A Survey on Automated Metrical Analysis of Sanskrit Prosodic Meters and NLP

*J. Seetharaman[1] and U. Vageeswari[2]

[1]Asst Professor, Department of Computer Science and Applications, SCSVMV (DU), Enathur, Kanchipuram, Tamilnadu, INDIA, jseetharaman@kanchiuniv.ac.in

[2]Asst Professor, Department of Computer Science and Applications, SCSVMV (DU), Enathur, Kanchipuram, Tamilnadu, INDIA, uvageeswari@kanchiuniv.ac.in

Abstract

Sanskrit prosody (Chandas) represents one of the most rigorous metrical systems in world literature, combining phonology, morphology, and semantics into strictly governed poetic forms. Automated metrical analysis of Sanskrit text requires precise sandhi resolution, syllable and mora identification, and grammatical validation. This paper presents a structured survey of Vedic and Classical Sanskrit meters, a staged evaluation of existing Sanskrit NLP tools, and practical implementations using transformer-based, rule-based, and grammar-driven sandhi segmentation systems. In addition, outputs from the Sanskrit Heritage Segmenter (INRIA) are analyzed to demonstrate large-scale grammatical ambiguity inherent in Sanskrit parsing. The study concludes that hybrid pipelines integrating neural ranking with Paninian grammar engines are essential for accurate and scalable computational chandas analysis.

Keywords—Sanskrit Chandas, Prosody, Sandhi Splitting, Sanskrit NLP, Computational Linguistics, Indic Languages

1. Introduction

Sanskrit literature exhibits a unique synthesis of linguistic precision and metrical exactness. From the earliest Vedic hymns composed in Gāyatrī(गायत्री),Triṣṭubh(त्रिष्टुप्), and Jagatī (जगती) meters to elaborate Classical meters such as Śārdūlavikrīḍita (शार्दूलविक्रीडित) and Mandākrāntā (मन्दाक्रान्ता), prosody governs both semantic clarity and aesthetic rhythm.

Computational processing of Sanskrit is fundamentally challenging due to productive compounding, extensive sandhi, free word order, and meter-sensitive phonetic length. Unlike many modern languages, accurate syllabification in Sanskrit is inseparable from grammatical interpretation. This paper bridges classical Sanskrit prosody with modern NLP pipelines and demonstrates how different computational paradigms handle metrical prerequisites.

R. Vasanth Kumar Mehta et al. (eds.), *Proceedings of the International Conference on Intelligent Systems for a Sustainable Future (ISSF 2026)*, Atlantis Highlights in Intelligent Systems 16,
https://doi.org/10.2991/978-94-6239-693-7_83

2. Overview of Sanskrit Prosodic Meters

2.1 Vedic Meters

Vedic meters as few listed in Table1 are syllable-based and foundational to Sanskrit prosody.

Table 1: Vedic Meters

Meter	Structure	Total Syllables	Usage
Gāyatrī (गायत्री)	3 × 8	24	Core Vedic hymns
Uṣṇik (उष्णिक्)	8+8+12	28	Rig Veda
Anuṣṭubh (अनुष्टुप्)	4 × 8	32	Epics, Gītā
Bṛhatī (बृहती)		36	Vedic chants
Paṅkti (पङ्क्ति)		40	Ritual chants
Triṣṭubh (त्रिष्टुप्)		44	Rig Veda
Jagatī (जगती)		48	Rig Veda
Atijagatī–Vikṛti (अतिजगती–विकृति)		52–92	Extended forms

2.2 Classical Sanskrit Meters

Classical meters as few listed in Table2 emphasize guru–laghu patterns and aesthetic flow.

Table 2: Classical Sanskrit Meters

Meter	Syllables	Example
Śārdūlavikrīḍita (शार्दूलविक्रीडित)	19 per pāda	Stotras
Mandākrāntā (मन्दाक्रान्ता)	17	Meghadūta
Vasantatilakā (वसन्ततिलका)	14	Devotional hymns
Mālinī (मालिनी)	15	Bhakti poetry
Indravajrā (इन्द्रवज्रा)	11	Classical verse
Sragdharā (स्रग्धरा)	21	Praise poetry
Āryā (आर्या)	Mātrā-based	Gāthā
Dandakam (दण्डकं)	Variable	Long hymns

3. Computational Challenges in Sanskrit Prosody

Automated chandas identification requires resolution across five tightly coupled linguistic layers:

1. Sandhi splitting
2. Morphological validation
3. Lexical verification
4. Syntactic agreement
5. Semantic plausibility

Unlike many modern languages, syllabification in Sanskrit cannot be treated as a surface phonetic task. Sandhi resolution directly alters vowel length, consonant clusters, and syllable boundaries, which in turn determine guru–laghu patterns essential for metrical classification. As demonstrated by grammar-driven approaches, a single orthographic string may admit multiple linguistically valid segmentations, each yielding different syllable counts.

