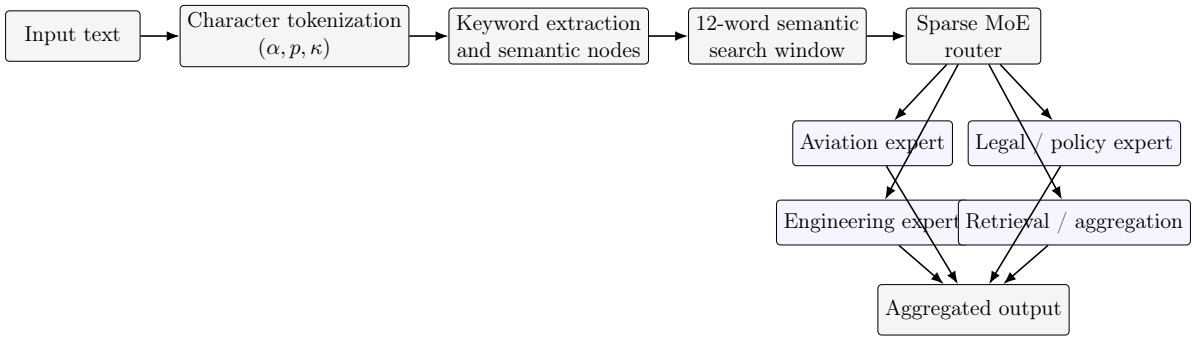

Figure 1: High-level view of OSN3P. A deterministic tokenizer feeds a semantic window encoder, and a sparse MoE router activates a small set of N^{3}P experts specialized by domain.

Word	Character tuples (α, p, κ)
`Drone`	$(4,1,1),(18,2,0),(15,3,0),(14,4,0),(5,5,0)$
`FAA`	$(6,1,1),(1,2,1),(1,3,1)$

Table 1: Examples of the deterministic tokenization rule used by OSN3P.

expert modules. The resulting architecture supports domain-specialized processing without forcing every input through every expert.

3 Deterministic Tokenization

For a word $w = (c_1, \ldots, c_m)$, OSN3P assigns each character a three-dimensional token

$$\mathbf{x}_i = [\alpha(c_i),\ i,\ \kappa(c_i)] \in \mathbb{R}^3, \tag{1}$$

where $\alpha(c_i) \in \{0, 1, \ldots, 26\}$ is the character's place in the alphabet, i is the character position within the word, and $\kappa(c_i) \in \{0, 1\}$ marks whether the character is capitalized. Non-alphabetic characters may be assigned $\alpha(c_i) = 0$.

The resulting word representation is obtained by pooling over the character tuples:

$$\phi(w) = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i W_x, \qquad W_x \in \mathbb{R}^{3 \times d}. \tag{2}$$

This formulation preserves deterministic lexical structure while still allowing a learned projection into a d-dimensional working space.

Table 1 gives two concrete examples. The word `Drone` begins with an uppercase `D`, so its capitalization bit is active only for the first character, while the acronym `FAA` retains capitalization across the entire token sequence.

4 Semantic Windows and Keyword Extraction

After each word has been embedded, OSN3P forms a local semantic search window of length 12:

$$S_t = (w_t, w_{t+1}, \ldots, w_{t+11}), \qquad \mathbf{h}_t = \frac{1}{12} \sum_{j=0}^{11} \phi(w_{t+j}). \tag{3}$$

This fixed-width window is intended to capture immediate context without requiring full-sequence dense attention.

Keyword extraction can then operate on both the pooled representation $\mathbf{h}_t$ and the lexical evidence retained by the underlying token tuples. In practice, salient words are promoted into semantic nodes that serve as anchors for routing. A representative 12-word example, aligned with the motivating use case of this paper, is the query:

> Why are drone flyovers over homes in suburbs regulated by the FAA?

This window highlights semantic anchors such as `drone`, `flyovers`, `homes`, `suburbs`, `regulated`, and `FAA`. These anchors are then used to assemble the graph context passed to the expert router.

5 Sparse Mixture-of-Experts Routing over N3P Experts

5.1 Top-r routing

Let K denote the number of available experts. The routing network produces logits

$$\mathbf{z}_t = W_g \mathbf{h}_t + \mathbf{b}_g, \tag{4}$$

and converts them into a sparse gate by retaining only the top-r entries:

$$g_k(\mathbf{h}_t) = \begin{cases} \dfrac{\exp(z_{t,k})}{\sum_{j \in \mathcal{T}_r(\mathbf{z}_t)} \exp(z_{t,j})}, & k \in \mathcal{T}_r(\mathbf{z}_t), \\ 0, & \text{otherwise}, \end{cases} \tag{5}$$

where $\mathcal{T}_r(\mathbf{z}_t)$ denotes the indices of the r largest routing logits. This preserves the flexibility of an MoE model while ensuring sparse execution.

