Chandomitra: Towards Generating Structured Sanskrit Poetry from Natural Language Inputs

Manoj Balaji Jagadeeshan[*1] Samarth Bhatia[*2] Pretam Ray[†1]
Harshul Raj Surana[†1] Akhil Rajeev P[3] Priya Mishra[4]
Annarao Kulkarni[3] Ganesh Ramakrishnan[4] Prathosh AP[2] Pawan Goyal[1]

[1]*Indian Institute of Technology, Kharagpur*

[2]*Indian Institute of Science, Bangalore*

[3]*Indian Heritage Language Computing Group, Centre for Development of Advanced Computing (C-DAC), Bangalore*

[4]*Indian Institute of Technology, Bombay*

Abstract

Text Generation has achieved remarkable performance using large language models. It has also been recently well-studied that these large language models are capable of creative generation tasks but prominently for high-resource languages. This prompts a fundamental question: *Is there a way to utilize these (large) language models for structured poetry generation in a low-resource language, such as Sanskrit?* We present *Chandomitra* , an English input to structured Sanskrit Poetry translation dataset, specifically adhering to the *Anushtubh* meter. We benchmark various open and closed models, and scrutinize specialized techniques such as constrained decoding and instruction fine-tuning, for the proposed task. Our constrained decoding methodology achieves 99.86% syntactic accuracy in generating metrically valid Sanskrit poetry, outperforming GPT-4o (1-shot: 31.24%). Our best-performing instruction-tuned model, on the other hand, performs better in semantic coherence with the English input, at the expense of slightly lower syntactic accuracy. Human evaluation further reveals that instruction fine-tuned model is better able to capture the poetic aspects. Data[1] and Code[2] are available.

1 Introduction

Language constitutes a key aspect of human intelligence; among its various forms, poetry stands out as a beautiful, creative genre that conveys emotions and ideas with relative brevity. Poetry is universally captivating across diverse nations, ethnicities, and cultures, significantly influencing the evolution of human civilization.

Large language models (LLMs) have made substantial progress in various domains including complex task-solving (Feng et al., 2024), reasoning (Wei et al., 2023). Studies also have shown that large language models such as GPT-3 (Brown et al., 2020), LLaMa (Touvron et al., 2023), etc., have advanced the state of the art in several natural language generation tasks such as text summarization (Zheng et al., 2020) and machine translation (Team et al., 2022). The use of these models for creative generation tasks, such as creative story or poetry generation, is, however, limited to a small number of works (Chakrabarty et al., 2022; Yang et al., 2020).

Most prior research has focused on high-resource languages, including English or Chinese, where the source and target are in the same language. Sanskrit, a classical language (Coulson, 1976), primarily comprises sentences structured as verses in its pre-classic and classic literature. Most existing Sanskrit corpora are composed in verse form (Hellwig, 2010–2021). While there have been works on identification of meter from a Sanskrit poetry (verse) (Terdalkar and Bhattacharya, 2022), there are only couple of works related to Sanskrit poetry (Krishna et al., 2019; Das et al., 2025), that too, given the corresponding Sanskrit prose format, thus tackling it as a reordering task given the meter constraints.

This study presents a dataset *Chandomitra* , derived from the Valmiki Ramayana corpus[3]. In our setting, the source (English) and target (Sanskrit) are in different languages, and we focus on generating Sanskrit verses in Anuṣṭubh Chandas form, which is a particular metrical form of Sanskrit poetry. Thus, this task has a unique setting that requires both translation as well as structured poetry generation.

Despite being provided with metrical constraints and examples in the k-shot prompt (with best performance for $k = 1$ when $k \in [0, 3]$), GPT-4o

---

*Equal contribution as first authors.

†Equal contribution as second authors.

[1]https://huggingface.co/collections/sanganaka/chandomitra

[2]https://github.com/sanganaka-iitkgp/chandomitra

[3]https://www.valmiki.iitk.ac.in

achieves only 31.24% syntactic correctness (52.84% being partially correct) on in-domain samples, and 21.84% on out-of-domain samples. This highlights the challenges of structured poetry generation in Sanskrit, a task that requires strict adherence to rhythm and meter, and motivates the need for more specialized approaches.

We propose multiple possible approaches to address the challenges of structured Sanskrit poetry generation. First, to enforce metrical constraints during inference, we design a constrained decoding strategy utilizing `NLLB-dist-1.3B` that counts syllables and employs precompiled regex according to the Sanskrit `Anuṣṭubh` meter to generate metrically valid outputs with an accuracy of 99.86%. However, this comes at a cost of reduced semantic similarity to the original input sentence.

Next, we enhance the models' adherence to metrical restrictions via instruction fine-tuning (Wei et al., 2022), providing the detailed instructions and constraints in the prompt. We find that the Instruction-finetuned models balance syntax and semantics: `Phi-4-14B` and `Mistral-Nemo-2407-12B` achieve syntactic accuracies of 57.42% and 52.5%, and semantic scores of 67.29 and 68.05.

Our key contributions are as follows:

(a) We introduce *Chandomitra*, a novel translation-cum-poetry generation dataset for structured Sanskrit `Anuṣṭubh` poetry from English inputs.

(b) We propose syntactic and semantic evaluation metrics for the task. For semantic evaluation in this cross-lingual prose-to-poetry setting, we collect human judgments for ~1k English-Sanskrit pairs. We experiment with candidate metrics and select the one with the best correlation, enabling a nuanced assessment of meaning preservation beyond syntax.

(c) We benchmark models including `GPT-4o`, and enhance performance via constrained decoding and instruction fine-tuning. We further perform human evaluation for the best performing `NLLB-distilled-1.3B` and `Mistral-Nemo-2407-12B` models. Human evaluation shows that while `NLLB-dist-1.3B` (constrained decoding) ensured metrical correctness, it led to grammatical discontinuities, lowering ratings. `Mistral-Nemo-2407-12B`, on the other hand, produced semantically and poetically superior verses.

2 Relevant Background

2.1 Sanskrit Poetry

Sanskrit poetry, a cornerstone of Indian literary tradition, is deeply rooted in metrical compositions known as Chandas. The study of Chandas ensures precision in poetic composition and facilitates memorization and recitation.

The term Chandas refers to the metrical framework of Sanskrit poetry. It regulates the arrangement of syllables into specific patterns of light (laghu) and heavy (guru) syllables within a line or stanza. These patterns create rhythm and enhance the aesthetic appeal of the verse. Each metrical unit is called a pāda, and a verse typically comprises four pādas. The science of Chandas is elaborated in ancient texts like `Pingala's Chandaśāstra` (Pingala, 1931).

Constituents of Chandas.

`Pāda`: A quarter of a stanza, each pāda contains a fixed number of syllables based on the meter.

Laghu (Light Syllable): A short syllable, generally with a short vowel and no consonant cluster following it.

Guru (Heavy Syllable): A long syllable, characterized by a long vowel, a consonant cluster, or ending with an `anusvāra/visarga`.

More details can be found in Appendix A.

