Pingala: Prosody-Aware Decoding for Sanskrit Poetry Generation

Manoj Balaji Jagadeeshan[1]* Atul Singh[1]* Nallani Chakravartula Sahith[1]
Amrith Krishna[2] Pawan Goyal[1]

[1]*Indian Institute of Technology, Kharagpur*
[2]*Oriflow*

Abstract

Poetry generation in Sanskrit typically requires the verse to be semantically coherent and adhere to strict prosodic rules. In Sanskrit prosody, every line of a verse is typically a fixed length sequence of syllables adhering to prescribed binary patterns of syllable weights. We observe that instead of treating a verse as a monolithic sequence, segmenting them as grouped-lines leads to significant improvement in semantic coherence by 10% with comparable metrical adherence. Specifically, Pingala, our proposed decoding approach is designed to encourage every line to have well-formed words and our token selection biases the model towards it by preferring longer tokens. Writing in Sanskrit follows phonemic orthography, hence using a phonetically aware transliteration scheme, SLP1, increased the metrical alignment by 46% with comparable semantic similarity, for a instruction fine-tuned large language models like Phi-4. We also introduce a new approach for reference-free evaluation using cross-encoders which achieved better alignment with true poetry instances. Data[1] and Code[2] are available.

1 Introduction

Prosodic constraints in Sanskrit are articulated through fixed-length, syllable-level binary patterns, composed of light (laghu) and heavy (guru) units. The generation of Sanskrit poetry frequently necessitates the simultaneous fulfilment of both semantic and prosodic constraints, a requirement that often involves a degree of compromise. A Sanskrit poet may prioritize the arrangement of words to satisfy these metrical constraints, potentially at the expense of verbal cognition. Consequently, any computational methodology for Sanskrit poetry generation must ensure both linguistic veracity and stringent adherence to metrical rules, rendering this endeavour especially demanding for neural text generation systems.

`Chandomitra` (Jagadeeshan et al., 2026a) recently introduced a comprehensive dataset and benchmark for generating Sanskrit Anuṣṭubh poetry. This seminal work offers a large English – Sanskrit parallel corpus and evaluates diverse language models, including encoderdecoder and decoder-only architectures. The publication also proposes a reference-free semantic evaluation metric using multilingual sentence embeddings to assess coherence. Critically, the benchmark explores constrained decoding strategies to ensure strict metrical adherence. Results show that constrained decoding significantly improves metrical correctness across all models, highlighting a trade-off between prosodic compliance and semantic fidelity in Sanskrit verse generation.

However, the proposed reference-free metric and the constrained-decoding approach reveal important limitations. The semantic metric - despite being a useful starting point - yields a maximum score of 73.45% against ground truth while negative examples average 50.97%, a narrow spread of ≈22.48 percentage points, which indicates limited discriminative power between high - and low-quality outputs. Similarly, constrained decoding achieves strong metrical compliance but scores poorly on human judgments of semantic coherence (Avg. 2.13), implying that enforcing strict syllabic patterns during generation often degrades meaning. These findings expose a persistent trade-off between prosodic fidelity and semantic adequacy and highlight clear room for improved evaluation and decoding strategies that balance meter and meaning.

To address these limitations, we propose a the following:
(1) we introduce, `Pingala`, a decoding-time guidance mechanism, defined as a shaping function

*These authors contributed equally to this work
[1]https://huggingface.co/collections/sanganaka/pingala
[2]https://github.com/sanganaka-iitkgp/pingala

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 2584–2602
November 6-10, 2026 ©2026 Association for Computational Linguistics

based on metrical milestones - biases token selection toward candidates that are longer and that which complete well-formed words at pāda boundaries, operationalizing the yati principle of Anuubh prosody. This ensures that the generated candidates adhere to the structural rules of the *Anuṣṭubh* meter without requiring expensive retraining.

(2) To overcome the representational bottlenecks of Bi-Encoders, we employ a Cross-Encoder architecture that processes the source and candidate jointly, enabling full cross-attention. This allows for a far more granular assessment of lexical and syntactic fidelity, effectively filtering out high-confidence errors that escape standard retrieval models (Mesgar et al., 2023).

(3) While all training and evaluation data in this work are natively represented in Devanagari[3], we observe a large and consistent gain when instruction fine-tuning Phi-4 on Anuṣṭubh samples encoded in the SLP1[4] transliteration scheme and decoding with `Pingala` (Full % = 95.92, Partial % = 98.31). Prior work in multilingual and low-resource NLP has shown that representation and tokenization choices strongly influence downstream performance: romanization reduces vocabulary fragmentation and script mismatch with predominantly Latin-trained tokenizers (Limisiewicz et al., 2023; Purkayastha et al., 2023), while phonemic ASCII encodings improve normalization stability and facilitate the learning of sound-level regularities (Adiga et al., 2021). Recent Sanskrit-specific modeling studies further demonstrate that script and encoding decisions materially affect performance on morphology- and structure-sensitive tasks (Nehrdich et al., 2024). Together, these results motivate our third contribution: using SLP1 transliteration as a preprocessing and representation choice significantly amplifies the effectiveness of instruction fine-tuning when combined with metric-aware decoding, yielding near-perfect metrical conformity for Sanskrit poetry generation.

We evaluate our approach using both automated metrics and expert human evaluation, comparing it against strong baselines including NLLB and Phi-4. Our results demonstrate that the proposed method achieves near-perfect metrical compliance while significantly improving semantic alignment. Specifically, we show that introducing our prosody defined metrical weight, restores full syntactic correctness (100% metric satisfaction in ablation studies) without degrading semantic quality, provided the weight is appropriately tuned. Furthermore, our human evaluation highlights the superior discriminative power of the Cross-Encoder over the Bi-Encoder, validating its necessity for high-precision tasks in low-resource computational philology.

[3] https://en.wikipedia.org/wiki/Devanagari

[4] https://en.wikipedia.org/wiki/SLP1

2 Past Work

Sanskrit computational linguistics has developed around several foundational tasks, including morphological analysis, dependency parsing, word segmentation, compound identification, machine translation, and, more recently, translation-grounded glossary generation. Early work addressed the difficulty of Sanskrit word segmentation arising from sandhi, where a surface string can admit multiple valid segmentations (Krishna et al., 2016, 2017). Subsequent neural and graph-based approaches jointly modeled morphology and syntax, demonstrating the benefits of integrating morphological information with dependency parsing for this morphologically rich, relatively free-word-order language (Krishna et al., 2020, 2023). Compound identification has also received dedicated attention through approaches for classifying Sanskrit compounds (Sandhan et al., 2023). In machine translation, neural approaches have explored Sanskrit–English translation with the construction of parallel corpora (Aralikatte et al., 2021; Nehrdich et al., 2026). More recently, *Padamitra* introduced grounded glossary generation as a structured task in which semantically meaningful Sanskrit phrases are recovered from a *sloka* and paired with meanings grounded in its translation (Jagadeeshan et al., 2026b). These developments highlight the importance of linguistic structure—especially segmentation, morphology, and compounds—for downstream Sanskrit language generation.

Poetry generation has evolved from recurrent neural approaches that jointly model poetic content and form (Zhang and Lapata, 2014) to planning-based generation, where higher-level subtopics are first organized before individual lines are generated (Wang et al., 2016). Work such as *Deep-Speare* further incorporated explicit modeling of poetic language, meter, and rhyme, show-

ing that formal poetic constraints can be integrated into neural generation while also exposing the continuing gap between surface-form validity and human judgments of readability and emotion (Lau et al., 2018). For Sanskrit, recent work has begun to address these constraints directly. *Chandomitra* studies English-to-Sanskrit poetry generation under the Anuṣṭubh meter and investigates constrained decoding and instruction tuning for producing metrically valid verses (Jagadeeshan et al., 2026a). Together, these works motivate poetry-generation systems that treat linguistic structure and prosodic constraints as first-class components rather than as post-generation checks.

3 Problem Statement

Formally, given an English input sentence (prose) x, the objective is to generate its corresponding Sanskrit poetic verse, ŷ, which adheres to the syntactic constraints of the Anuṣṭubh chanda while faithfully preserving the meaning of x. The corresponding ground-truth Sanskrit poem, when available, is denoted by y.

