Indian Journal of Computer Science and Technology
https://www.doi.org/10.59256/indjcst.20260501047
Volume 5, Issue1 (January-April 2026), PP: 344-349.
www.indjcst.com
ISSN No: 2583-5300

Dhatu-Former: Redesigning Transformer Architectures Through Pan.ini's As..tadhyay

Nagesh Jayanti
Founder, Conscious Bridge Labs.

To Cite this Article: *Nagesh Jayanti, "Dhatu-Former: Redesigning Transformer Architectures Through Pan.ini's As..tadhyay ", Indian Journal of Computer Science and Technology, Volume 05, Issue 01 (January-April 2026), PP: 344-349.*

Abstract: *Contemporary large language models (LLMs) rely on sub-word tokenizers and at attention mechanisms that treat every language as a statistical surface-form distribution. This paper proposes Dhatu-Former, a transformer architecture that internalizes the formal linguistic machinery of Pan.ini's As..tadhyay the oldest known generative grammar. We hypothesize that (i) morphologically-aware, root-based (dhatu-based) tokenization can reduce vocabulary size and sequence length by 40 60%, (ii) hierarchical attention guided by Pan.inian derivation trees can yield sparse, interpretable attention with O(nlogn) complexity, and (iii) a hybrid symbolic neural reasoning layer that executes sutra-style rewrite rules can substantially reduce hallucination while enabling uni ed language math logic reasoning. We further introduce a modular Retrieval-Augmented Generation (RAG) subsystem grounded in Sanskrit lexical databases (Amarakos.a, Dhatupat.ha) and a continual learning framework inspired by the paribhas.a sutra (meta-rules) of the As..tadhyay . We present order-of-magnitude parameter reduction estimates, architectural blueprints with TikZ diagrams, and a research roadmap for empirical validation. This is a position paper; no experiments have been conducted.*

I. INTRODUCTION

Modern transformer-based language models such as GPT-4, Claude, and Gemini are trained on trillions of tokens drawn overwhelmingly from English and other analytic languages. Their subword tokenizers (BPE, SentencePiece) fragment text into statistically frequent byte sequences with no awareness of morphological structure [7]. This design choice has profound downstream consequences: bloated vocabularies (32k 256k tokens), long sequence lengths, quadratic attention costs, and most critically a reasoning substrate that is purely distributional, lacking the compositional guarantees that a formal grammar provides. Pan.ini's As..tadhyay [1] (c. 4th century BCE) is a system of ∼4,000 sutras (rules) that generates the entirety of Sanskrit morphology and signi cant portions of its syntax from ∼2,000 verbal roots (dhatus). The system is:

Generative: every valid surface form is derived via a deterministic rule chain from a root plus a xes.

Hierarchical: rules are organized by domain (phonology, morphology, syntax) with explicit meta-rules (paribhas.a sutras) governing rule interaction.

Compact: the grammar is renowned for extreme brevity; Pan.ini famously preferred saving half a short vowel over verbosity.

These properties align remarkably well with desirable attributes of neural architectures: compositionality, sparsity, and parameter e ciency. Recent work has validated component-level aspects of this idea: morphology-aware tokenizers [4, 5], Sanskritspeci c neural tokenization [11], and quantitative studies showing that Sanskrit can require 50% fewer tokens than English under unbiased tokenization [3]. Nevertheless, no prior work has proposed a complete transformer architecture redesign that internalizes Pan.inian principles as core architectural components for general-purpose language modeling. This paper explores how the structural principles of the As..tadhyay can be transplanted into transformer design to yield a new class of models we call Dhatu-Former.

II.BACKGROUND

2.1 Pan.ini's Grammar: A Computational Perspective

The As..tadhyay [1, 2] operates on a pipeline that is strikingly analogous to a modern compiler:

1. Lexicon lookup: Select a dhatu (root) from the Dhatupat.ha and a desired meaning (artha).
2. A x selection: Apply pratyahara (su x groups) based on tense, voice, person, number.
3. Sutra chain: Execute a deterministic sequence of phonological transformations (sandhi), augmentation (agama), and deletion (lopa).
4. Surface form: Produce the nal in ected word.