2.2 Anuṣṭubh Chandas

The `Anuṣṭubh` meter is one of the most prevalent meters in Sanskrit literature, used extensively in texts like the Mahabharata, Ramayana, and Bhagavad Gita. It consists of 32 syllables divided into four pādas, each containing eight syllables[4].

The `Anuṣṭubh` meter follows these precise rules:

- Each pāda consists of exactly eight syllables

- The sixth syllable in every pāda must be guru (heavy)

- The fifth syllable is always laghu (light)

- For the seventh syllable:

  - Must be deergha (long) or guru (heavy) in odd-numbered pādas (1st and 3rd)

  - Must be hrasva (short) in even-numbered pādas (2nd and 4th)

---

[4]The definition provided in sanskrit is available in Appendix subsection B.2

Figure 1: An example of Anuṣṭubh Chanda. Each pada has 8 syllables. □: odd pada; □: even pada.

2.3 Poetry Generation in Other Languages

Early approaches to poetry generation used rule-based, statistical, and RNN-based models (Gonçalo Oliveira, 2012; Jiang and Zhou, 2008; Zhang and Lapata, 2014). Transformer models like GPT show mixed results-while fine-tuned variants of GPT-2 can produce stylistically rich poetry (Bena and Kalita, 2020), zero-shot outputs often lack meter and rhyme (Köbis and Mossink, 2020; Wöckener et al., 2021). To address this, BYGPT5 introduces a token-free architecture tailored to poetic forms, outperforming larger models (Belouadi and Eger, 2023). Fine-tuned GPT-3 can emulate author styles (Sawicki et al., 2023), whereas GPT-3.5 and GPT-4 underperform without fine-tuning. Post-editing methods have also been proposed to improve stylistic fidelity (Ma et al., 2023) While prior work has explored prose-to-poetry transformations in Sanskrit (Krishna et al., 2019), we present the first dataset and model targeting the inverse task: generating structured Sanskrit verse from English prose.

2.4 Problem Definition

Formally, given an input sentence (prose) in English x, we need to generate it's corresponding output (poetry) $\hat{y}$ in Sanskrit s.t. $\hat{y}$ follows the syntactic rules of the Anuṣṭubh chanda as detailed in subsection 2.2 and preserves the meaning of the input x. The ground-truth Sanskrit poetry, if required, is denoted by y.

In other words, this task implies *translation and structured poetry generation at the same time*.

3 The *Chandomitra* Dataset

To facilitate this research, we construct an initial dataset derived from the Valmiki Ramayana corpus[5], retaining only verses in the Anuṣṭubh meter using Chandojñānam[6], a tool that lets us find the meter of Sanskrit poetry. This curated dataset forms the basis for training and evaluating our models. Table 1 gives the statistics of the *Chandomitra* dataset, consisting of 9,727 verses in Anuṣṭubh meter, along with the English translation. One such pair is shown here, with more examples given in Appendix G.

English Translation (Prose):
Effulgent Rama looked at Khara who stood with a mace in hand minus his chariot and said to him first in a gentle voice and then harshly
Ground Truth Sanskrit (Anuṣṭubh Poetry):
खरं तु विरथं रामो गदापाणिमवस्थितम् ।
मृदुपूर्वं महातेजाः परुषं वाक्यमब्रवीत् ॥

Dataset	# Train samples	# Test samples
Chandomitra	8306	1421
OOD	–	520

Table 1: Dataset statistics for *Chandomitra* and out-of-domain (OOD) test set.

In addition, to assess the generalizability of our models, we prepare an out-of-domain dataset. Specifically, we included Raghuvansh by Maharshi Kalidas (468 verses) and Leelavati by Acharya Bhaskaracharya (52 verses). Although these texts belong to different poetic traditions compared to the Valmiki Ramayana corpus, the selected verses are also in the Anuṣṭubh meter, ensuring consistency in the metrical structure across datasets.

The total number of out-of-domain samples amounts to 520 pairs.

4 Proposed Solution

To generate Sanskrit poetry conforming to the strict Anuṣṭubh metrical constraints, we adopt two complementary strategies aligned with the strengths of distinct model families. For raw translation models, inherently sequence-to-sequence and lacking instruction-following abilities, we use constrained decoding to enforce syllabic and structural rules during generation. This guides the de-

---

[5] https://www.valmiki.iitk.ac.in/
[6] https://github.com/hrishikeshrt/chanda/

coder toward metrically valid outputs without altering the architecture. For instruction-following models, we apply instruction-fine-tuning (IFT) to internalize syntactic and semantic patterns of Sanskrit verse via curated prompts and data. These models interpret explicit constraints in natural language, making them apt for controllable generation when fine-tuned on metrically aligned samples. These dual approaches exploit architectural strengths − constrained decoding for symbolic control and IFT for contextual fluency − yielding metrically faithful and semantically coherent verse.

Next, we describe the proposed approaches for structured Sanskrit Anuṣṭubh generation below.

4.1 Constrained Decoding

To ensure that generated Sanskrit poetry adheres to the strict syllabic patterns of metrical form of Anuṣṭubh, we employ a constrained decoding strategy tailored for symbolic control during inference. This approach, detailed in algorithm 1, enables metrically faithful generation without requiring any modification to the underlying language model.

At each generation step, the LM provides probabilities for each token in its vocabulary. To maintain computational efficiency and semantic relevance, we select the top k tokens as a possible next token, and verify which of the resulting syllable weight sequences satisfy our constraints. Precompiled regex filters are used on the syllable weights to enforce the patterns allowed for each pada.

For instance, the filter for the first pada of an Anuṣṭubh verse (8 syllables with the 5th and 6th syllables being laghu and guru, respectively, and the 7th syllable being guru) would be `"^.{4}lgg.$"` (signifying that the first 4 syllable weights are . (unconstrained) and the next three can only be l(laghu), g(guru) and g(guru), in that order, followed by the 8th one being unconstrained again. The `^` and `$` signify the beginning and end of a line. Sequences that do not match the filter are discarded. We then use greedy sampling (we also test with other sampling techniques) to select one from the tokens that satisfy the constraint, pass it to the LLM for next token prediction, repeating until a length constraint is met (such as the 32 syllable rule in Anuṣṭubh).

While the algorithm has a similar form to general constrained decoding algorithms, the implementation details here demand effort to get pre-

cisely right. Particularly, we had to get strict, fine-grained control over the generation and control exactly which tokens can be predicted near pada boundaries, refined syllable-pada division checks, preventing certain duplicate tokens, when to allow modifier-only tokens (such as matras), etc.