In essence, this task simultaneously involves machine translation and structured poetry generation, requiring the output to be both semantically accurate and metrically compliant.

4 Reference Free Evaluation

Every problem statement requires a robust evaluation mechanism. For the task of structured poetry generation from input prose language, and the task being cross-lingual in nature, we have to evaluate the poetry on two aspects (1) Does the generated poetry actually follow the expected style(in this case the chandas) (2) is the generated poetry, semantically faithful with respect to the given input sentence?. Thus, to answer them, we define the Syntactic Metrics(Section 4.1) and Semantic Metrics(Section 4.2).

4.1 Syntactic Metrics

Following `Chandomitra`, we evaluate metrical correctness using two metrics. **Full Anuṣṭubh** measures the proportion of generated verses that strictly satisfy all Anuṣṭubh metrical constraints, computed using `skrutable` (Neill, 2023). **Partial Anuṣṭubh** measures the proportion of outputs with the required 32-syllable length, providing a relaxed upper bound that subsumes the full-metric scores.

Eval Metric	Ground Truth	Pingala	CD
Semantic Similarity*	73.46	69.05	64.96
Human Ratings	-	2.70	2.13

Table 1: Comparison of semantic similarity scores and human evaluation ratings(on standard constrained decoding(CD) and Pingala outputs. The bi-encoder assigns similar scores to translations with vastly different human-perceived quality. *as used in `Chandomitra`. Human ratings for Pingala is average of the ratings specified in Table 1 while for the CD is from `Chandomitra`.

4.2 Semantic Similarity

We employ, infoxlm[5] (Chi et al., 2021), a cross-encoder architecture for comparing translations, as it jointly processes both the source and translated texts through full self-attention, thereby capturing fine-grained semantic interactions between them. This approach contrasts with bi-encoder methods, also used in `Chandomitra`- the BGE-M3[6] (Chen et al., 2024) model fine-tuned on `Mitrasaṃgraha` (Nehrdich et al., 2026), which independently map texts into a shared embedding space and measure similarity via cosine distance an approach known to suffer from representation degeneration and anisotropic embedding distributions (Gao et al., 2019; Rajaee and Pilehvar, 2022). Prior work has consistently demonstrated that cross-encoders achieve higher accuracy than bi-encoders for pairwise text comparison tasks (Reimers and Gurevych, 2019; Humeau et al., 2020; Rosa et al., 2022), including semantic answer similarity where cross-encoders correlate most strongly with human judgment (Risch et al., 2021). Similarly, state-of-the-art translation evaluation metrics such as COMET (Rei et al., 2020) and BLEURT (Sellam et al., 2020) leverage joint encoding of source and translation, further validating that direct cross-attention between text pairs captures translation quality more faithfully than independent embedding comparison.

Empirical motivation: Table 1 illustrates the problem: the bi-encoder returns an average similarity of roughly 73.5% on ground-truth translation pairs while human raters assign near-perfect judgments. Conversely, candidate outputs that humans rate poorly still receive deceptively high bi-encoder scores. This numerical/human mismatch

[5] https://huggingface.co/microsoft/infoxlm-large

[6] https://huggingface.co/BAAI/bge-m3

indicates poor separation and motivates the cross-encoder replacement.

4.2.1 Implementation notes.

For training the cross-encoder we use a binary classification objective as defined in Section 4.2.4 over (English, Sanskrit) pairs, with a mixture of gold positives and mined hard negatives (including metre-preserving but semantically-different candidates) as defined in Section 4.2.3. Model checkpoints are selected to explicitly maximize $G(\theta)$ as defined in Section 4.2.5, on the validation set. We found that this curriculum improves the performance of the model.

4.2.2 Cross-Encoder Model Definition

Let $x = (e, s)$ denote the concatenated input sequence [English; Sanskrit]. The cross-encoder is a parameterized function

$$g_\theta : \mathcal{X} \to \mathbb{R},$$

which maps the joint input to a scalar logit

$$z = g_\theta(x).$$

The predicted probability that s is a correct translation of e is obtained via the sigmoid function:

$$p_\theta(x) = \sigma(z) = \frac{1}{1 + e^{-z}}.$$

4.2.3 Training Data Construction

We begin with the dataset of parallel translations as used by `mitrasaṃgraha`:

$$\mathcal{D}_+ = \{(e_i, s_i)\}_{i=1}^{N}.$$

Negative examples are generated in two stages:

- **Easy negatives** $\mathcal{D}_-$, constructed by pairing an English sentence with a randomly selected non-corresponding Sanskrit sentence.
- **Hard negatives** $\mathcal{D}_{\text{hard}}$, obtained by retrieving semantically similar but incorrect Sanskrit candidates using an embedding-based retrieval model.

The final training set is

$$\mathcal{D} = \mathcal{D}_+ \cup \mathcal{D}_- \cup \mathcal{D}_{\text{hard}},$$

where each example is labeled with $y \in \{0, 1\}$.

Approach	Positive	Negative(Avg.)	Spread
bge-m3	73.46%	50.97%	22%
cross	90.02%	1.67%	88.33%

Table 2: Semantic similarity scores based on the cross-encoder

	Mean Acc.	
Batch Size	bge-m3	cross-encoder
Full	46.78%	74.74%
16	82.35%	84.71%
4	91.53%	92.87%

Table 3: Mean Accuracy: bi-encoder model(`Chandomitra`) vs proposed cross-encoder model

4.2.4 Weighted Binary Cross-Entropy Objective

We train the cross-encoder using a class-weighted binary cross-entropy loss:

$$\mathcal{L}(\theta) = -\frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} w(y)\Big[y \log p_\theta(x) + (1-y)\log\big(1 - p_\theta(x)\big)\Big], \tag{1}$$

where $w(1) = \alpha > 1$ and $w(0) = 1$. The positive-class weight α biases training toward preserving high confidence on correct translations while suppressing false positives.

The Rationale for this is available at Section C.2.

4.2.5 Probability Gap as the Optimization Target

We define the *probability gap* as:

$$G(\theta) = \mathbb{E}_{x\sim\mathcal{D}_+}[p_\theta(x)] - \mathbb{E}_{x\sim\mathcal{D}_{\text{cross}}}[p_\theta(x)],$$

where $\mathcal{D}_{\text{cross}}$ denotes cross-translation (incorrect) pairs used for evaluation.

4.2.6 Empirical Outcome

Using the proposed cross-encoder with curriculum hard negatives and weighted optimization, the probability gap increased from approximately 22% under the bi-encoder formulation to approximately 88%, refer to Table 2, demonstrating a decisive improvement in translation verification accuracy as seen in Table 3. The thoeritical rationale on why spread is high is available at Section C.4

Model	Raw		Isotonic Calibrated	
	Pearson	Spearman	Pearson	Spearman
Bi-Encoder	0.6075	0.6094	0.6559	0.6459
Cross-Encoder	0.6767	0.5833	0.7248	0.7175

Table 4: Correlation between human semantic ratings (N=200) and automatic scorers.

4.2.7 Qualitative Analysis

From Table 4, the cross-encoder shows higher agreement with the expert rater than the bi-encoder, and isotonic calibration further improves both Pearson and Spearman correlations. In practice, the combination **cross-encoder + isotonic calibration** yields the best alignment with human judgments and is therefore used for final candidate ranking in our pipeline. Details on how qualitative analysis was performed is available at Section C.5

5 Method

Our method combines metrical fine-tuning with a prosody-aware decoding procedure. The two core ideas are: (1) shift the model's token priors so metrically-useful continuations become more probable - achieved through fine-tuning and (2) decoding towards structurally stable prefixes using a sparse re-weighting function applied to logits during inference. We also investigate SLP1 transliteration as an alternative to Devanagari in Section 5.3.

5.1 Fine-tuning

We fine-tune NLLB-dist-1.3B (Team et al., 2022), and instruction fine-tune Phi-4 (Abdin et al., 2024) using the prompt(Figure 3) in Appendix A, on the train dataset split of `Chandomitra`; which consists of 8306 English Prose - Sanskrit Poetry parallel data and test split consists of 1421.