Each step is governed by context-sensitive rewrite rules of the form:

$$A \xrightarrow{\text{sutra } i.j.k} B \,/\, C\_D \qquad (1)$$

where C and D de ne the left and right contexts, respectively. This is formally equivalent to a context-sensitive grammar

(Type-1 in the Chomsky hierarchy), strictly more powerful than the contextfree grammars typically assumed in NLP.

2.2 Limitations of Current Tokenization

BPE-style tokenizers treat text as a byte stream [7]. For a morphologically rich language like Sanskrit (or Hindi, Tamil, Finnish, Turkish), a single semantic unit may be split into 5 12 sub-word tokens.

Consider: This fragmentation wastes compute: attention is $O(n^2)$ in sequence length, and each additional token consumes KV-cache memory at inference time.

Word	BPE tokens	Dhatu tokens
pratisamskaran.am	6 8	1 (root + a xes)
vyavasthapayati	5 7	1 (root + a xes)
nirvis.ayatva	4 6	1 (root + a xes)

2.3Token E ciency: Sanskrit vs. English in LLMs

Recent quantitative studies [3] con rm that current LLMs consume signi cantly more tokens for Sanskrit text than English, despite Sanskrit's greater information density per morpheme. Under unbiased SentencePiece tokenization, Sanskrit requires approximately 50% fewer tokens than English; however, legacy tokenizers like cl100k_base invert this relationship, in ating Sanskrit's token count by nearly 2.5×. This discrepancy functions as a hidden token tax on Indic language users. The ArthaLabs Pan.ini Tokenizer [5] demonstrates 2 4× token reduction using recursive morphological parsing, validating the tokenizer component of our architecture independently.

III. ARCHITECTURAL DESIGN: DHATU-FORMER

We propose the Dhatu-Former as a modular transformer architecture with four novel subsystems layered atop a standard decoder backbone. Figure 1 provides the high-level blueprint.

3.1 Component 1: Dhatu-Aware Tokenizer

The tokenizer performs morphological analysis before embedding, replacing BPE with a three-stage pipeline:

1. Sandhi splitting: Reverse sandhi (euphonic combination) to recover word boundaries, following deterministic algorithms [11].
2. Morphological decomposition: Parse each word into root (dhatu or pratipadika), pre xes (upasarga), and su xes (pratyaya).
3. Hierarchical encoding: Emit a structured token tuple (root_id, affix_vector, sandhi_context) rather than a at integer.

For non-Sanskrit input, a learned morphological analyzer [4] produces analogous decompositions.

Vocabulary reduction. Sanskrit has ∼2,000 roots and ∼500 productive a xes. A dhatu-aware vocabulary can base units versus 32k 256k for BPE. Even for multilingual models, we estimate a combined morphological vocabulary of 15k 20k units