Algorithm 1: Constrained Decoding for Sanskrit Poetry Generation

---

Input : LLM, metrical constraints as regular expression filters
Output : Generated Sanskrit poetry adhering to metrical constraints

2 $|V| \leftarrow$ Vocab size of the LLM $k \leftarrow 25$
3 **while** *not end of generation* **do**
4   **Step 1:** Generate token probabilities from the LLM $P \in \mathbb{R}^{|V|} \leftarrow$ **LLM**(input);
5   **Step 2:** Sample top-k possible tokens   Top-k Indices: $I_k \leftarrow$ topk(P, k);
6   **for** *each token at index I_i in top-k token indices I_k* **do**
7   Token $T_i \leftarrow$ vocab$[I_i]$, $O_{\text{temp}} \leftarrow \{...\text{input tokens}\}$;
8   **Step 3:** Temporarily add T_i to output O_{temp} $O_{\text{temp}} \leftarrow O_{\text{temp}} || T_i$;
9   **Step 4:** Calculate the new syllable weighting for O_{temp} $w_{\text{temp}} \leftarrow$ syllable-weights(O_{temp});
10   **Step 5:** Match w_{temp} against pre-compiled regex filters of the same length;
11   **if** *w_{temp} does not match the filters* **then**
12 | **Step 6:** $P[I_i] \leftarrow -\inf$
13   **Step 7:** Check length rules (e.g., end of pada or verse);
14   **if** *no token matches the constraints* **then**
15 | **Step 8:** Increase k and repeat the process from Step 1;
16   **Step 9:** Return modified probabilities for further sampling (greedy/nucleus/etc.)

4.2 Instruction Fine-tuning

Previous works have demonstrated the effectiveness of improving a model's in-context learning abilities through prompts (Zhong et al., 2021; Weir et al., 2022) or by leveraging demonstration examples (Min et al., 2022). Building on this, we perform instruction fine-tuning (IFT) on a set of quantized decoder-only language models using our curated *Chandomitra* training corpus. This includes models from high-performing families such as LLaMA-3, Mistral-Nemo, and Phi-4 (Sreenivas et al., 2024).

Through carefully constructed prompts and fine-tuning data, anchored in metrical and lexical rules, these models learn to respect poetic constraints specified in natural language, making them well-suited for controllable generation.

5 Experiments

5.1 Evaluation metrics

We cannot use standard metrics such as BLEU (Papineni et al., 2002), chrF (Popović, 2015), or ROUGE (Lin, 2004) because these metrics are based on the ordering of the words in the output being the same as the ordering in the ground truth. However, these metrics do not evaluate the task we are trying to solve.

Table 2 shows the results for the above standard metrics on randomly sampled 30 sentences from *Chandomitra* test split. Clearly, the normal translation-based metrics do not do justice to this task.

Type	Approach	bleu	chrF
Llama3.1-8B-Instruct	Instruction Fine-Tuning	0.0	17.75
NLLB-dist-1.3B-FT	Constrained Decoding	0.0	17.09
GPT-4o	1-shot prompt	0.0	19.14

Table 2: Issue with the standard metrics – *Chandomitra* test dataset

Thus, we choose to analyze generated poetry through:

Syntactic metrics: whether the generation follows metrical constraints properly

Semantic metrics: whether the generation conveys the same meaning as the English input

5.1.1 Syntactic Metrics

Full Anuṣṭubh: The percentage of generated samples which completely adheres to the rules of Anuṣṭubh. We use skrutable for this (Tylergneill), which allows us to syllabify text and identify it's meter in a structured manner.

Partial Anuṣṭubh: The percentage of samples that contain the required 32 syllables in length. This is a more relaxed metric and includes samples that follow Anushutbh rules, so this metric will always be ≥ (in magnitude) the Full Anuṣṭubh metric.

5.1.2 Semantic metrics

It is important for the generated poetry to convey the same meaning as the input text. Thus, we need an automated metric to quantify semantic similarity between English (prose) and Sanskrit (poetry). To this end, we compared three methods and chose the best one based on their correlation to human evaluation of semantic similarity.

Human Annotation. We constructed a dataset of 1000 samples taking English input and Sanskrit poetry pairs from the *Chandomitra* dataset as well as generated poetry of various models. Then, we distributed 500 each to two human annotators and asked them to give a score between 1-5 to each (generated) poetry following the scale given in subsection B.3 asking if the original English input intent is preserved.

Semantic metrics comparison. Let x_i be the i-th input sample from the dataset D, and $\hat{y}_i$ be it's corresponding output $\forall i \in \{1, 2, \cdots |D|\}$.

We consider the following possible candidates for semantic similarity computation.

Cross-lingual metrics

A Fine-tuning sentence transformer models and calculating the *linearized* cosine similarity between embeddings $\mathsf{emb}(\cdot)$ of x_i and $\hat{y}_i$. More details pertaining to the fine-tuning of these models are given later.

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

We linearize the cosine similarity to improve metric interpretability and comparability, so that it is easier to compare directly between different models/approaches.

B Using the margin-based scoring method introduced in Artetxe and Schwenk (2019). We calculate the embeddings $\mathsf{emb}(\cdot)$ using a finetuned sentence-transformer model trained in **A**, and use "ratio" as the margin function.

$$\text{score}_i = \frac{\cos(\mathsf{emb}(x_i), \mathsf{emb}(\hat{y}_i))}{\frac{1}{2}(\text{Avg}(\mathsf{emb}(x)) + \text{Avg}(\mathsf{emb}(\hat{y})))}$$

$$\text{where, Avg}(t) = \frac{1}{k} \sum_{z \in \text{NN}_k(t)} \cos(t, z)$$

$\text{Avg}(t)$ is the average cosine similarity of t with its k nearest neighbors (NN_k) in the **other** language.

Monolingual metriccs

C Translating the generated poetry back to English using NLLB-3.3B and calculating the BERTScore between the English input and the English translation of the generated Sanskrit Zhang et al. (2020)

D Using fine-tuned sentence transformer models trained in **A** to calculate the cosine similarity between embeddings $\mathsf{emb}(y_i)$ (the Sanskrit Ground Truth verse) and $\mathsf{emb}(\hat{y}_i)$.

Fine-Tuning Sentence Transformer Models for Embedding

Given the absence of models specifically trained on Sanskrit data, we fine-tune existing models using the mitrasaṃgraha dataset(Nehrdich et al., 2026), followed by assessing the models' translation accuracy on a holdout test set.

The fine-tuning makes use of the multiple negatives ranking loss. The translation accuracy is calculated by `TranslationEvaluator` from the `sentence-transformers` package (Reimers and Gurevych, 2019), which internally checks pairwise if a sample in English has the highest similarity to that same sample in Sanskrit or another Sanskrit sample. This is done in both directions and the average translation accuracy results are presented in Table 3. Based on these results, we choose `BAAI/bge-m3` as the best embedding model for metric choices **A**, **B** and **D**.

Model	# of Params (M)	Mean Acc.†
paraphrase-multilingual-MiniLM-L12-v2	118	85.39
LaBSE	470	88.61
Alibaba-NLP/gte-multilingual-base	305	90.52
BAAI/bge-m3	560	**92.88**
FacebookAI/xlm-roberta-large	560	92.08

Table 3: Evaluation of sentence-transformers models trained on a Sanskrit dataset.
† Mean Acc. refers to the mean translation accuracy (mean of English to Sanskrit, and Sanskrit to English).

Correlation with Human Annotation. We see in Table 4 that linearized cosine similarity based semantic similarity metric (**A**) has a high positive correlation with human evaluation.