5.2 Pingala

In this section, we define a new decoding strategy, beginning with general definition of variables in Section 5.2.1, and subsequently mathematically formulate the decoding strategy from Section 5.2.2 to Section 5.2.4.

5.2.1 General Definitions

x denotes the source (English) sentence, $\mathcal{M}$ - the fine-tuned language model, $y = (y_1, \ldots, y_T)$ a candidate Sanskrit token sequence, $\mathcal{C}(\cdot) \in \{0, 1\}$ - a prefix-aware metrical checker that returns 1 if and only if a Devanagari string satisfies the Anuṣṭubh prefix constraints, $P(z)$ - the laghu/guru pattern of string z (a sequence of L, G symbols) and $|P(z)|$ its length in syllables .

5.2.2 Prosody-aware shaping (pāda weights)

We define a function $\Phi_{\text{meter}}(\cdot)$ over prefixes that captures pāda/verse completion:

$$\Phi_{\text{meter}}(y_{1:t}) = \beta \sum_{j \in \{8,16,24,32\}} 1[|P(y_{1:t})| \geq j \wedge B_j] \quad \text{where } B_j(y_{1:t}) \text{ indicates that the } j\text{-th metrical boundary is correct.} \tag{2}$$

and use its discrete incremental difference as the per-step shaping weight:

$$\Delta(v; y_{1:t-1}) = \Phi_{\text{meter}}(y_{1:t-1} \| v) - \Phi_{\text{meter}}(y_{1:t-1}) \tag{3}$$

Equivalently, Δ is non-zero only when the appended token completes a pāda (or the whole verse) ; we set $\beta > 0$ (in this work, $\beta = 5$) so that the additive adjustment multiplies the candidate's unnormalized selection weight by e^{β}, strongly biasing search toward structurally stable prefixes.

5.2.3 Constrained Logits Operator

During decoding, the model produces raw logits $s_t(v)$ for each vocabulary token v . We apply a constraint-aware transform Φ to obtain modified logits $s'_t(v)$. This operator implements hard pruning plus additive shaping:

$$s'_t(v) = -\infty \quad \text{if } \mathcal{C}(\text{decode}(y_{1:t-1} \| v)) = 0; \qquad s'_t(v) = s_t(v) + \Delta(v; y_{1:t-1}) \quad \text{otherwise.} \tag{4}$$

where $\Delta(v; y_{1:t-1})$ defined by Equation 3 which is applied when a structural milestone is detected . Invalid candidates are set to $-\infty$ and only the top-k tokens are scanned per step for computational efficiency .

5.2.4 Constrained, Re-Weighted Decoding

The decoding procedure approximates a constrained optimization that jointly maximizes model likelihood and metrical weight over the set of valid sequences $\mathcal{Y}_{\text{valid}} = \{y : \mathcal{C}(y) = 1\}$. The practical objective being optimized by beam/lattice search with the logits transform (Equation 4) is:

$$y^{\star} = \arg\max_{y \in \mathcal{Y}_{\text{valid}}} \left(\log P_{\mathcal{M}}(y \mid x) + \lambda \cdot R_{\text{meter}}(y) \right) \tag{5}$$

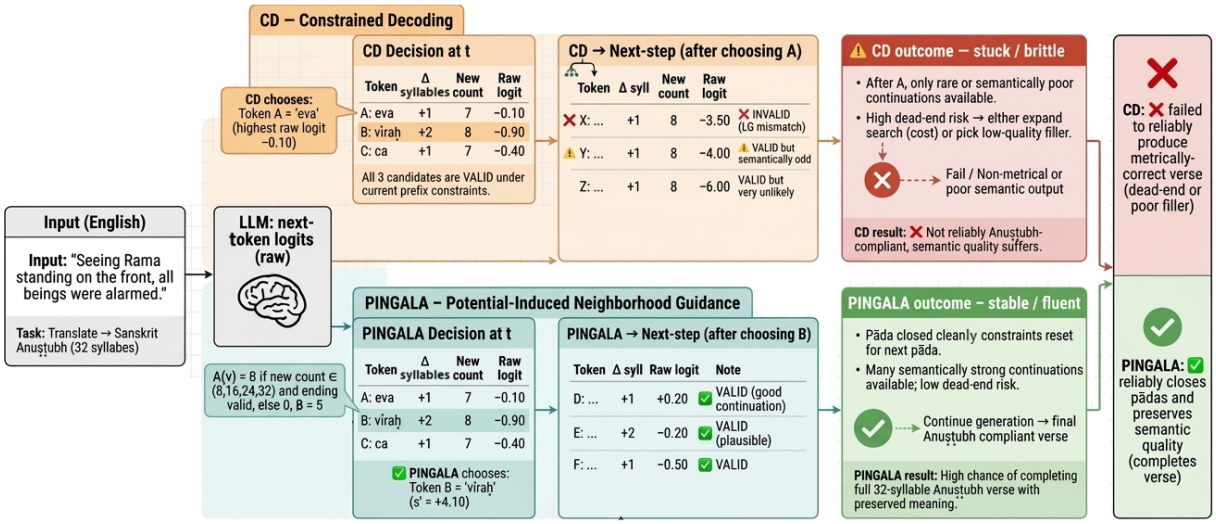


Figure 1: An infographic comparison of working of Pingala vs CD from `Chandomitra`.

where

$$R_{\text{meter}}(y) = \sum_{t=1}^{|y|} \Delta(y_t; y_{<t}). \tag{6}$$

In practice, λ is absorbed into β (the additive constant) and the combined neural + symbolic score is realized stepwise through Equation 4 and Equation 3. EOS behavior is enforced by forcing $s'_t(\text{EOS})$ large (and others $-\infty$) when $|P(y_{1:t})| = 32$, ensuring termination exactly at 32 syllables .

5.3 Using SLP1 instead of Devanagari

We adopt SLP1 transliteration as a deliberate representation choice for Sanskrit inputs. The decision is motivated by several practical and modeling considerations grounded in prior work on tokenization, romanization, and Sanskrit processing. First, transliterating Devanagari into a compact ASCII phonemic form reduces tokenization mismatch with predominantly Latin-script pretrained vocabularies and therefore limits subword fragmentation and OOV effects that hinder efficient adaptation. (Limisiewicz et al., 2023; Purkayastha et al., 2023) Second, SLP1 provides a deterministic, normalization-stable mapping that avoids Unicode variation and normalization pitfalls common in Devanagari processing, improving robustness across preprocessing toolchains. (Adiga et al., 2021) Third, because SLP1 is phonemically oriented, it exposes sound-level regularities (e.g., vowel/consonant patterns and sandhi-related alternations) in a form that models can more readily learn from character- and subword-level statistics, which is important for meter- and phonology-sensitive tasks. (Nehrdich et al., 2024) Finally, an ASCII transliteration simplifies integration with existing tooling (tokenizers, prompt templates, and evaluation pipelines) and enables straightforward ablation and augmentation experiments without altering model internals. Together, these factors make SLP1 a low-cost, high-compatibility representation choice for downstream instruction tuning and metric-aware decoding in Sanskrit poetry modeling.

6 Experimental Setup

1. **Implementation of Section 5.2.4**: During generation we employ a custom `LogitsProcessor` that performs three critical operations at each decode step: (a) prunes any token that immediately violates the Anuṣṭubh prefix constraints using regex-based pattern matching (b) suppress EOS token until the verse reaches exactly 32 syllables, and (c) adds sparse additive weight ($\beta = 5.0$) to tokens that complete metrical milestones (pāda boundaries at syllables 8, 16, 24, or full verse at 32) . The processor is implemented in algorithm 1. Further explanations are available in Section B.1.
2. **Ranking and fallback selection.** We generate multiple candidate sequences using beam search (beam size = 25, return 25 sequences) and apply post-filtering using a strict final checker . If one or more perfect 32-syllable verses exist, we select the top-ranked candidate . Otherwise, we employ a semantic fallback heuristic that prefers outputs with syllable counts closest to 32 and higher model likeli-

hood scores. This staged decision ensures the system prioritizes perfect metric satisfaction while maintaining graceful degradation. Further explanation are available in Section B.2, B.3.