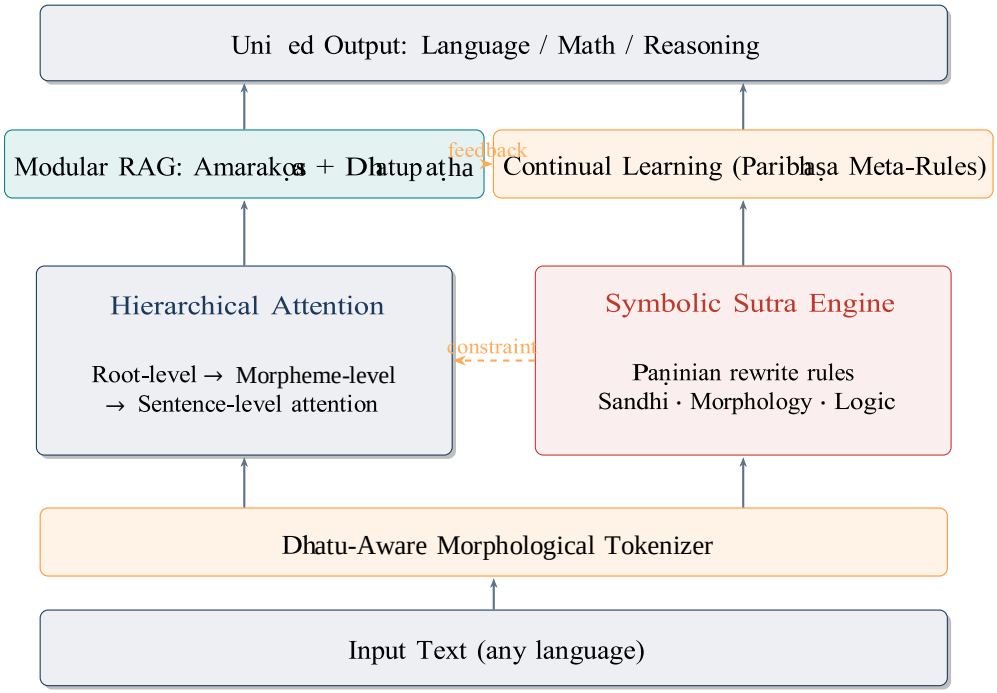


Figure 1: High-level architecture of Dhatu-Former. Sa ron blocks denote Sanskrit-speci c modules; blue blocks denote standard transformer components; red blocks denote symbolic reasoning; teal blocks therefore be bounded at ∼4,000 denote retrieval..

3.2 Component 2: Hierarchical Attention Standard multi-head attention [7] computes:

$$\text{Attn}(Q,K,V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \quad (2)$$

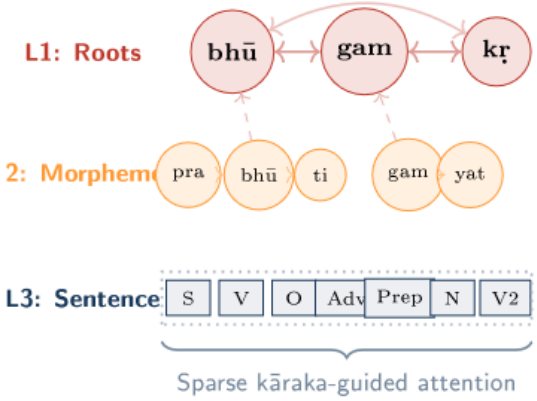
over the full sequence, yielding $O(n^2)$ complexity. We replace this with a three-level hierarchical scheme inspired by Pan.ini's rule domains:

Level 1Root attention ($O(r^2)$ where $r \ll n$): Attends only over root tokens.

Level 2 Morpheme attention ($O(m)$ per word, local): Within each word, a xes attend to their root.

Level 3 Sentence attention ($O(n\log n)$ via sparse patterns): Full-sequence attention with a sparsity mask derived from karaka (case-role) structure.

The combined complexity is $O(r^2+wm+n\log n)$, dominated by $n\log n$a substantial improvement over.



3.3 Component 3: Symbolic Sutra Engine

A critical innovation of Dhatu-Former is a dieren- tiable symbolic reasoning module [9, 10] that exe- cutes Pan. inian rules as constrained neural operations.

Architecture. The engine maintains a rule bank R = {r1, . . . , rK } where each ri is a learned context-sensitive rewrite rule parameterized as:ri : (hleft, hfocus, hright) 7 → h′focus (3)

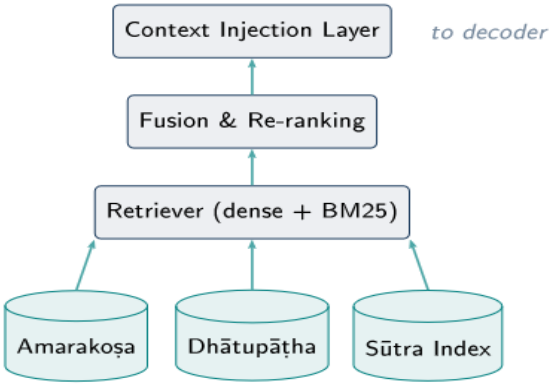


Figure 3: Modular RAG with three Sanskrit knowledge bases.

where **h** are hidden state vectors. Rules are applied via a gating mechanism $g_i \in [0,1]$ analogous to the adhikara system in the As..tadhyay.

Hallucination reduction. The engine enforces:

- Morphological validity: Generated tokens must be derivable via a valid sutra chain.
- Logical consistency: For reasoning tasks, each inference step corresponds to a valid rule application.