For **A**, the linearization does not have an effect on the Spearman correlation as it is based on the relative ranking, which does not change with this operation. However, the Pearson correlation does increase slightly (by 0.02). For further analysis, we had also evaluated other sentence transformer models against human annotations in Appendix E.

We note that the embedding models used in metrics **B** and **D** are identical to those used in **A** (specifically, the fine-tuned `BAAI/bge-m3` model), to ensure consistency across semantic evaluations.

Model	Pearson	Spearman
Cross-lingual		
A Linearized cosine similarity	**0.66**	**0.40**
B kNN ratio-margin scoring	0.21	0.23
Monolingual		
C BERTScore after translation	0.22	0.16
D Sanskrit cosine similarity	0.42	0.25

Table 4: Pearson and Spearman correlations of different semantic similarity metrics and Human Annotation. *Note:* **A** **B** and **D** rely on the same encoder (fine-tuned `BAAI/bge-m3`).

Thus, we choose **A** (the linearized cosine similarity of fine-tuned `BAAI/bge-m3`) to be the representative metric for semantic similarity.

5.2 Models considered

We consider a diverse set of models spanning three broad families to ensure comprehensive coverage and representative evaluation: (a) **Raw Translation Models**: These are encoder-decoder models specialized for translation tasks and typically require fine-tuning. They do not support instruction prompts and are not instruction-following. Models in this category include `IndicBART` (Dabre et al., 2022) and `NLLB200-distilled-1.3B` (Team et al., 2022). (b) **Instruction-Following Models**: These are decoder-only models that have been instruction-tuned and can be prompted with task instructions. We use 4-bit quantized versions for efficiency, and they are also suitable for downstream instruction fine-tuning. This category includes `LLama-3.1-8B`, `LLama3.2-3B` (Grattafiori et al., 2024), `Phi-4-14B` (Abdin et al., 2024), `Mistral-Nemo-2407-12B` (MistralNemo), `Mistral-7B` (Jiang et al., 2023), `Qwen2.5-7B` (Yang et al., 2024; Team, 2024), as well as Indic instruction-tuned models such as `Airavata-7B` (Gala et al., 2024) and `Navarasa-9B`. (c) **Closed-source Models**: We include a commercial model, `GPT-4o` (Mishra et al., 2023), which is accessible only via few-shot prompting and cannot be fine-tuned.

The prompt used for instruction-finetuned models is provided in Appendix B.1. All training and LoRA hyperparameters are detailed in Appendices C and D.

5.3 Results

Table 5 shows that, despite the high zero-shot semantic similarity achieved by the `NLLB-200-dist-1.3B` model, its syntactic performance on Sanskrit Anuṣṭubh verse generation remains poor. Notably, applying constrained decoding to the base model improves syntactic well-formedness to 92.61%, albeit with a marginal degradation in semantic fidelity. When our proposed custom-constrained decoding strategy is applied to the `NLLB` model fine-tuned on *Chandomitra*, syntactic correctness reaches 99.86%, marking a substantial improvement over both the base and simply constrained variants, as well as `indicBART`.

Interestingly, the unmodified base model without any fine-tuning or decoding constraints achieves the highest semantic similarity among all

configurations. This can likely be attributed to the models pretraining objective, which centers on prose-to-prose translation, thereby optimizing for semantic fidelity at the expense of structural constraints inherent to metrical verse.

Our experiments reveal a consistent trade-off between syntactic structure and semantic accuracy: while constrained decoding (with or without fine-tuning) enhances syntactic adherence, it does so by compromising semantic alignment. These findings highlight the fundamental traction between form and meaning in neural generation of metrically bound poetic forms.

Model	CD	FT	Full %	Partial %	Sim
Raw Translation Models					
IndicBART	-	✓	0.0	0.63	52.60
IndicBART	✓	✓	73.75	74.95	54.53
NLLB-dist-1.3B	-	-	0.00	0.92	**79.33**
NLLB-dist-1.3B	✓	-	92.61	93.53	64.96
NLLB-dist-1.3B	-	✓	1.69	20.76	68.33
NLLB-dist-1.3B	✓	✓	**99.86**	99.86	64.91
Instruction Following Models					
Llama3.1-8B	-	-	0.00	0.42	61.88
Llama3.1-8B	-	✓	44.48	82.13	66.45
Llama3.1-8B	✓	✓	40.62	40.62	61.2
LLama3.2-3B	-	✓	34.69	69.31	65.1
LLama3.2-3B	✓	✓	43.75	46.88	61.2
Phi-4-14B	-	✓	**57.42**	75.01	67.29
Mistral-Nemo-2407-12B	-	✓	52.5	80.79	**68.05**
Mistral-7B	-	✓	43.39	66.40	67.53
Qwen2.5-7B	-	✓	34.41	75.58	64.97
Airavata-7B	-	✓	0.0	0.0	49.78
Navarasa-9B	-	✓	2.08	10.55	63.08
Closed-source models					
GPT-4o 1-shot	-	-	31.24	52.84	56.39

Table 5: Evaluation of different models on the *Chandomitra* Test dataset. CD : Constrained Decoding; FT : Fine-Tuning on *Chandomitra* (Train subset). FT in Instruction Following models represents task-specific Instruction Fine-Tuning (IFT).

Instruction-tuned models perform poorly in zero-shot settings but show strong improvements when fine-tuned with Sanskrit-specific prompts. Llama-3.1-8B-Instruct demonstrates this trend, while Phi-4-14B (Abdin et al., 2024) achieves the best performance, with 64.81% syntactic ac-

Model	CD	FT	Full %	Partial %	Sim
GPT-4o (1-shot)	-	-	21.84	37.16	56.57
NLLB-dist-1.3B	✓	✓	**99.62**	99.62	61.97
Mistral-Nemo-2407-12B	-	✓	52.89	77.50	66.08
Phi-4-14B	-	✓	64.81	84.23	**66.55**

Table 6: Out of Dataset Sample Evaluation on various models. CD : Constrained Decoding; FT : Fine-Tuning on *Chandomitra* (Train subset). FT in Instruction Following models represents task-specific Instruction Fine-Tuning (IFT).

curacy and 84.25% of outputs conforming to partial Anuṣṭubh structure, alongside a 66.55 semantic similarity score (Table 5). We also experimented with Indic-LLMs namely Airavata-7B, Navarasa-9B (LLMs finetuned on Indic data) which did not perform well on the task. Interestingly, our analysis reveals a trade-off between syntactic well-formedness and semantic similarity in Sanskrit Anuṣṭubh poetry generation, observable across both IFT and CD approaches.

Effect of Constrained Decoding with Instruction-Finetuned Models The constrained decoding approach, when applied to the LLaMa model (one of the instruct-finetuned model), produced suboptimal results, in part because the model exhibited a propensity to write English text prior to composing Sanskrit poetry. This issue was absent in the NLLB paradigm, which proficiently employs tags for regulated language creation. As a result, efforts to implement custom decoding directly on LLaMa's outputs failed to produce poetry in Anuṣṭubh, despite attempts to limit the production to Devanagari tokens.