5.2 N3P expert computation

Each expert is defined over three local relation types: neuron-contact edges, next-neuron edges, and next-neighborhood edges. For expert k, let $\hat{A}_k^{(c)}$, $\hat{A}_k^{(n)}$, and $\hat{A}_k^{(b)}$ denote normalized adjacency matrices for these three relations, and let X_t be the node-feature matrix derived from the current semantic window and expert memory. A one-shot expert update is written as

$$H_k = \sigma\Big(\hat{A}_k^{(c)} X_t W_k^{(c)} + \hat{A}_k^{(n)} X_t W_k^{(n)} + \hat{A}_k^{(b)} X_t W_k^{(b)}\Big), \tag{6}$$

where $\sigma(\cdot)$ is a pointwise nonlinearity. The expert output is then

$$f_k(X_t) = \mathrm{Pool}(H_k) W_k^{(o)}. \tag{7}$$

Because the expert performs one graph propagation step and returns a pooled route summary, inference remains lightweight and directly attributable to a small set of active paths.

5.3 Expert aggregation

The routed prediction for the semantic window is the weighted sum

$$\hat{\mathbf{y}}_t = \sum_{k=1}^{K} g_k(\mathbf{h}_t)\, f_k(X_t). \tag{8}$$

This formulation makes the route trace explicit: every output can be decomposed into its active experts, their gates, and the semantic nodes that triggered them.

Expert family	Primary responsibility
Aviation	Flight terminology, airspace concepts, FAA-specific reasoning
Legal / policy	Regulation, compliance, and rule-based interpretation
Engineering	Causal reasoning, feasibility, and technical implications
HOA / civic	Neighborhood governance and local property constraints
Document retrieval	Grounded lookup over stored references or indexed notes
Statistics aggregation	Consensus building, ranking, and route summarization

Table 2: Representative domain-specialized experts within OSN3P.

6 Optimization Objective and Computational Profile

For a supervised target $\mathbf{y}_t$, OSN3P can be trained with a task loss plus a load-balancing regularizer:

$$\mathcal{L} = \mathcal{L}_{\text{task}}(\hat{\mathbf{y}}_t, \mathbf{y}_t) + \lambda K \sum_{k=1}^{K} \left(\bar{g}_k - \frac{1}{K} \right)^2, \tag{9}$$

where $\bar{g}_k$ is the average routing mass assigned to expert k over a minibatch. The regularizer discourages router collapse while still allowing sharp top-r decisions on individual windows.

The computational advantage of OSN3P comes from sparse activation. For each semantic window, routing costs $\mathcal{O}(Kd)$ to score experts, but only $r \ll K$ experts are executed. If the active expert graph has edge set E_a, the expert-side cost scales as $\mathcal{O}(r|E_a|d)$ rather than $\mathcal{O}(K|E_a|d)$. This is the operational sense in which OSN3P is optimized: most experts remain idle for most windows, and each selected expert performs only a one-shot graph update.

7 Interpretable Routing Example

The query

> Why are drone flyovers over homes in suburbs regulated by the FAA?

illustrates the intended behavior of the model. The tokenizer exposes both lexical identity and local structure, preserving capitalization in `FAA` and word position across the full 12-word window. Keyword extraction promotes the main semantic anchors, after which the router can prioritize expert families that align with the observed context.

Table 2 lists a representative expert layout derived from the proposed framework.

In this example, OSN3P would naturally bias toward aviation and legal experts, while retrieval provides grounding and engineering or civic experts can be recruited if the downstream task requires additional causal or neighborhood-level context. The resulting route is sparse, inspectable, and easy to debug: one can review the extracted keywords, the active experts, and the final aggregation weights without unpacking an opaque dense stack.

8 Discussion and Conclusion

OSN3P is designed as an interpretable routing framework rather than a replacement for every dense language model. Its deterministic tokenizer intentionally favors transparency over maximal expressiveness, and its performance will depend on the quality and specialization of the available

experts. That trade-off is a design choice: the framework emphasizes explicit routing logic, modular expert growth, and controllable domain decomposition.

As an arXiv-ready system description, this paper provides the reference formulation for the model: a lexical tokenizer based on alphabet position, word position, and capitalization; a 12-word semantic search stage; and a sparse MoE controller over N^3P experts. Future work can build on this foundation by benchmarking the model, expanding expert inventories, and integrating retrieval-backed supervision for grounded responses.