3.4 Component 4: Modular RAGwith Sanskrit Knowledge Bases

The RAG subsystem indexes three types of knowledge, building on existing Sanskrit NLP infrastructure [6, 2]:

1. **Amarakos.a (thesaurus**): Semantic eld retrieval.
2. **Dhatupat.ha (root catalogue):** Root meanings and derivation patterns.
3. **Sutra index:** Context-sensitive retrieval of applicable rules given a derivation state.

IV.PARAMETER AND COMPUTE REDUCTION ESTIMATES

We hypothesize parameter reductions across three axes (Table 1, Figure 4), following e ciency analyses similar to [8].

Embedding layer. A 256k-token BPE vocabulary at dimension $d = 8192$ requires ∼2.1B parameters for the embedding matrix alone. A morpho- logical vocabulary of 20k units requires ∼164M a 12.8× reduction.

Component	Reduction	Mechanism
Embedding layer	8- 16x	Smaller vocab
Attention (KV cache)	3 -5x	Shorter sequences
Attention (compute)	5 -10x	Hierarchical sparse
FFN / MoE layers	2- 4x	Modular experts
Total parameters	4- 10x	Combined

Table 1: Estimated reductions vs. a 70B-parameter baseline.

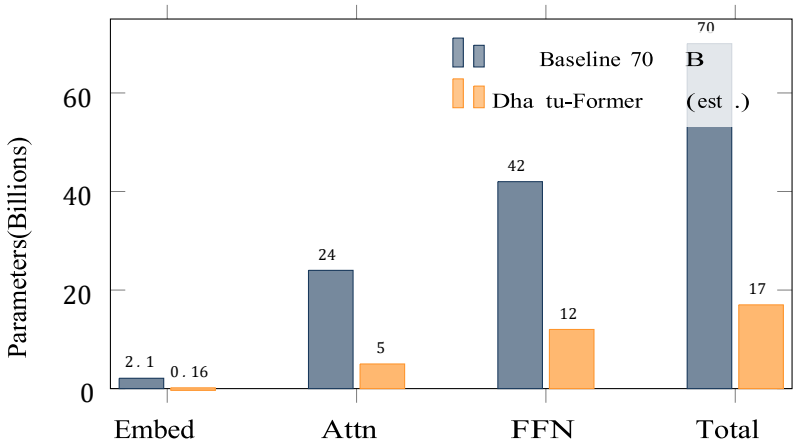


Figure 4: Estimated parameter distribution.

Attention layers. Hierarchical attention with root-level sparsity reduces e ective sequence length from n to $\sim n/4$ at Level 1, yielding ∼16× FLOPs reduction for that level. Averaged across three levels, we estimate 5 10× savings.

Feedforward layers. A Mixture-of-Experts design where expert modules correspond to linguistic domains can reduce active parameters per token by 2 4×.

Net estimate. For a model targeting Llama3 70B quality, we estimate Dhatu-Former could achieve comparable performance at 7 17B parameters, a reduction of 4 10×.

V.GPU AND COMPUTE IMPLICATIONS

Training. A 70B model requires ∼2048 H100 GPUs for ∼90 days. A 17B Dhatu-Former would require ∼512 H100 GPUs for ∼45 days. At 7B, training could be accomplished on ∼128 GPUs in ∼30 days.

Inference. With 3 5× shorter e ective sequences, KV-cache requirements drop proportionally, enabling larger batches,longer contexts, and consumer-grade deployment.

Energy. Training compute scales as $6 \times N \times D$ FLOPs. A 4× reduction in N and 2× reduction in D yields ∼8× energy savings.

VI.HALLUCINATION REDUCTION

Hallucination arises from: (i) distributional uncertainty, (ii) lack of grounding, and (iii) absence of logical consistency checks [9]. Dhatu-Former addresses all three:

6.1 Symbolic Constraints (Source i)

The sutra engine constrains the output distribution at each decoding step:

$$P_{\text{constrained}}(w_t|w_{<t}) = \frac{P_\theta(w_t|w_{<t}) \cdot \mathbf{1}[w_t \in \mathcal{V}_{\text{valid}}(w_{<t})]}{\sum_{w' \in \mathcal{V}_{\text{valid}}} P_\theta(w'|w_{<t})} \quad (4)$$

6.2 Retrieval Grounding (Source ii)

Every factual claim triggers a retrieval query against indexed knowledge bases [6].