Analysis of GPT-4o's Performance on Structured Poetry Generation Our analysis reveals three critical failure modes in commercial models, directly addressing the reviewer's concern regarding their metrical limitations. First, syllable-weight constraint dissociation is evident: while 21.6% of the generated outputs achieved the correct 32-syllable length, they consistently violated the underlying Anuṣṭubh metrical patterns. This demonstrates a failure by large commercial models to internalize the mandatory coupling between syllable quantity and prosodic weight, a fundamental requirement for generating valid Sanskrit poetry. Second, we observed systematic near-miss errors: 27.45% of the outputs deviated from the target length by only 1-2 syllables. Specifically, 328 out of the 390 off-length cases were precisely clustered at 31 or 33 syllables. This clustering indicates that the models exhibit incomplete learning of basic structural constraints rather than generating random errors.

Out of Domain Evaluation. Table 6 shows the inference results of our framework on the OOD dataset, which shows the generalization capability of the models.

Our proposed constrained decoding framework with NLLB outperforms GPT-4o in terms of generating syntactically correct Anuṣṭubh verse by a large margin. Instruction fine-tuned models also per-

Model	Full Anuṣṭubh	32-syllable with Metrical Violations (Partial-Full)	Off-length (30/31/33/34 syllables)	Other Syllable Count Errors (<30 or >34)
GPT-4o	31.24%	21.6%	27.45%	19.7%
Phi-3-14B (Fine-tuned)	57.42%	17.59%	23.93%	1.06%
NLLB-OD (Ours)	99.86%	0%	0.14%	0%

Table 7: Error analysis on Anuṣṭubh generation, illustrating differences in metrical validity and types of verse-length violations across models.

formed well, with `Mistral-Nemo` giving the best semantic similarity. We use these two models to perform human evaluation.

5.4 Human Evaluation

Model	CD	FT	Syntactic Coherence (0-1)	Semantic Coherence (1-5)	Poeticness (1-5)
NLLB-dist-1.3B	✓	✓	0.65	2.125	2.225
Mistral-Nemo-2407-12B	-	✓	0.725	3.775	3.85

Table 8: Human evaluation metrics (avg). The score range for Syntactic Coherence is 0-1 whereas for Semantic Coherence and Poeticness, it is 1-5.

Human evaluation is crucial in this task, as automated metrics alone cannot fully capture the metrical accuracy or poetic expressiveness of Sanskrit verse. To evaluate the quality of generated poetry, we conducted a human evaluation with three expert annotators, all PhD scholars in Sanskrit. Each annotator assessed non-overlapping subset from the sample of 40 random samples from the test set, rating outputs from two models based on syntactic coherence (adherence to Anuṣṭubh meter as well as grammaticality), semantic coherence (faithfulness to the input sentence), and poeticness (aesthetic and literary qualities). We refer readers to the Appendix I for the detailed design, scoring rubric, and instructions provided to annotators.

The instruction-tuned `Mistral-Nemo` model consistently outperforms the constrained NLLB variant in all three aspects. Despite NLLB's, technically high structural compliance with the Anuṣṭubh meter, it lags behind slightly in syntactic coherence due to grammatical incoherence, as judged by human evaluators. This gap highlights a key limitation of hard-constrained decoding, as it enforces formal metrical correctness but often at the expense of fluency and naturalness.

`Mistral-Nemo` surpasses NLLB in semantic coherence, which is consistent with automated evaluation metrics.

`Mistral-Nemo`'s superior performance in poeticness can be explained by the models strong transfer learning capabilities, augmented by fine-tuning on literary-style instruction data and curated prompts that promote stylistic and creative fluency.

5.5 Ablation Analysis

We perform an ablation analysis focusing on various hyper-parameters such as the choice of k as well as sampling strategy in the constrained decoding algorithm.

On the role of k. k represents the number of tokens with the highest probabilities under consideration - in order to check whether they satisfy metrical constraints. We set this to $k = 25$, but also test what happens when we have *too few* or *too many* token options. From Table 13 in Appendix F, we can see that $k = 25$ still performs the best overall.

Different Sampling Strategies. We experiment with different sampling strategies after applying constrained decoding. Table 14 of Appendix F summarizes the results for different sampling strategies such as greedy, multinomial, nucleus, and top-k. We find that greedy decoding performs the best, both structurally and semantically.

Effect of Incorporating Constrained Decoding Algorithm in the Prompt. We perform ablation analysis of incorporating constrained decoding algorithm in the prompt for generating Sanskrit Anuṣṭubh Poetry. We experimented by including algorithm 1 in the prompt provided to GPT-4o (see Figure 2). Table 12 shows that it does not improve performance further and in fact led to a decrease, reaffirming the necessity of our decoding-time constraints.

Latency and Throughput Analysis of CD. We analyze the effect our custom decoding strategy (Constrained Decoding) has on the speed of generation. As expected, there is a decrease caused by *non-parallelized decoding* and the overhead of the `LogitsProcessor` interface with the

transformers library. Notably, 90% of the time is spent inside the skrutable library (Tylergneill), syllabifying the received generated tokens and available options at each step, signalling that the actual algorithm and regex checking/length checking don't add much. To somewhat alleviate this, we do add a Least-Recently-Used (LRU) cache of size 1000 to the syllable scanning, which helps, as shown in Table 9.

Model	CD	Latency (s)	Throughput (tok/s)
NLLB-dist-1.3B	-	0.041	932.8
NLLB-dist-1.3B	✓	0.280	147.6
NLLB-dist-1.3B + cache	✓	0.219	181.4
Llama3.1-8B-Instruct	-	0.936	73.7
Llama3.1-8B-Instruct + cache	✓	2.550	30.0

Table 9: Impact of CD on the Latency and Throughput of generation.

6 Generalization of Constrained Decoding for Poetry Generation

The performance of the constrained decoding approach is tabuled in Table 10. The subsections dive deeper into generalization across Sanskrit meter and other languages.

6.1 Generalization Across Sanskrit Meters

The constrained decoding approach exhibits robust extensibility across diverse Sanskrit meters, facilitated by a modular design encompassing regex compilation and length constraints (algorithm 1, Steps 5 & 7). To substantiate this claim, we evaluated the fine-tuned NLLB model on Triṣṭubh Chandas (a meter comprising four quarters of eleven syllables each). Using a test set of 30 English translations from the Bhagavadgita, the model demonstrated high structural fidelity, achieving 96.67% full compliance and 96.67% partial compliance. The semantic similarity was measured at 68.39. These results confirm the effective generalization of our methodology across

Evaluation	Sanskrit (Anuṣṭubh) [1421]	Sanskrit (Triṣṭubh) [30]	Awadhi (Doha) [30]	Bengali (Payar) [30]
Full Compliance	99.86	96.67	86.67	83.33
Partial Compliance	99.86	96.67	90.0	83.33%
Semantic Similarity	64.91	68.39	-	-

Table 10: Performance of constrained decoding approach for different languages and it's respective chanda(s). [.] denotes the sample size, (.) contains the chanda name.

varied metrical frameworks inherent to Sanskrit.