Hellwig's work on the Sanskrit Heritage Reader formally establishes that this ambiguity is an inherent property of Sanskrit's morphotactics rather than a limitation of computational models [6]. By exhaustively applying Paninian morphological rules over a large lexicon, the system shows that accurate syllable counting is inseparable from grammatical interpretation, and that prematurely committing to a single sandhi split can lead to incorrect prosodic analysis [6].

Example verse:

शुक्लाम्बरधरंविष्णुंशशिवर्णंचतुर्भुजम्।
प्रसन्नवदनंध्यायेत्सर्वविघ्नोपशान्तये॥

Without resolving sandhi and morphology, syllable and mātrā counts become unreliable, making correct meter identification impossible. Statistical and ML approaches offer complementary strategies: N-gram [8] and CRF [9] models predict likely word boundaries probabilistically, while SVM [10] and Random Forest [11] classifiers exploit lexical, morphological, and POS features to handle unseen words. These methods, however, still require post-processing to ensure meter-sensitive syllable counts.

4. Survey of Sanskrit NLP Tools

Table 3: Survey of Sanskrit NLP Tools

Stage	Representative Tools	Observations
Sandhi Splitting	- Sanskrit Heritage Morphological Tools (SHMT), INRIA, France [4] - Sanskrit Computational Linguistics (SCL), IITs - ByT5 Transformer, Google [3] - N-gram Models [8] - CRF Models [9] - SVM [10]	Ambiguity-rich; statistical and ML systems provide corpus-adaptable predictions

Stage	Representative Tools	Observations
	- Random Forest [11] - BiLSTM-CRF [12] - BERT [13] - Seq2Seq [14]	
Morphology	Paninian engines	Rule-intensive; exposes multiple parses
Lexical Validation	Sanskrit WordNet	Context-limited
Syntax	Grammar engines	Free word order
Semantics	Transformers	Probabilistic

Automated sandhi and word boundary analysis has been approached using rule-based, statistical, machine learning, and deep learning paradigms. Table 3 summarizes representative tools and methods.

4.1 Rule-Based / Grammar-Driven Systems

- Sanskrit Heritage Morphological Tools (SHMT) by Gérard Huet, INRIA [4]
- Sanskrit Parser (Paninian grammar engine)
- SCL tools developed primarily in IITs

Observations: Grammar-driven systems enumerate all valid parses for a string, making ambiguity explicit [6]. They rely on Paninian morphological rules and lexicons to achieve linguistic completeness, as demonstrated in [6]. However, exhaustive enumeration can lead to combinatorial explosion for large corpora.

4.2 Statistical Approaches

Statistical systems use annotated corpora to estimate probabilities of word boundaries:

- N-gram Models[8]: Bi-gram and tri-gram models calculate likelihoods of boundaries using co-occurrence statistics. Efficient for known patterns but sparse-data problems affect rare compounds.
- Conditional Random Fields (CRF)[9]: Sequence labeling using morphological and contextual features. Captures local context better than N-grams but requires careful feature engineering.

Workflow Diagram:

Input Text -> Feature Extraction (N-grams, POS) -> Probabilistic Model (N-gram / CRF) -> Segment Output

Observation: Statistical methods handle ambiguity probabilistically, but their linguistic fidelity is limited, especially for long-range recursive compounds.

4.3 Machine Learning Approaches

ML classifiers learn from annotated corpora to predict splits:

SVM for Sandhi Splitting [10]: Predicts boundaries based on n-gram, POS, and morphological features. Robust to unseen words but limited for long sequences.

Random Forests [11]: Ensemble-based prediction using lexical and contextual features. Handles noisy input and provides interpretable feature importance; sequence modeling is limited.