6.3 Logical Consistency Checking (Source iii)

For reasoning chains, the sutra engine operates as a proof checker; steps that fail validation are rejected and re-generated.

VII.UNI ED LANGUAGE MATH REASONING

The Pan.inian rule formalism is inherently mathematical. This suggests a natural uni cation:

Language: Morphological/syntactic generation via sutra chains.
Mathematics: Symbolic manipulation using the same rewrite engine, with operators as roots and operands as arguments.
Reasoning: Logical inference via rule chaining.

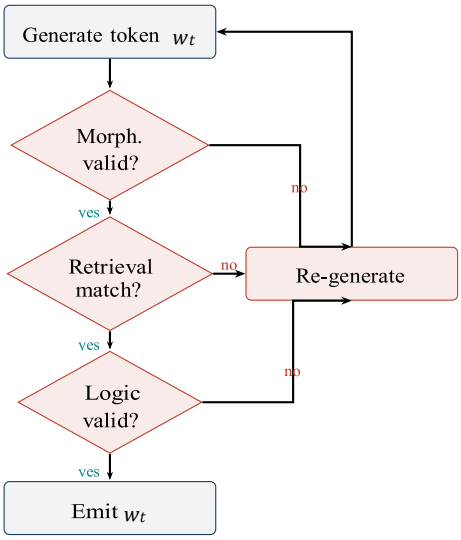


Figure 5: Three-gate hallucination reduction pipeline

VIII.CONTINUAL AND REAL TIME LEARNING

The As..tadhyay's meta-rule system This uni cation eliminates the need for separate tool-use modules for code execution or symbolic math.sutras) provides inspiration for continual learning:

Meta-rule priority. When a new sutra con icts with an existing one, Pan.ini de nes explicit priority ordering. Dhatu-Former maintains a priority queue of learned rules.

Modular updates. Because the architecture separates linguistic knowledge (sutra engine), world knowledge (RAG indices), and reasoning patterns (attention weights), each can be updated independently:

RAG index: Updated in real time.
Sutra bank: Fine-tuned via LoRA.
Attention core: Updated via gradient-based training.

IX.CHALLENGES AND LIMITATIONS

Sanskrit-speci c bias. The hierarchical attention mechanism must be validated across typologically diverse languages; analytic languages bene t less than agglutinative ones.

Differentiability of symbolic rules. Hard constraints can create gradient discontinuities. Soft relaxations (Gumbel-softmax) are needed during training.

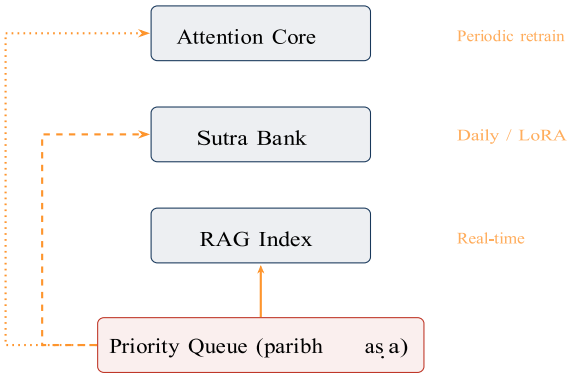


Figure 6: Continual learning with modular update frequencies.

Computational overhead. With $K \approx 4000$ rules and context of ~8192 tokens, naive rule matching is O(Kn) per step. E cient indexing is essential.

Data requirements. Morphologically analyzed corpora are scarce outside major languages.

Evaluation. Standard benchmarks may not capture the advantages of morphological generalization and reasoning faithfulness.

X.RELATED WORK

Morphological tokenization. Work such as MorphBPE [4] and the ArthaLabs Pan.ini Tokenizer [5] demonstrate 2 4× token reduction, validating component-level feasibility. Our architecture extends morphological awareness beyond tokenization to attention and reasoning.

Sparse and hierarchical attention. Longformer, BigBird, and e cient attention variants [7, 8] reduce complexity but use xed or random sparsity. Dhatu-Former's sparsity is linguistically structured.