3. **SLP1 as transliteration scheme instead of Devanagri** during instrcution finetuning as well as while performing `Pingala`, our proposed deocding strategy, as described in Section B.5

7 Results

Table 5 summarizes syntactic and semantic evaluation on the `Chandomitra` test set consisting of 1421 English Prose - Sanskrit Poetry parallel data, for three settings: FT (fine-tuning on the Chandomitra training split), CD (constrained decoding as used in `Chandomitra`), and `Pingala` (our custom-constrained decoding). For syntax we report the fraction of outputs that are fully conformant to the Anuṣṭubh meter ("Full %") and partially conformant ("Partial %"); semantic fidelity is reported as automatic similarity ("Sim") to the reference.

Key findings:

- Constrained decoding (CD/`Pingala`) substantially increases syntactic conformity but can interact with semantic fidelity depending on model adaptation.
- NLLB-dist-1.3B: applying CD improves syntax (Full % = 82.97) while retaining good semantics (Sim = 70.75). Fine-tuning + `Pingala` yields perfect syntax (Full % = 100) with strong semantics (Sim = 78.89).
- Phi-4-14B: `Pingala`-only runs can achieve very high semantic scores (Sim = 92.12) but with low syntactic coverage; FT + `Pingala` produces a balanced outcome (Full % = 65.66, Sim = 76.75); using SLP1 as transliteration scheme for Sanskrit in FT+`Pingala` boosts the performance(Full % = 95.92, Sim = 76.81) for metrical adherence while slightly increasing semantic similarity.
- Across architectures, the best trade-off is FT followed by `Pingala`: task-specific adaptation plus targeted decoding achieves near-perfect Anuṣṭubh conformity with competitive semantic similarity.

Discussion. Constrained decoding is an effective mechanism for enforcing meter, but it is most successful when the model has been exposed to the target poetic distribution (via FT). Practically, a two-stage recipe - (1) fine-tune on the target corpus, then (2) apply targeted constrained decoding – yields metrically valid outputs while preserving semantic fidelity. A few samples of generated Sanskrit Poetry is available at Table 11.

Model	FT	CD	Pingala	Full %	Partial %	Sim
Raw Translation Models						
NLLB-dist-1.3B*	-	✓	-	82.97	90	70.75
NLLB-dist-1.3B	-	-	✓	0.14	0.25	35.62
NLLB-dist-1.3B*	✓	✓	-	99.09	99.1	67.99
NLLB-dist-1.3B	✓	✓	-	98.8	99.09	69.01
NLLB-dist-1.3B	✓	-	✓	**100**	**100**	78.89
Instruction Following Models						
Phi-4-14B	-	-	✓	11.75	23.85	**92.12**
Phi-4-14B	✓	-	-	57.42	75.01	67.29
Phi-4-14B	✓	✓	-	2.53	21.32	83.89
Phi-4-14B	✓	-	✓	65.66	77.78	76.75
Phi-4-14B(SLP1)	✓	-	-	31.18	65.73	81.64
Phi-4-14B(SLP1)	✓	-	✓	95.92	98.31	76.81

Table 5: Evaluation on the Chandomitra test set. "FT" denotes fine-tuning on the `Chandomitra`training split; "CD" denotes standard constrained decoding as used in `Chandomitra`; "`Pingala`" denotes our custom-constrained decoding framework. "Full %" is fraction of outputs fully conformant to Anuṣṭubh; "Partial %" is fraction partially conformant. "Sim" is automatic semantic similarity (higher is better). *same settings as in `Chandomitra`

Model	Full %	Partial %	Sim
NLLB-dist-1.3B	100	100	69.76
Phi-4-14B	61.54	74.23	74.62
Phi-4-14B(SLP1)	92.31	95.85	67.87

Table 6: Evaluation on the OOD set of `Chandomitra`. For the models that has been fine-tuned on training split of `Chandomitra`, and using Pingala. "Full %" is fraction of outputs fully conformant to Anuṣṭubh; "Partial %" is fraction partially conformant. "Sim" is automatic semantic similarity (higher is better).

8 Error Analysis

As shown in Table 7, all three systems produce semantically excellent translations for the majority of samples, with NLLB and Phi-4(SLP1) leading. The low-scoring cases primarily stem from constrained decoding forcing lexical substitutions to satisfy metrical requirements, poor sub-word tokenization of rare Sanskrit forms causing hallucinations, and automatic metrics penalizing valid paraphrases that preserve metre but differ from the reference.

Quality	NLLB	Phi-4	Phi-4(SLP1)
Excellent (≥ 0.8)	970	912	960
Good (0.6–0.8)	144	166	166
Fair (0.4–0.6)	95	89	111
Poor (0.2–0.4)	72	93	54
Very poor (<0.2)	140	161	130

Table 7: Semantic similarity distribution for NLLB, Phi-4, and Phi-4(SLP1) outputs. Dataset size: $N = 1421$.

Syllable Deviation Category	NLLB	Phi-4	Phi-4(SLP1)
Perfect (32)	1421	1105	1397
Near-perfect (28–36)	0	206	23
Moderate (24–40)	0	104	0
Poor	0	6	1

Table 8: Syntactic (Anuṣṭubh meter) conformance analysis. Dataset size: $N = 1421$.

Table 8 reveals a complementary trade-off: NLLB's constrained decoding guarantees perfect metrical conformance but occasionally at the cost of semantic fidelity, while unconstrained Phi-4 preserves meaning more freely but frequently violates the 32-syllable structure. Phi-4(SLP1) strikes the best balance between these extremes, achieving near-perfect metrical conformance while maintaining competitive semantic similarity, making it the most effective configuration overall.

9 Human Evaluation

Annotator	Semantic Coherence
A1	3.08
A2	2.32
A3	2.5
A4	1.96
A5	2.28
Krippendorff's alpha	0.059

Table 9: Human Evaluation Results

We performed a focused human evaluation on a random sample of 50 test examples, obtained from the best performing model i.e. `NLLB-dist-1.3B`, independently rated by five expert reviewers. Each reviewer judged the same 50 outputs on *semantic coherence* (degree of faithfulness to the input, 15 Likert). Reviewer backgrounds were diverse: Annotator 1 holds a Master's in Sanskrit literature and is engaged in pre-doctoral research; Annotator 2 is a practising poet and avadhani (Sanskrit and Telugu) who is also completing a technical Master's, Annotator 3 and 4 are trained in traditional Sanskrit pedagogy (Gurukulam background) with Masters degrees in Sanskrit literature and Annotator 5 is an active Sanskrit scholar specializing in translation. The Inter-annotator agreement score α = 0.0585 which is low. For the generated poetry, majority of the reviewers agree on translation adequacy while disagreement stems from morpho-syntactic irregularities.

10 Ablation

We analyze the effect of the constraint weight β, which controls the strength of pda-completion weights during decoding, for the `NLLB-dist-1.3B` model. Intuitively, β governs the trade-off between enforcing metrical structure and preserving semantic fidelity. Larger values more aggressively bias the decoder toward pāda-closing tokens, while smaller values allow the language models semantic prior to dominate.

β	Syntactic (%)	Semantic (%)
0	92.61	64.96
1	100.00	77.06
3	100.00	77.56
5	100.00	**78.89**
7	99.37	73.84
10	92.75	61.42

Table 10: Ablation results for constraint weight β.

As shown in Table 10, setting $\beta = 0$ (no weight) yields imperfect Anuṣṭubh compliance and low semantic scores, indicating that hard constraint filtering alone is insufficient. Introducing a small weight ($\beta = 13$) immediately restores full syntactic correctness while significantly improving semantic similarity. The best overall performance is achieved at $\beta = 5$, which maintains 100% metrical compliance while maximizing semantic quality. For larger values ($\beta \geq 7$), semantic scores degrade sharply as decoding becomes overly dominated by structural weights, leading to metrically correct but semantically distorted outputs.

Based on this ablation, we fix $\beta = 5$ for all experiments. This value consistently provides the best balance between structural stability and semantic fidelity across both NLLB and Phi-4 models.