6.2 Generalization to Other Languages

The proposed constrained decoding approach generalizes beyond Sanskrit to other Indic languages exhibiting analogous prosodic organization. To assess this, we applied the method to Awadhi and Bengali poetic forms. For Awadhi[7], we targeted Doha Chandasdefined by 13 mātrās in the first and third quarters and 11 mātrās in the second and fourth, totaling 48 mātrās. Using the base NLLB-600M model on 30 Doha samples drawn from Rāmacaritamānasa, the system achieved 86.67% full compliance and 90.0% partial compliance. Bengali experiments used Pāyarcomprising two lines of 14 syllables eachwith 30 samples sourced from Śrī Caitanya-caritāmṛta. Under identical modeling conditions, NLLB-600M attained 83.33% full and partial compliance. These findings underscore that the core notion of laghuguru weighting (1 vs 2 mātrās) extends naturally across Indo-Aryan and Dravidian meters, facilitating rapid adaptation to additional regional traditions, including Telugu, Tamil, and Kannada.

7 Conclusion

We introduced *Chandomitra* , a benchmark for generating Sanskrit poetry from English in the Anuṣṭubh meter, and evaluated a range of open and closed models. Constrained decoding and instruction fine-tuning emerged as effective strategies: NLLB-dist-1.3B attains 99.86% metrical accuracy with constrained decoding, while Phi-4-14B achieves 57.42% fully valid verses after instruction tuning. Human judgments confirm that instruction-tuned models better capture poetic form and style, and out-of-domain testing demonstrates that constrained decoding outperforms larger commercial systems such as GPT-4o in enforcing prosodic rules.

Preliminary experiments across other chandas and languages suggest that constrained decoding generalizes reliably beyond Sanskrit and Anuṣṭubh. Overall, our work provides initial baselines for English-to-Sanskrit poetry generation with metrical control and points toward expanding datasets, broader meter coverage, improved semantic metrics, and scaling tuning strategies to multilingual and larger models.

---

[7]https:en.wikipedia.org/wiki/Awadhi_language

8 Limitations

We could not evaluate larger LLMs due to limited availability of compute resources (single GPU); hence, we selected a few representative smaller models for our experiments. Furthermore, while our methods remain meter-agnostic, we have not tested them with other meters. Also, while the proposed method is general, we have not explored generating Sanskrit poetry from inputs in other languages than English.

One limitation of CD is it's application alongside IFT models. This is problematic because the IFT models might generate some generic English/part of the instruction before actually generating the verse. Constrained Decoding skews the output distribution (and increases perplexity) by not allowing this to happen and hence does not perform well. This problem is exacerbated by the BPE (byte pair encoding) of the sentencepiece tokenizer, which breaks some Sanskrit tokens into a pair of two tokens ids. But, CD is limited to only predicting one token from the `LogitsProcessor` interface and hence cannot verify/predict these tokens.

As a preliminary probe of cross-chandas and cross-lingual generalization, we evaluated constrained decoding on a small sample of 30 instances drawn from one representative meter each in Awadhi and Bengali. Given this limited scope, our findings should be interpreted as indicative rather than comprehensive. Future work will expand these experiments with larger datasets and extended meter coverage, and will additionally assess the effects of instruction fine-tuning in these settings.

9 Acknowledgement

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, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

Mikel Artetxe and Holger Schwenk. 2019. Margin-based parallel corpus mining with multilingual sentence embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3197–3203, Florence, Italy. Association for Computational Linguistics.

Jonas Belouadi and Steffen Eger. 2023. ByGPT5: End-to-end style-conditioned poetry generation with token-free language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7364–7381, Toronto, Canada. Association for Computational Linguistics.

Brendan Bena and Jugal Kalita. 2020. Introducing aspects of creativity in automatic poetry generation. *Preprint*, arXiv:2002.02511.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Tuhin Chakrabarty, Vishakh Padmakumar, and He He. 2022. Help me write a poem - instruction tuning as a vehicle for collaborative poetry writing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6848–6863, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

M. Coulson. 1976. *Sanskrit: An Introduction to the Classical Language*. Teach yourself books. Hodder and Stoughton.

Raj Dabre, Himani Shrotriya, Anoop Kunchukuttan, Ratish Puduppully, Mitesh Khapra, and Pratyush Kumar. 2022. Indicbart: A pre-trained model for indic natural language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*. Association for Computational Linguistics.

Kunal Kingkar Das, Manoj Balaji Jagadeeshan, Nallani Chakravartula Sahith, Jivnesh Sandhan, and

Pawan Goyal. 2025. Still not there: Can llms outperform smaller task-specific seq2seq models on the poetry-to-prose conversion task? *Preprint*, arXiv:2511.08145.

Xueyang Feng, Zhi-Yuan Chen, Yujia Qin, Yankai Lin, Xu Chen, Zhiyuan Liu, and Ji-Rong Wen. 2024. Large language model-based human-agent collaboration for complex task solving. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1336–1357, Miami, Florida, USA. Association for Computational Linguistics.

Jay Gala, Thanmay Jayakumar, Jaavid Aktar Husain, Aswanth Kumar M, Mohammed Safi Ur Rahman Khan, Diptesh Kanojia, Ratish Puduppully, Mitesh M. Khapra, Raj Dabre, Rudra Murthy, and Anoop Kunchukuttan. 2024. Airavata: Introducing hindi instruction-tuned llm. *arXiv preprint arXiv: 2401.15006*.

Hugo Gonçalo Oliveira. 2012. Poetryme: a versatile platform for poetry generation. volume 1, article 21.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, and Abhishek Kadian. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Oliver Hellwig. 2010–2021. Dcs - the digital corpus of sanskrit.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Long Jiang and Ming Zhou. 2008. Generating Chinese couplets using a statistical MT approach. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 377–384, Manchester, UK. Coling 2008 Organizing Committee.

Amrith Krishna, Vishnu Sharma, Bishal Santra, Aishik Chakraborty, Pavankumar Satuluri, and Pawan Goyal. 2019. Poetry to prose conversion in Sanskrit as a linearisation task: A case for low-resource languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1160–1166, Florence, Italy. Association for Computational Linguistics.

Nils Köbis and Luca Mossink. 2020. Artificial intelligence versus maya angelou: Experimental evidence that people cannot differentiate ai-generated from human-written poetry. *Preprint*, arXiv:2005.09980.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jingkun Ma, Runzhe Zhan, and Derek F. Wong. 2023. Yu sheng: Human-in-loop classical Chinese poetry generation system. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 57–66, Dubrovnik, Croatia. Association for Computational Linguistics.

Keshav S Melnad, Pawan Goyal, and Peter Scharf. 2013. Meter identification of sanskrit verse. *The Sanskrit Library, USA*.

So Yeon Min, Devendra Singh Chaplot, Pradeep Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov. 2022. Film: Following instructions in language with modular methods. *Preprint*, arXiv:2110.07342.

Mayank Mishra, Prince Kumar, Riyaz Bhat, Rudra Murthy V, Danish Contractor, and Srikanth Tamilselvam. 2023. Prompting with pseudo-code instructions. *Preprint*, arXiv:2305.11790.