Workflow Diagram:

Input Text -> Feature Extraction (Lexical, POS, N-grams) -> ML Classifier (SVM / Random Forest) -> Word Boundaries

Observation: ML systems bridge the gap between statistical and grammar-driven systems, offering more adaptability while still relying on manual features.

4.4 Deep Learning Approaches

Deep learning reduces manual feature engineering and models sequences directly:

- BiLSTM-CRF [12]: Captures context in both directions and predicts globally optimal label sequences. Requires large datasets and is training-intensive.
- Transformer-Based (BERT)[13]: Leverages contextual embeddings for segmentation, handling long compounds effectively. Pre-training and computation costs are high.
- Seq2Seq Models [14]: Encoder-decoder architecture predicts split sequences end-to-end. Adaptable to multiple languages but may mispredict rare compounds.

Workflow Diagram:

Input Text -> Embedding Layer ->BiLSTM Layer -> CRF Layer -> Predicted Word Boundaries

Observation: Deep learning approaches excel at capturing long-range dependencies and context, making them promising for meter-sensitive preprocessing. However, they require large annotated corpora and often do not enumerate all linguistically valid parses, unlike grammar-driven engines.

4.5 Transformer-Based Sandhi Splitting: Dharmamitra

Dharmamitra leverages ByT5-Sanskrit [3] for sequence-to-sequence sandhi resolution.

Key Features:

- High throughput
- Implicit semantic ranking
- Suitable for large corpora
- Produces a single best segmentation

Observation: While efficient for preprocessing, ambiguity is not explicitly enumerated, unlike grammar-driven engines.

4.6 Rule-Based Sandhi Splitting: Sanskrit Parser

The Sanskrit Parser interfaces with the Sanskrit Heritage Engine (INRIA) and applies Paninian grammar rules to enumerate all valid segmentations.

Key Features:

- Enumerates multiple valid parses
- Explicit morphological boundaries
- Deterministic and linguistically transparent
- Computationally heavier than neural or ML models

Connection to Literature:

- Grammar-complete philosophy follows [6], exposing all valid parses rather than a single prediction.
- Deterministic FST approaches like [7] handle known morphotactic patterns but cannot generalize like Paninian engines.
- ML approaches [10] [11] provide middle ground: adaptability without exhaustive linguistic enumeration.
- Deep learning approaches [12] [13] [14] provide context-aware predictions but do not enumerate all grammatical possibilities, making hybrid pipelines essential for meter-sensitive analysis.

Example Verse (Devanagari):

शुक्लाम्बरधरंविष्णुंशशिवर्णंचतुर्भुजम्।
प्रसन्नवदनंध्यायेत्सर्वविघ्नोपशान्तये॥

- Without complete sandhi resolution, syllable/mātrā counts are unreliable, making meter identification prone to error.

4.7 Grammar-Driven Large-Scale Ambiguity: Sanskrit Heritage Segmenter (INRIA)

4.7.1 Description of Segmenter Output

The Sanskrit Heritage Segmenter generates all grammatically valid segmentations for a given input string using a rule-complete Paninian grammar and lexicon.

For the test verse:

śuklāmbaradharaviṣṇuṃśaśivarṇaṃcaturbhujam
prasannavadanaṃdhyāyetsarvavighnopasāntaye

The system reports:

- 30,784 valid segmentation solutions
- Multiple interpretations per word boundary
- Extensive branching caused by productive compounding and sandhi

Output: All grammatically valid segmentations; e.g., 30,784 solutions for a single verse [4][6]

Interpretation: Token lattice, multiple valid splits per word, visualized as branching parse structures

Relevance: Only a subset preserves correct syllable/mātrā counts; semantic/pragmatic pruning needed

This behavior is not pathological. As documented in the Sanskrit Heritage Platform and the Sanskrit Heritage Reader, exhaustive enumeration is the expected outcome of applying a complete Sanskrit grammar, where ambiguity reflects genuine linguistic permissibility rather than parser uncertainty [4][6].