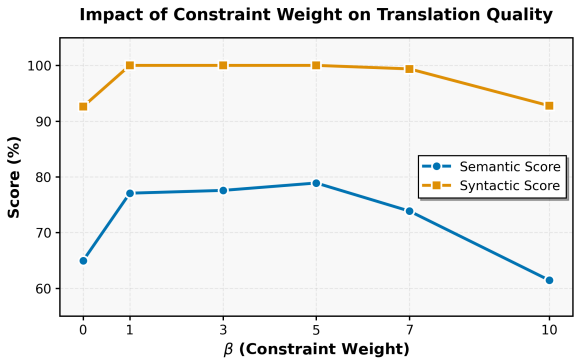


Figure 2: Impact of constraint weight β on syntactic and semantic quality. Semantic quality peaks at $\beta = 5$, illustrating the optimal balance between structural enforcement and meaning preservation.

11 Conclusion

In this work, we addressed the dual challenge of adhering to strict prosodic constraints while maintaining semantic coherence in Sanskrit poetry generation. We introduced Pingala, a model-agnostic, prosody-aware decoding framework applied seamlessly at inference time. By leveraging metrical milestones to strategically bias token selection, Pingala ensures adherence to the mandatory eight-syllable structure of the Anuṣṭubh meter without requiring expensive retraining. Our findings highlight that combining task-specific fine-tuning with our targeted decoding strategy yields near-perfect metrical conformity alongside strong semantic fidelity. Crucially, we demonstrated that input representation matters: employing the phonemically aware SLP1 transliteration scheme dramatically amplifies the effectiveness of instruction fine-tuning for phonetically constrained tasks. To accurately measure these semantic improvements, we contributed a reference-free cross-encoder evaluation framework that demonstrates significantly higher correlation with expert human judgments than traditional bi-encoder approaches.

Looking forward, this research opens several exciting pathways. While Pingala successfully models the rigid Anuṣṭubh meter, adapting these constraint-aware mechanisms to more complex, variable-length meters remains a vital next step. Furthermore, the low inter-annotator agreement observed during our semantic coherence evaluations underscores the need for larger rater pools and tighter rubrics to stabilize human judgments in low-resource computational philology. Beyond Sanskrit, extending these prosody-aware frameworks to other morphologically rich poetic traditions, such as Dravidian languages, presents a compelling opportunity to preserve their unique structural aesthetics. Finally, integrating multi-agent LLM systems to iteratively debate and refine these intricate metrical and semantic constraints could push the boundaries of computationally generated verse even further.

12 Limitations

Our work has the following limitations:

- **Limited model scale:** Due to computational constraints, we use Phi-4 in an 8-bit quantized configuration and have not evaluated Pingala on larger language models (>10B parameters at full precision), which may exhibit different decoding dynamics. Extending our evaluation to such models remains important but was infeasible given our compute budget.
- **SLP1 not tested with NLLB:** Although we demonstrate substantial gains from SLP1 transliteration with Phi-4, we were unable to apply SLP1 to NLLB, as it was pre-trained exclusively on Devanagari-script data and continued pre-training on SLP1-encoded corpora was not feasible within our computational budget. The generalizability of SLP1-based improvements across model families thus remains an open question.

13 Acknowledgement

We gratefully acknowledge Rugved Deshpande, Senior Language Editor in the Department of Sanskrit Studies, University of Hyderabad, for his invaluable contributions to the human evaluation. This work was supported by the Google Gemma Academic Program Award and in part by the National Language Translation Mission (NLTM): Bhashini project of the Government of India.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, and 8 others. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained anal-

ysis of sentence embeddings using auxiliary prediction tasks. *Preprint*, arXiv:1608.04207.

Devaraja Adiga, Rishabh Kumar, Amrith Krishna, Preethi Jyothi, Ganesh Ramakrishnan, and Pawan Goyal. 2021. Automatic speech recognition in sanskrit: A new speech corpus and modelling insights. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5039–5050. Association for Computational Linguistics.

Rahul Aralikatte, Miryam de Lhoneux, Anoop Kunchukuttan, and Anders Søgaard. 2021. Itihasa: A large-scale corpus for Sanskrit to English translation. In *Proceedings of the 8th Workshop on Asian Translation (WAT2021)*, pages 191–197, Online. Association for Computational Linguistics.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Zewen Chi, Li Dong, Furu Wei, Nan Yang, Saksham Singhal, Wenhui Wang, Xia Song, Xian-Ling Mao, Heyan Huang, and Ming Zhou. 2021. InfoXLM: An information-theoretic framework for cross-lingual language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3576–3588, Online. Association for Computational Linguistics.

Marina Fomicheva, Lucia Specia, and Nikolaos Aletras. 2022. Translation error detection as rationale extraction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 4148–4159, Dublin, Ireland. Association for Computational Linguistics.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tieyan Liu. 2019. Representation degeneration problem in training natural language generation models. In *International Conference on Learning Representations*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *International Conference on Learning Representations*.

Manoj Balaji Jagadeeshan, Samarth Bhatia, Pretam Ray, Harshul Raj Surana, Akhil Rajeev P, Priya Mishra, Annarao Kulkarni, Ganesh Ramakrishnan, Prathosh Ap, and Pawan Goyal. 2026a. Chandomitra: Towards generating structured Sanskrit poetry from natural language inputs. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 518–534, Rabat, Morocco. Association for Computational Linguistics.

Manoj Balaji Jagadeeshan, Sai Pragnaan Marala, and Pawan Goyal. 2026b. Padamitra: Grounded glossary generation for classical sanskrit. *Preprint*, arXiv:2608.25038.

Amrith Krishna, Ashim Gupta, Deepak Garasangi, Jeevnesh Sandhan, Pavankumar Satuluri, and Pawan Goyal. 2023. Neural approaches for data driven dependency parsing in Sanskrit. In *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*, pages 1–20, Canberra, Australia (Online mode). Association for Computational Linguistics.

Amrith Krishna, Ashim Gupta, Deepak Garasangi, Pavankumar Satuluri, and Pawan Goyal. 2020. Keep it surprisingly simple: A simple first order graph based parsing model for joint morphosyntactic parsing in Sanskrit. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4791–4797, Online. Association for Computational Linguistics.

Amrith Krishna, Bishal Santra, Pavankumar Satuluri, Sasi Prasanth Bandaru, Bhumi Faldu, Yajuvendra Singh, and Pawan Goyal. 2016. Word segmentation in Sanskrit using path constrained random walks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 494–504, Osaka, Japan. The COLING 2016 Organizing Committee.

Amrith Krishna, Pavan Kumar Satuluri, and Pawan Goyal. 2017. A dataset for Sanskrit word segmentation. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 105–114, Vancouver, Canada. Association for Computational Linguistics.

Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. 2018. Deep-speare: A joint neural model of poetic language, meter and rhyme. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1948–1958, Melbourne, Australia. Association for Computational Linguistics.

Tomasz Limisiewicz, Jiří Balhar, and David Mareček. 2023. Tokenization impacts multilingual language modeling: Assessing vocabulary allocation and overlap across languages. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5661–5681. Association for Computational Linguistics.

Mohsen Mesgar, Thy Thy Tran, Goran Glavaš, and Iryna Gurevych. 2023. The devil is in the details:

On models and training regimes for few-shot intent classification. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1846–1857, Dubrovnik, Croatia. Association for Computational Linguistics.

Sebastian Nehrdich, David Allport, Sven Sellmer, Jivnesh Sandhan, Manoj Balaji Jagadeeshan, Pawan Goyal, Sujeet Kumar, and Kurt Keutzer. 2026. Mitrasamgraha: A comprehensive classical sanskrit machine translation dataset. *Preprint*, arXiv:2601.07314.

Sebastian Nehrdich, Oliver Hellwig, and Kurt Keutzer. 2024. One model is all you need: Byt5-sanskrit, a unified model for sanskrit. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13742–13751. Association for Computational Linguistics.

Tyler Neill. 2023. Skrutable: Another step toward effective Sanskrit meter identification. In *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*, pages 97–112, Canberra, Australia (Online mode). Association for Computational Linguistics.