Team MistralNemo. Mistral nemo.

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.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311318, USA. Association for Computational Linguistics.

Pingala. 1931. *Chhanda Stra*. Original Sanskrit text. Retrieved from Internet Archive.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. 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*. Association for Computational Linguistics.

Piotr Sawicki, Marek Grzes, Luis Fabricio Góes, Dan Brown, Max Peeperkorn, Aisha Khatun, and Simona Paraskevopoulou. 2023. On the Power of Special-purpose GPT Models to Create and Evaluate New Poetry in Old Styles.

Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Gerald Shen, Jiaqi Zeng, Zijia Chen, Yoshi Suhara, Shizhe Diao, Chenhan Yu, Wei-Chun Chen, Hayley Ross, Oluwatobi Olabiyi, Ashwath Aithal, Oleksii Kuchaiev, Daniel Korzekwa, Pavlo Molchanov, Mostofa Patwary, Mohammad

Shoeybi, Jan Kautz, and Bryan Catanzaro. 2024. Llm pruning and distillation in practice: The minitron approach. *Preprint*, arXiv:2408.11796.

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, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation. *Preprint*, arXiv:2207.04672.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Hrishikesh Terdalkar and Arnab Bhattacharya. 2022. Chandojnanam: A sanskrit meter identification and utilization system. *Preprint*, arXiv:2209.14924.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Tylergneill. Github - tylergneill/skrutable: Toolkit for manipulating sanskrit text with python.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. *Preprint*, arXiv:2109.01652.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Nathaniel Weir, Xingdi Yuan, Marc-Alexandre Côté, Matthew Hausknecht, Romain Laroche, Ida Momennejad, Harm Van Seijen, and Benjamin Van Durme. 2022. One-shot learning from a demonstration with hierarchical latent language. *Preprint*, arXiv:2203.04806.

Jörg Wöckener, Thomas Haider, Tristan Miller, TheKhang Nguyen, Thanh Tung Linh Nguyen, Minh Vu Pham, Jonas Belouadi, and Steffen Eger. 2021. End-to-end style-conditioned poetry generation: What does it take to learn from examples alone? In *Proceedings of the 5th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 57–66, Punta Cana, Dominican Republic (online). Association for Computational Linguistics.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Xiaopeng Yang, Xiaowen Lin, Shunda Suo, and Ming Li. 2020. Generating thematic chinese poetry using conditional variational autoencoders with hybrid decoders. *Preprint*, arXiv:1711.07632.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. *Preprint*, arXiv:1904.09675.

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.

Changmeng Zheng, Yi Cai, Guanjie Zhang, and Qing Li. 2020. Controllable abstractive sentence summarization with guiding entities. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5668–5678, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. 2021. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2856–2878, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Appendix

A Extended background on Chandas

Meters in Sanskrit are broadly classified into:

1. Varṇavṛtta: Based on a fixed number and sequence of syllables (laghu and guru).

2. Mātrāvtta: Based on the count of mātrās (morae or units of time), regardless of the syllable count.

3. Samavṛtta, Ardhasamavṛtta, Viṣamavṛtta: Subcategories based on uniformity or variation across the pādas.

The work by Melnad et al. (2013) explains the meters in detail. Anuṣṭubh is an example of Samavṛtta meter.

The patterns for the meters is specified through the use of Gaṇas. Gaṇas denote patterns of three syllables, combining light and heavy syllables. There are eight types of gaṇas, represented mnemonically as:

- ya: laghu-guru-guru
- ra: guru-laghu-guru
- ta: guru-guru-laghu
- bha: guru-laghu-laghu

- ja: laghu-guru-laghu
- sa: laghu-laghu-guru
- ma: guru-guru-guru
- na: laghu-laghu-laghu

B Additonal Details

B.1 Sanskrit Poetry Generation 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 2: Prompt used for Instruction fine-tuning, providing necessary grammatical and Anuṣṭubh meter rules

Please refer to Figure 2 for the prompt used for zero-shot / few-shot and instruction fine-tuning.

B.2 Definition of Anuṣṭubh in original Sanskrit

This metrical pattern is traditionally encoded in the following Sanskrit mnemonic verse:

श्लोके षष्ठम् गुरु ज्ञेयं सर्वत्र लघु पञ्चमम् ।
द्विचतुष्पादयोः ह्रस्वं सप्तमं दीर्घमन्ययोः ॥
[8]

Translation:
"The fifth syllable is always light;
The seventh is light in the second and fourth lines;
The sixth is always heavy this defines the characteristics of the śloka."

B.3 Human Annotation Scoring for Metric Evaluation

Refer to subsubsection 5.1.2

Score Metric:
5: Meaning fully preserved with nuanced fidelity
4: Mostly preserved; minor omissions or shifts
3: General sense preserved, but key elements missing
2: Only a vague resemblance to input meaning
1: Completely unrelated

B.4 Constrained Decoding Algorithm

Algorithm 2: Constrained Decoding for Sanskrit Poetry Generation

---

   Input : Pre-trained LLM, input context, metrical constraints (e.g., rules) in the form of pre-compiled regular expression filters

   Output : Generated Sanskrit poetry adhering to metrical constraints

1 $|V| \leftarrow$ Vocab size of the LLM $k \leftarrow 25$

2 **while** *not end of generation* **do**

3   **Step 1:** Generate token probabilities from the LLM $P \in \mathbb{R}^{|V|} \leftarrow$ **LLM**(input);

4   **Step 2:** Sample top-k possible tokens   Top-k Indices: $I_k \leftarrow$ topk(P, k);

5   **for** *each token at index I_i in top-k token indices I_k* **do**

6   Token $T_i \leftarrow$ vocab$[I_i]$, $O_{\text{temp}} \leftarrow \{...\text{input tokens}\}$;

7   **Step 3:** Temporarily add T_i to output O_{temp} $O_{\text{temp}} \leftarrow O_{\text{temp}} || T_i$;

8   **Step 4:** Calculate the new syllable weighting for O_{temp}
        $w_{\text{temp}} \leftarrow$ syllable-weights(O_{temp});

9   **Step 5:** Match w_{temp} against pre-compiled regex filters of the same length;

10   **if** *w_{temp} does not match the filters* **then**

11   **Step 6:** $P[I_i] \leftarrow -\inf$

12   **Step 7:** Check length rules (e.g., end of pada or verse);

13   **if** *no token matches the constraints* **then**

14   **Step 8:** Increase k and repeat the process from Step 1;

15   **Step 9:** Return modified probabilities for further sampling (greedy/nucleus/etc.)