Neuro-symbolic reasoning. [9] Prior work integrates symbolic reasoning with neural networks, but Dhatu-Former is unique in using a natural language grammar as its symbolic substrate.

Sanskrit NLP. Foundations include the Sanskrit Heritage Engine [2], SanskritShala [6], and TransLIST [11]. The closest related proposal is the Panini Sutra-based AI model concept [10], which identi es the need for Panini-based architectures but does not specify one. Our work extends these from analysis to generation and from NLP tools to architectural principles.

XI.FUTURE WORK AND RESEARCH ROADMAP

1. **Phase 1 (Months 1 6):** Build the dhatuaware tokenizer and benchmark against BPE on Sanskrit, Hindi, and English.
2. **Phase 2 (Months 6 12):** Implement hierarchical attention in a 1B-parameter model; benchmark on FLORES, TyDi QA, and morphological generalization tasks.
3. **Phase 3 (Months 12 18):** Integrate the symbolic sutra engine and RAG subsystem; benchmark on GSM8K, MATH, LogiQA, TruthfulQA, and FActScore.
4. **Phase 4 (Months 18 24):** Scale to 7 17B parameters and conduct head-to-head comparisons with Llama-3, Mistral, and Gemma.
5. **Phase 5 (Months 24 36):** Evaluate continual learning and cross-lingual transfer across 50+ languages.

XII.CONCLUSION

Pan.ini's As..tadhyay is a rigorously formal system whose design principles of compositionality, hierarchical organization, and extreme compression are directly applicable to modern neural architecture design. The Dhatu-Former architecture proposed here represents a rst attempt to operationalize these principles, with potential for 4 10× parameter reduction, substantially reduced hallucination, and a uni ed framework for language, mathematics, and reasoning.

The approach faces signi cant engineering challenges particularly in making symbolic constraints di erentiable and in building high-quality morphological analyzers across languages but the theoretical foundations are sound and the potential e ciency gains are too signi cant to ignore. We believe this line of research can contribute to a paradigm shift from brute-force scaling toward linguistically informed, architecturally e cient language models.

This is a position paper. Parameter estimates are hypothetical and subject to empirical validation. No experiments have been conducted; all gures represent architectural proposals and projected outcomes

References

1. P. Kiparsky. On the architecture of Pan.ini's grammar. In G. Huet, A. Kulkarni, and P. Scharf, editors, Sanskrit Computational Linguistics, volume 5402 of Lecture Notes in Computer Science, pages 33 94. Springer, 2009.
2. G. Huet. Sanskrit heritage engine: A computational platform for Sanskrit processing. Technical report, INRIA, 2009.
3. Anonymous. Is Sanskrit the most tokene cient language? A quantitative study using GPT, Gemini, and SentencePiece. arXiv preprint arXiv:2601.06142, 2026.
4. A. Lundin et al. MorphBPE: A morphologyaware tokenizer bridging linguistic complexity for e cient LLM training across morphologies. arXiv preprint, 2025.
5. ArthaLabs. Panini tokenizer: Grammarrst Sanskrit tokenization. Hugging Face, 2025. https://huggingface.co/ArthaLabs/ panini-tokenizer.
6. J. Sandhan et al. SanskritShala: A neural Sanskrit NLP toolkit with web-based interface for pedagogical and annotation purposes. In Proceedings of ACL 2023 System Demonstrations, 2023.
7. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, . Kaiser, and I. Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems 30 (NeurIPS 2017), 2017.
8. H. Wang et al. TokenFormer: Rethinking transformer scaling with tokenized model parameters. arXiv preprint arXiv:2410.23168, 2024.
9. L. C. Lamb, A. d'Avila Garcez, M. Gori, M. Prates, P. Avelar, and M. Vardi. Graph neural networks meet neural-symbolic computing: A survey and perspective. arXiv preprint arXiv:2003.00330, 2020.
10. R. Singh et al. Sanskrit LLM: Proving the need for a Panini sutra rule-based AI model by benchmarking the capabilities of existing LLMs. In Springer, 2025.
11. J. Sandhan, R. Singha, N. Rao, S. Samanta, L. Behera, and P. Goyal. TransLIST: A transformer-based linguistically informed Sanskrit tokenizer. In Findings of EMNLP 2022, pages 6902 6912, 2022.