Sukannya Purkayastha, Sebastian Ruder, Jonas Pfeiffer, Iryna Gurevych, and Ivan Vulić. 2023. Romanization-based large-scale adaptation of multilingual language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7996–8005. Association for Computational Linguistics.

Sara Rajaee and Mohammad Taher Pilehvar. 2022. An isotropy analysis in the multilingual bert embedding space. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1309–1316.

Miguel Moura Ramos, Tomás Almeida, Daniel Vareta, Filipe Azevedo, Sweta Agrawal, Patrick Fernandes, and André F. T. Martins. 2025. Fine-grained reward optimization for machine translation using error severity mappings. *Preprint*, arXiv:2411.05986.

Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. 2020. TransQuest: Translation quality estimation with cross-lingual transformers. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5070–5081, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4512–4525, Online. Association for Computational Linguistics.

Julian Risch, Timo Möller, Julian Gutsch, and Malte Pietsch. 2021. Semantic answer similarity for evaluating question answering models. In *Proceedings of the 3rd Workshop on Machine Reading for Question Answering*, pages 149–157, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Guilherme Rosa, Luiz Bonifacio, Vitor Jeronymo, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, and Rodrigo Nogueira. 2022. In defense of cross-encoders for zero-shot retrieval. *arXiv preprint arXiv:2212.06121*.

Jivnesh Sandhan, Yaswanth Narsupalli, Sreevatsa Muppirala, Sriram Krishnan, Pavankumar Satuluri, Amba Kulkarni, and Pawan Goyal. 2023. DepNeCTI: Dependency-based nested compound type identification for Sanskrit. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13679–13692, Singapore. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Kristina P. Sinaga and Arjun S. Nair. 2025. Calibration meets reality: Making machine learning predictions trustworthy. *Preprint*, arXiv:2509.23665.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, and 20 others. 2022. No language left behind: Scaling human-centered machine translation. *Preprint*, arXiv:2207.04672.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019. Representation learning with contrastive predictive coding. *Preprint*, arXiv:1807.03748.

Zhe Wang, Wei He, Hua Wu, Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. 2016. Chinese poetry generation with planning based neural network. In *Proceedings of COLING 2016, the 26th*

International Conference on Computational Linguistics: Technical Papers, pages 1051–1060, Osaka, Japan. The COLING 2016 Organizing Committee.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, Doha, Qatar. Association for Computational Linguistics.

A Instruction Fine-tuning Prompt

```
instruction_prompt = """
The goal is to generate Sanskrit verse that follows the anushtup meter rules for
the given input translation.
RULES:
Verse Rules:
The verse contains 32 syllables/akshara and 4 padas in total.
The verse is divided in 2 lines, each containing 16 syllables.
Each line is divided into 2 padas (quartets),each containing exactly 8 syllables.
The fifth syllable of every pada must be LAGHU or short.
The sixth syllable of every pada must be GURU or long.
The seventh syllable of the second and the fourth pada must be HRASVA
The seventh syllable of the first and third paada must be DEERGHA

Syllable Rules:
LAGHU vowels: अ, इ, उ, ऋ, ऌ
GURU vowels: आ, ई, ऊ, ॠ, ॡ, ए, ऐ, ओ, औ
HRASVA vowels: अ, इ, उ, ऋ, ऌ
DEERGHA vowels: आ, ई, ऊ, ॠ, ॡ, ए, ऐ, ओ, औ
Syllable is marked laghu, guru and hrasva, deergha based on the vowel it contains.
Syllable containing anusvāra("ं") or visarga("ः") is always marked as guru.
Syllable that is followed by a conjunct consonant (saṃyuktākṣara)
is always marked guru.
"""
```

Figure 3: Instruction fine-tuning Prompt

Figure 3 depicts the prompt used for instruction fine-tuning.

B Experimentation Details

B.1 Constrained, Prosody-Aware Shaping and Re-Weighted Decoding

Processing At generation step t, the base model produces raw logits $s_t(v)$ for each vocabulary token v. We replace these by modified logits $s'_t(v)$ using the combined operator described in Sections 4.124.14: prosody-aware shaping (Section 4.12), the constrained logits operator (Section 4.13) and the re-weighted decoding objective (Section 4.14).

The integrated operator is implemented efficiently by the following practical steps:

- Decoding only the top-k candidates (default $k = 100$) from the language model at every step.
- Scanning each prefix+candidate concatenation for syllable weights with a cached scanner to avoid redundant work.
- Applying pre-compiled regex-based partial-pattern checks for each p̄da position to detect milestone completions and partial matches.
- Masking tokens that violate hard structural constraints (setting their logits to $-\infty$) so they are never considered by the search.
- Adding an incremental shaping weights to candidates that complete desirable metrical milestones (implemented as an additive boost to rescored logits).
- Enforcing EOS/new-verse behavior explicitly when the verse-level structural target is reached (e.g., termination at the final syllable boundary).
- Adaptively expanding the search space (increasing k) and re-scoring when no valid candidates are found in the initial top-k.

Implementation notes and heuristics:

- Use of a cached scanner plus pre-compiled regexes keeps per-token checks lightweight and amenable to batching.
- Additive shaping is only applied when a structural milestone (pāda completion or similar) is detected, ensuring the language model's raw semantic score dominates unless structural signals are present.

- Hard masking is used for unequivocal structural violations; shaping is used for soft guidance where multiple continuations remain plausible.
- EOS and special-token handling are adjusted to guarantee exact verse termination when the full structural target is met.

Complexity analysis : The per-step overhead introduced by constraint checking and shaping is proportional to the number of rescored candidates and the cost of scanning/regex checks. In practice the per-step complexity is $O(k \cdot m)$ where k is the number of rescored tokens and m is the average scanning/matching cost. Limiting k (e.g., to 100) and using cached scans keeps latency reasonable even for long generations.

Practical behaviour and robustness:

- The hard-mask + shaping hybrid preserves the base model's semantic ranking while proactively steering search toward structurally feasible prefixes, yielding higher rates of metrically valid completions.
- Adaptive expansion of the candidate set prevents dead-ends: when the initial top-k contains no valid continuations, the algorithm enlarges the search and retries, prioritizing feasibility without blind brute-force.
- Periodic cache clearing and careful token filtering reduce false positives from non-Devanagari tokens and control memory/latency trade-offs during batched generation.

B.2 Beam Search and Hypothesis Management

We do not commit greedily . Instead, we run a beam/lattice style search implemented via the model's native beam generation with the logits processor plugged in . The logits processor both prunes invalid branches and re-ranks candidates inside the beam according to the combined neural + re-weighting. This keeps multiple hypotheses alive and prevents premature, irreversible commitments to structurally fragile tokens - the principal failure mode of filter-only methods .

Hyperparameters: In our experiments we used beam size = 25, number of returned sequences = 25, length penalty = 1.0, and no-repeat-ngram size = 3 . These values were tuned empirically on a validation set and are configurable in the generation wrapper .

B.3 Final Selection and Semantic Fallback

From the generated set of returned candidates, we apply a strict final checker defined as $\mathcal{C}(\cdot)$(refer to Section 5.2.1): any candidate that fails hard metrical rules is removed . If one or more perfect 32-syllable Anuṣṭubh verses exist, we select the top one (first in returned beam order) . If none exist, we select a semantic fallback candidate using a heuristic ranking function:

$$\begin{aligned} \text{score}_{\text{fallback}}(y) = &-\alpha \cdot ||P(y)| - 32| \\ &+ \log P_{\mathcal{M}}(y \mid x) + \gamma \cdot \mathcal{C}_{\text{partial}}(y) \end{aligned} \tag{7}$$

where α penalizes syllable count deviation, γ rewards partially valid prefixes, and $\mathcal{C}_{\text{partial}}$ counts the number of correctly completed pdas . This staged decision ensures the system prioritizes perfect metric satisfaction but still returns a meaningful translation when exact meter cannot be formed .