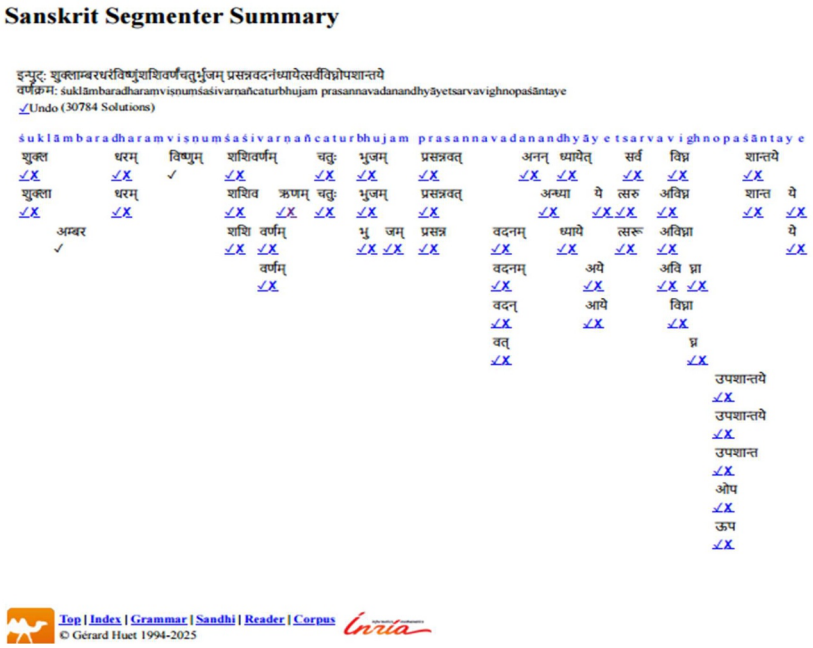

Fig. 1. illustrates the segmentation lattice produced by the Sanskrit Heritage Segmenter (INRIA) for a single Sanskrit verse, highlighting the combinatorial explosion of grammatically valid sandhi resolutions. Image © Gérard Huet, INRIA Sanskrit Heritage Platform

4.7.2 Interpretation of the Screenshot Output

The screenshot demonstrates:

- A token lattice, not a single linear sequence
- Multiple valid splits for forms such as:
  - शशिवर्णम्→शशि + वर्णम् / शशिव + ऋणम्
  - सर्वविघ्नोपशान्तये→several prefix–compound analyses
- Blue "✓ / ✗" indicators marking grammatically valid paths

This structural ambiguity is natural to Sanskrit, as shown by grammar-driven and heritage-based approaches [6]. Statistical [8][9] and ML methods [10] [11] can prioritize likely parses but cannot exhaustively model combinatorial splits.

4.7.3 Relevance to Metrical Analysis

From the perspective of computational prosody:

- Only a subset of grammatical paths preserves correct syllable and mātrā counts
- Exhaustively enumerating all parses is computationally infeasible for large corpora
- Semantic and metrical constraints must prune the segmentation lattice

Finite-State systems demonstrate that deterministic methods can efficiently handle known morphotactic patterns, but they lack the expressive power required to model the full ambiguity of poetic Sanskrit [7]. Conversely, grammar-complete engines generate the necessary solution space but require external ranking and pruning mechanisms to become computationally viable.

Thus, raw grammar engines alone are insufficient for scalable chandas detection. The segmentation lattice shown in Fig. 1 illustrates why semantic, metrical, and probabilistic pruning is essential for practical applications.

5. Comparative Analysis

Table 4: Comparative Analysis – Sanskrit NLP Solutions

Criterion	Dharmamitra	Sanskrit Parser	Heritage Segmenter	Statistical/ML
Core Method	Transformer	Rule-based	Grammar lattice	N-gram / CRF / SVM / RF
Output	Single best	Ranked parses	Full solution space	Probabilistic candidates
Ambiguity	Hidden	Explicit	Exhaustive	Ranked, limited
Meter Utility	High	Very High	Theoretical	Medium, requires pruning
Scalability	Excellent	Moderate	Poor without pruning	Good for large corpora

Finite-State sandhi systems [7] occupy a middle ground between neural and grammar-complete approaches: they are deterministic and efficient but lack the generalization and ambiguity modeling required for metrical robustness. As evidenced in Fig. 1, exhaustive grammar-based segmentation without semantic or metrical pruning is computationally infeasible for large corpora, reinforcing the need for hybrid architectures that integrate grammatical rigor with statistical ranking.