Refer to algorithm 2

---

[8]Another definition of Anuṣṭubh:
पञ्चमं लघु सर्वत्र सप्तमं द्विचतुर्थयोः ।
गुरु षष्ठं च सर्वेषाम् एतच्छ्लोकस्य लक्षणम् ॥

C Training Hyperparameters for Instruction-Finetuned Models

- **Maximum Sequence Length:** 1024

- **Warmup Steps:** 5

- **Learning Rate:** 2×10^{-4}

- **Optimizer:** adamw_8bit

- **Weight Decay:** 0.01

- **Learning Rate Scheduler Type:** linear

D LoRA Hyperparameters for Instruction-Tuned Models

- $r = 16$

- **Target Modules:** ["q_proj", "k_proj", "v_proj", "o_proj", "gate_proj", "up_proj", "down_proj"]

- **LoRA Alpha:** 16

- **LoRA Dropout:** 0

- **4-Bit Quantization:** *If used*

E Extended evaluation of A on Sentence Transformer Models

Refer to Table 11.

Model	Pearson	Spearman
paraphrase-multilingual-MiniLM-L12-v2	0.63	0.38
LaBSE	0.64	0.38
Alibaba-NLP/gte-multilingual-base	0.65	0.38
BAAI/bge-m3	**0.66**	**0.40**
FacebookAI/xlm-roberta-large	0.64	0.38

Table 11: Pearson and Spearman correlations of **Sentence Transformer Models** for metric **A**(linearized cosine similarity) (subsubsection 5.1.2) and Human Evaluation.

F Ablation: Constrained Decoding

F.1 Effect of pseudo-code in prompt

Model	Full %	Partial %
GPT-4o (1-shot)	**31.24**	**52.84**
GPT-4o (1-shot + CD)	26.46	48.35

Table 12: Evaluation of GPT-4o on the Test dataset. CD : Constrained Decoding Algorithm as pseudo-code in prompt.

k	Full %	Partial %	Sim	Latency (s)	Throughput (tok/s)
10	99.30	99.93	64.91	0.160	248.3
25	99.86	99.86	64.91	0.219	181.4
50	99.93	99.93	64.92	0.413	96.1
100	99.93	99.93	64.92	0.741	53.604

Table 13: Effect of increasing and decreasing k on constrained decoding. The model used was NLLB-dist-1.3B-FT. The metrics mean the same as in Table 5

F.2 Effects of k

We test what happens when we have *too few* or *too many* token options in constrained decoding. From Table 13, we can see that k = 25 still performs the best overall.

- **Structurally**, a moderate value of k gives the best results. This is because when k is low, there may not be enough options to satisfy metrical criteria.

- **Semantically**, it is intuitively best to choose the highest probability option - which may not be chosen when k is high, because of either constraints not being met, or sampling. The metrics show a clear trend of decreasing similarity with increasing k.

F.3 Sampling Strategies

Sampling strategy	Full %	Partial %[2]	Sim[3]
Greedy	**99.86**	99.86	**64.91**
Multinomial (Temperature $T = 0.7$)	72.56	97.19	64.03
Nucleus/Top-p ($p = 0.9$)	62.35	95.07	63.41
Top-k ($k = 10$)	79.80	97.33	63.42
Contrastive Search ($\alpha = 0.6, k = 4$)	92.12	99.16	64.25

Table 14: Effect of changing the sampling strategy on constrained decoding. The model used was NLLB-dist-1.3B-FT. The metrics mean the same as in Table 5

As mentioned in subsection 5.5, we explore the effects choosing a different sampling strategy has, after applying constrained decoding. These are shown in Table 14.

G Dataset Examples

We present some extracts from *Chandomitra* in Table 15.

H Examples of Input-Output pairs from Different Models

For the following input, model generated Sanskrit poetry is shown in Table 16.
Input:

Effulgent Rama looked at Khara who stood with a mace in hand minus his chariot and said to him first in a gentle voice and then harshly

I Human Evaluation

We conducted a structured human evaluation study to complement our automated metrics and assess the linguistic and poetic quality of the generated Sanskrit verses. Three expert annotators, each holding a PhD in Sanskrit, were engaged to assess the outputs of two models (NLLB-dist-1.3B-FT+CD and

Input (English Prose)	Output (Sanskrit Poetry)
Effulgent Rama looked at Khara who stood with a mace in hand minus his chariot and said to him first in a gentle voice and then harshly	खरं तु विरथं रामो गदापाणिमवस्थितम् । मृदुपूर्वं महातेजाः परुषं वाक्यमब्रवीत् ॥
Having witnessed the destruction of those regions earned by him through asceticism, the son of Jamadagni left for the best of mountains Mahendra.	स हतान् दृश्य रामेण स्वांल्लोकांस्तपसार्जितान् । जामदग्न्यो जगामाशु महेन्द्रं पर्वतोत्तमम् ॥
Again, and again looking back in fear, covertly, bewildered, crushing one another, they went into the Lanka city.	अन्योन्यं प्रममन्थुस्ते विविशुर्नगरंभयात् । पृष्ठतस्ते तु सम्मूढाः प्रेक्षमाणामुहुर्मुहुः ॥

Table 15: Samples from *Chandomitra* dataset

Model	Generated Sanskrit poetry
`Mistral-Nemo-2407-12B`	तं गदापाणिनं दृष्ट्वा विहीनं रथसत्तमम् । प्रोवाच प्रथमं रामो मृदुं वाक्यं ततस्ततः ॥
`NLLB-dist-1.3B` (CD)	रथात्क्षिप्रं तु रामो रज्ज्वलन् खलु समीरितम् । उवाच भार्गवं रामो ध्यायन् स्त्रियमिवाचले ॥
`GPT-4o` (1-shot)	रथहीनं खरं दृष्ट्वा रामो मृद्वा जगाद तम् । पश्चात् तं कुपितो वाक्यं उवाच तेजसा युतः ॥

Table 16: Sample output from different models for the input, "Effulgent Rama looked at Khara who stood with a mace in hand minus his chariot and said to him first in a gentle voice and then harshly".

`Mistral-Nemo-2407-12B`) across 40 samples. The evaluation dataset was allocated to the annotators in a non-overlapping manner, with sample sizes of 15, 15, and 10, respectively..

Each evaluation sample consisted of:

- The input English sentence.

- The ground truth Sanskrit verse composed in Chandas

- The verses generated by `NLLB-dist-1.3B-FT+CD` and `Mistral-Nemo-2407-12B`.

Annotators were provided with detailed instructions outlining the evaluation criteria, along with a few illustrative examples to calibrate their assessments. They were asked to score each models output independently on the following three dimensions using a 0-1 for Syntactic Coherence and 1-5 Likert scale (1 being poor and 5 being excellent) for rest of the metrics:

Syntactic Coherence: Whether the generated verse conforms to the structural constraints of Chandas, including syllabic count and metrical pattern.

Semantic Coherence: The degree to which the generated verse semantically aligns with the input English sentence.

Poeticness: The extent to which the output demonstrates poetic features such as use of rasa (aesthetic flavor), `alankāra` (figures of speech), rhyming or alliteration, and elevated or ornate vocabulary.

The annotators were also informed about the intended use of their evaluations in the analysis section of this paper. Inter-annotator agreement is not applicable here due to the disjoint set of samples assigned to each annotator. Nonetheless, given the annotators' domain expertise and the clear evaluation rubric, the scores provide meaningful insights into the relative strengths and weaknesses of the two models in terms of generating metrically accurate, semantically faithful, and aesthetically compelling Sanskrit poetry.