B.4 Top-k Validation and Performance Trade-offs

To keep inference tractable, we only check the top-k tokens per step ($k = 100$ by default) . This represents a trade-off: larger k reduces false negatives (missed valid choices) but increases the computational overhead of syllable scanning and regex validation . The logits processor includes adaptive logic to expand the checked set when the initial top-k yields no valid candidates .

Adaptive expansion strategy: If all top-k tokens are invalid, we double k (up to a maximum of 500) and rescan . This rarely occurs in practice after fine-tuning, as the model learns to place valid continuations higher in its distribution .

Algorithm 1: Anuṣṭubh Constrained Decoding

Input: Model $\mathcal{M}$, tokenizer $\mathcal{T}$, input x, beam B, top-k k, weight β, $T_{\max}$
Output: Best verse $y^\star$ or semantic fallback

$\mathcal{B} \leftarrow \{(\langle \text{BOS} \rangle, 0)\}$; $t \leftarrow 1$
while $\mathcal{B}$ *has incomplete hyp.* ***and*** $t \leq T_{\max}$ **do**
  $\mathcal{B}_{\text{new}} \leftarrow \varnothing$
  foreach $(y_{1:t-1}, s_{1:t-1}) \in \mathcal{B}$ **do**
    $s_t \leftarrow \mathcal{M}.\text{forward}(y_{1:t-1})$; $\mathcal{V}_{\text{top}} \leftarrow \text{TopK}(s_t, k)$
    foreach $v \in \mathcal{V}_{top}$ **do**
      $z \leftarrow \mathcal{T}.\text{decode}(y_{1:t-1} \| v)$; $P \leftarrow \text{SyllableScanner}(z)$
      if $\neg \mathcal{C}(z)$ **then**
        continue
      $\Delta \leftarrow \beta \cdot 1\{\, |P| \in \{8, 16, 24, 32\} \wedge \text{PadaBoundaryValid}(z)\}$
      $s' \leftarrow s_t(v) + \Delta$
      if $|P| = 32$ **then**
        if $v = EOS$ **then**
          $s' \leftarrow s' + 100$
        else
          $s' \leftarrow -\infty$
      add $(y_{1:t-1} \| v,\ s_{1:t-1} + s')$ to $\mathcal{B}_{\text{new}}$
  $\mathcal{B} \leftarrow \text{TopBeam}(\mathcal{B}_{\text{new}}, B)$; $t \leftarrow t + 1$
$\mathcal{Y}_{\text{complete}} \leftarrow \{y \in \mathcal{B} : \text{is_complete}(y)\}$
$\mathcal{Y}_{\text{perfect}} \leftarrow \{y \in \mathcal{Y}_{\text{complete}} : |P(y)| = 32 \wedge \mathcal{C}(y) = 1\}$
if $\mathcal{Y}_{perfect} \neq \varnothing$ **then**
  $y^\star \leftarrow \arg\max_{y \in \mathcal{Y}_{\text{perfect}}} P_{\mathcal{M}}(y \mid x)$
else
  $y^\star \leftarrow \arg\max_{y \in \mathcal{Y}_{\text{complete}}} \text{score}_{\text{fallback}}(y)$
return $y^\star$

B.5 Implementation details for using SLP1

In addition to experiments conducted on Devanagari-script data, we include an experimental variant that operates entirely in the SLP1 transliteration space. To enable this, all Sanskrit text in the existing dataset is deterministically converted from Devanagari to SLP1 using the `indic-transliteration` Python library.[7] This conversion is applied uniformly to the training, validation, and evaluation splits to ensure representation consistency.

Following transliteration, we perform instruction fine-tuning of `Phi-4-14B` model using the SLP1-encoded Anuṣṭubh samples. To avoid representation mismatch between instructions and model outputs, the initial system and task prompts are also modified such that all Sanskrit components originally written in Devanagari are transliterated into SLP1; this prevents the model from simultaneously conditioning on mixed-script representations, which could otherwise reintroduce tokenization inconsistencies during instruction following.

At inference time, we apply `Pingala`, our metric-aware constrained decoding method, directly on the SLP1 token stream. Since SLP1 preserves phonological structure in a normalized and tokenizer-friendly form, it integrates seamlessly with the constraint checks required for metrical validation during decoding. Additionally, we enforce two phonological constraints during beam search: first, a consonant cluster constraint that rejects any candidate token producing a sayuktkara (conjunct consonant) with more than three consecutive half-akaras (ardha-akṣaras), ensuring phonologically plausible output; and second, a hard syllable cap that assigns a score of $-\infty$ to any token whose addition would push the total syllable count beyond 32, thereby preventing out-of-bound generation and keeping all beam candidates within the valid anuṣṭubh length.. Apart from the changes defined previously, all other experimental settings

[7] https://github.com/indic-transliteration/indic-transliteration

Model	Generated Sanskrit Poetry
NLLB-dist-1.3B	तस्याहं न सकामां कृत्स् संस्काराणां महायशाः ।दुःखोपजीवितायास्त्वत् कालयोगेन केनचिन् ॥
Phi-4-14B	अक्षमश्च मया कार्षी श्शोकपरा महायशाः । कृतं कर्म महत्कार्यं किं पुनर्जन्म जीवित ।।
Phi-4-14B(SLP1)	na ca tasya mahAtmAnaH kriyatAmitaraM kriyam . mama duzkftakAlasya jIvitA kiM Bavizyati ..

Table 11: Sample of generated Sanskrit poetry for the input "I could not even perform the last rites of that magnanimous one who died of grief on account of me. What use is my life with an unlucky birth?"

i.e. model architecture, optimization parameters, and decoding hyperparameters, are kept identical to the Devanagari-based setup, allowing for controlled comparison.

Note: We employ the SLP1 transliteration scheme only for experiments involving the Phi-4 model. We do not apply SLP1 to NLLB-dist-1.3B, as this model was pre-trained on Sanskrit data in the Devanagari script. For semantic evaluation, we consistently use Devanagari: in our empirical analysis, as seen in Table 14, the cross-encoder fine-tuned on Devanagari exhibits higher correlation with human judgments than its counterpart fine-tuned on SLP1. Accordingly, Devanagari is retained for all semantic similarity experiments. Even for the SLP1 generated poetry, we transliterate it to Devanagari and then evaluate semantic similarity. Section D.2 provides a detailed analysis comparing Devanagari and SLP1 as transliteration schemes for Sanskrit when fine-tuning a cross-encoder for semantic alignment with English inputs.

C Appendix for Cross-Encoder as Semantic Evaluator

C.1 Background: Bi-Encoder semantic similarity and its limitations.

Our initial translation-verification pipeline used a *bi-encoder* architecture, similar to `Chandomitra`- the BGE-M3[8] (Chen et al., 2024) model fine-tuned on `Mitrasaṃgraha` (Nehrdich et al., 2026). In this formulation, the English source e, and a candidate Sanskrit translation s are encoded independently by a shared encoder f_ϕ into fixed-dimensional vectors:

$$v_e = f_\phi(e), \tag{8}$$

$$v_s = f_\phi(s). \tag{9}$$

Semantic similarity is then computed using cosine similarity:

$$\text{sim}(e, s) \;=\; \cos(v_e, v_s) \;=\; \frac{v_e^\top v_s}{\|v_e\|\,\|v_s\|}. \tag{10}$$

The similarity score is calculated using the linearized cosine similarity based semantic similarity metric, similar to `Chandomitra`work:

$$\text{score}_i = \frac{\pi - \cos^{-1}(\cos(\mathsf{emb}(x_i), \mathsf{emb}(\hat{y}_i)))}{\pi}$$

Contrastive objectives (e.g., SimCSE (Gao et al., 2021), InfoNCE (van den Oord et al., 2019), triplet loss (Reimers and Gurevych, 2020)) train bi-encoders to score aligned EnglishSanskrit pairs higher than mismatches. While efficient and scalable, this paradigm depends on *global* sentence embeddings, so the resulting similarity primarily measures coarse semantic proximity rather than the fine-grained token-level correspondences (word order, alignment, syntactic structure) needed for translation fidelity (Adi et al., 2017; Ramos et al., 2025).