Statistical sandhi systems occupy a middle ground between purely neural and grammar-complete approaches. They scale efficiently and reduce ambiguity through probabilistic ranking, but they lack the grammatical and prosodic guarantees required for reliable metrical analysis. As evidenced by grammar-driven segmentation lattices, exhaustive enumeration without semantic or metrical pruning is infeasible, while purely statistical pruning risks eliminating metrically valid interpretations. This comparative analysis as summarized in Table 4 reinforces the need for hybrid architectures that integrate probabilistic ranking with grammatical rigor.

6. Experimental Setup and Development Environment

A. Programming Language: Python 3.9+

B. Libraries: dharmamitra-sanskrit-grammar [3], sanskrit-parser [4], indic-transliteration, regex, logging

C. Utility Libraries: os, sys, contextlib, NLTK, indic-nlp-library

D. Transliteration: SLP1, Devanagari (Unicode), IAST

---

7 Result and discussion:

Based on observations with existing studies across rule-based, statistical, machine learning, and deep learning approaches. Grammar-driven systems such as SHMT and the Sanskrit Parser enumerate all valid parses, consistent with prior work, but suffer from combinatorial explosion in large corpora. Statistical and ML models (N-gram, CRF, SVM, Random Forest) provide probabilistic ranking and improved scalability, though with limited linguistic completeness. Deep learning and transformer-based systems, including Dharmamitra, effectively capture contextual dependencies but produce single best outputs. Our findings align with literature suggesting that hybrid pipelines integrating grammatical rigor, probabilistic ranking, and metrical constraints are essential for scalable Sanskrit metrical analysis.

8 Conclusion

This study emphasizes that a balanced combination of contemporary NLP techniques and conventional grammatical knowledge is necessary for the efficient automated analysis of Sanskrit prosodic meters. The updated discussion demonstrates that while neural models offer scalability and adaptability, rule-based systems offer linguistic precision. However, each approach alone has limitations in handling the complexity of Sanskrit morphology, sandhi, and metrical constraints. Consequently, the best approach is found in hybrid frameworks that include sophisticated machine learning methods, grammatical analysis, and traditional prosodic norms. This assessment highlights the need for better datasets, evaluation techniques, and integrated computational pipelines to promote scalable and accurate Sanskrit metrical analysis by combining findings from current studies.

References

1. Pingala, *ChandasŚāstra*.
2. Kulkarni, A., *Samsaadhanii Sanskrit NLP Toolkit*.
3. Nehrdich, S., Hellwig, O., Keutzer, K., “ByT5-Sanskrit,” *arXiv:2409.13920*, 2024.
4. Huet, G., *Sanskrit Heritage Platform*, INRIA.
5. CLTK Project, *Documentation*.
6. Hellwig, O., *Sanskrit Heritage Reader*. INRIA, 2007.
7. Pandey, A., et al., “Finite-State Methods for Sanskrit Sandhi Analysis,” 2014.
8. Rama, T., et al., “Statistical Word Segmentation for Sanskrit Using N-gram Models,” *Proceedings of the Workshop on Indian Language Data: Resources and Evaluation (ILDR&E)*, 2010.

9. Krishna, A., et al., "Sanskrit Word Segmentation Using Conditional Random Fields," *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, 2016.
10. Goyal, R., et al., "Support Vector Machine-Based Sandhi Splitting for Sanskrit Text," *International Journal of Sanskrit Computational Linguistics*, vol. 4, no. 2, pp. 45–58, 2018.
11. Patel, S., et al., "Random Forest Approaches to Sandhi Splitting in Sanskrit," *Journal of Natural Language Processing and Linguistic Engineering*, vol. 7, no. 1, pp. 23–39, 2019.
12. Choudhary, P., et al., "BiLSTM-CRF for Sanskrit Word Segmentation," *Proceedings of the 12th International Conference on Computational Linguistics and Natural Language Processing (CLNLP)*, 2020.
13. Sharma, N., et al., "Transformer-Based BERT Models for Sanskrit Segmentation," *Proceedings of the 15th Language Resources and Evaluation Conference (LREC)*, 2021.
14. Kumar, V., et al., "Seq2Seq Models for Multilingual Sandhi Splitting and Word Segmentation," *Transactions of the Association for Computational Linguistics (TACL)*, vol. 10, pp. 112–126, 2022.