Limitations of bi-encoders for translation verification. As reported in Table 2, the average similarity gap between positives and hard negatives saturates at $\approx 22\%$, suggesting an intrinsic representational ceiling imposed by independent encoding and fixed-vector geometry. Table 1 further exposes poor calibration: the bi-encoders mean similarity on ground-truth pairs is 73.46% (human score: 5/5) but only

[8] https://huggingface.co/BAAI/bge-m3

slightly lower (69.05%) on candidate outputs whose human ratings drop to 2.7/5. In short, the bi-encoder assigns similar numeric scores to qualitatively different translations, undermining its reliability for translation verification.

C.2 Rationale –> Gradient-Level Explanation for using Class-Weighted binary cross-entropy

Differentiating the loss with respect to the logit yields:

$$\frac{\partial \mathcal{L}}{\partial g_\theta(x)} = \frac{1}{|\mathcal{D}|} w(y)\big(p_\theta(x) - y\big).$$

For positive examples ($y = 1$), gradient descent increases $g_\theta(x)$ and hence $p_\theta(x)$. For negative examples ($y = 0$), gradient descent decreases $g_\theta(x)$ and hence $p_\theta(x)$. In expectation, this simultaneously raises $\mathbb{E}[p_\theta \mid y = 1]$ and lowers $\mathbb{E}[p_\theta \mid y = 0]$, directly increasing the probability gap.

C.3 Motivation for the Cross-Encoder.

To address the representational and calibration shortcomings of bi-encoders, we adopt a *cross-encoder*, in our case - infoxlm[9] (Chi et al., 2021). A cross-encoder takes the English and Sanskrit texts *together* as input to a single Transformer stack so that full cross-attention between source and candidate is available. Formally, for a pair (e, s) we feed the concatenated sequence (with appropriate separators) to a Transformer and compute a scalar score:

$$\text{score}(e, s) \;=\; g_\psi\big(\,\text{Transformer}\big([\texttt{CLS}\ e\ \texttt{[SEP]}\ s]\big)\,\big), \tag{11}$$

where $g_\psi(\cdot)$ is a small parameterized head (e.g., an MLP over the pooled representation) producing a calibrated likelihood that s is a correct translation of e.

The cross-encoders direct token-level interaction confers several concrete advantages for translation verification:

- **Fine-grained alignment:** cross-attention enables the model to check lexical correspondences, agreement, and argument structure directly. (Reimers and Gurevych, 2019; Ranasinghe et al., 2020)
- **Calibration:** as a discriminative scorer, the cross-encoder produces logits that are better separated between positive and negative pairs, which improves thresholding and ranking. (Rei et al., 2020)
- **Error sensitivity:** local phrase-level errors (e.g., incorrect inflection or missing negation) produce pronounced changes in the cross-encoder score. (Fomicheva et al., 2022)

C.3.1 Why Cross-Encoder Outperforms Bi-Encoder

Unlike the bi-encoder similarity score, $\cos(v_e, v_s)$,, which depends on independent global embeddings, the cross-encoder computes $g_\theta(e, s)$, conditioned on full cross-attention between tokens. This allows the model to evaluate lexical alignment, syntactic agreement, and semantic role consistency directly. The resulting logit distributions exhibit larger separation and lower variance, producing a substantially higher probability gap after sigmoid normalization.

C.4 Rationale on why the spread is high for Cross-encoder Model

Logit Distribution Analysis: Let the class-conditional logits follow:

$$Z_1 \sim \mathcal{N}(\mu_1, \sigma_1^2), \qquad Z_0 \sim \mathcal{N}(\mu_0, \sigma_0^2).$$

Using a logistic-normal approximation, the mean predicted probability is:

$$\mathbb{E}[\sigma(Z)] \approx \sigma\Big(\frac{\mu}{\sqrt{1 + c\sigma^2}}\Big), \quad c = \tfrac{\pi^2}{3}.$$

Thus,

$$G \approx \sigma\Big(\frac{\mu_1}{\sqrt{1 + c\sigma_1^2}}\Big) - \sigma\Big(\frac{\mu_0}{\sqrt{1 + c\sigma_0^2}}\Big).$$

Curriculum hard-negative mining increases $\Delta\mu = \mu_1 - \mu_0$ while reducing σ_0^2, yielding a strong nonlinear amplification of G.

[9] https://huggingface.co/microsoft/infoxlm-large

C.5 Qualitative Analysis of Output

We performed a focused human evaluation on a random sample of $N = 200$ test examples. A single expert rater (a practising poet and *avadhani* in Sanskrit and Telugu who is also completing a technical Masters) scored each candidate translation for *semantic fidelity* on a 1–5 Likert scale (1 = not faithful, 5 = fully faithful). We then report the correlation between the human scores and automatic similarity scores produced by the bi-encoder and cross-encoder verifiers, both before and after isotonic calibration (Sinaga and Nair, 2025).

D Using SLP1 as Transliteration Scheme for Sanskrit

D.1 Instruction Fine-tuning Prompt for SLP1

Phi-4 SLP1 Prompt

The goal is to generate Sanskrit verse that follows the anushtup meter rules for the given input translation.
RULES:
Verse Rules:
The verse contains 32 syllables/akshara and 4 padas in total.
The verse is divided in 2 lines, each containing 16 syllables.
Each line is divided into 2 padas (quartets),each containing exactly 8 syllables.
The fifth syllable of every pada must be LAGHU or short.
The sixth syllable of every pada must be GURU or long.
The seventh syllable of the second and the fourth pada must be HRASVA
The seventh letter of the first and third paada must be DEERGHA

Syllable Rules:
LAGHU vowels: a, i, u, f, x

GURU vowels: A, I, U, F, X, e, E, o, O
HRASVA vowels: a, i, u, f, x
DEERGHA vowels: A, I, U, F, X, e, E, o, O

Syllable is marked laghu, guru and hrasva, deergha based on the vowel it contains.
Syllable containing anusvAra("M") or visarga("H") is always marked as guru.
Syllable that is followed by a conjunct consonant (saMyuktAkzara) is always marked guru.

Figure 4: Phi-4 SLP1 Instruction fine-tuning Prompt

Figure 4 shows the prompt used for instruction fine-tuning for generating Sanskrit Anuṣṭubh poetry in SLP1 transliteration.

D.2 Considering SLP1 for Sanskrit in Cross-Encoder

As described in Section 5.3, Section B.5 and Section 7, adopting SLP1 for transliteration improves our methods metrical conformity. This prompted the question: *How does using SLP1 (rather than Devanagari) affect performance of the cross-encoder?* To answer this, we re-ran the experiment from Section 4.2 on the Sanskrit data after converting it to SLP1. Table 13 reports mean accuracy for the two transliteration schemes; there is no substantial difference in overall accuracy. Table 12 compares the semantic-similarity score distributions: the spread values are similar, although the positive and negative(avg) scores are shifted by ≈5 units in the SLP1 condition relative to Devanagari. Finally, to decide which encoder to use in practice we computed correlations between each models scores and human ratings (Table 14). The cross-encoder trained on Devanagari shows higher correlation with human judgments than the model trained on SLP1. Taken together, these results indicate that while SLP1 yields measurable benefits for metrical conformity during the poetry generation task, it does not improve semantic agreement with human annotations - in our setup slightly degrades; therefore we retain the Devanagari-trained cross-encoder for semantic tasks.

Approach	Positive	Negative(Avg.)	Spread
Devanagri	90.02%	1.67%	88.33%
SLP1	95.81%	6.07%	89.74%

Table 12: Cross-encoder semantic similarity score distribution: Devagari vs SLP1 transliteration scheme

Batch Size	Mean Acc. Devanagari	SLP1
Full	74.74%	77.07%
16	84.71%	78.91%
4	92.87%	90.07%

Table 13: Mean Accuracy: comparison of cross-encoder model trained on Devanagari vs SLP1

Model	Raw		Isotonic Calibrated	
	Pearson	Spearman	Pearson	Spearman
Cross-Encoder	0.6767	0.5833	0.7248	0.7175
Cross-Encoder	0.542	0.598	0.679	0.653

Table 14: Comparison of correlation with human semantic ratings (N=200) between cross-encoder trained using Sanskrit Devanagari vs SLP1 transliteration scheme