An introduction to computational identification and classification of *Upamā alaṅkāra*

Bhakti Jadhav, Himanshu Dutta, Shruti Kanitkar
Malhar Kulkarni, Pushpak Bhattacharyya
Indian Institute of Technology Bombay, India
{bhakti.sj, shruti.kanitkar}@iitb.ac.in
{himanshud, pb}@cse.iitb.ac.in malhar@hss.iitb.ac.in

Abstract

Upamā alaṅkāra, a prominent figure of speech in Sanskrit literature, comprises of four components: *Upamāna* (standard of comparison), *Upameya* (object of comparison), *Sādhāraṇadharma* (shared attribute), and *Upamādyotaka* (comparator). It is broadly classified into *Pūrṇopamā* (complete simile) and *Luptopamā* (elliptical simile), with the former containing all four components and the latter omitting one or more. This paper employs large language models (LLMs), specifically Llama-3.1 7B, with a prompt-based strategy to classify instances as *Pūrṇopamā*, *Luptopamā*, or none, and to extract the components for *Pūrṇopamā* cases. Using datasets from the *Rāmāyaṇa* and *Raghuvaṃśa*, the approach demonstrates promising results for both classification and component extraction tasks, showcasing the potential of LLMs in computational philology and Sanskrit literary analysis. [1]

1 Introduction

Amongst the various disciplines of knowledge coded in the Sanskrit language, *Kāvyaśāstra* is an important study discipline that encompasses systematic analysis of various elements of poetry such as *alaṅkāra*, *rasa*, *rīti*, *guṇa*, etc. There has been significant in-depth analysis of the element *alaṅkāra*, which is translated as "figures of speech". These *alaṅkāra*-s are broadly classified into the following three sub-types: *Śabdālaṅkāra*- Figures of sound; *Arthālaṅkāra* -Figures of meaning; and *Ubhayālaṅkāra* - Figures of both sounds and meaning.

One of the most prominent *arthālaṅkāra*-s is *Upamā alaṅkāra*, similar to Simile in the English language, which is extensively employed across diverse domains in Sanskrit literature. This usage primarily seems to be either for embellishment, as in the case of *kāvya*-s, or to explicate a point using examples, as in *śāstra*-s.

However, the interpretation of *Upamā alaṅkāra* is found to be quite a complicated task. While rhetorical texts provide clear theoretical definitions, their practical application in literature often exhibits significant complexity. This complexity arises due to Sanskrit's morphological richness, free word order, employment of multiple adjectives, etc. This variability increases the cognitive load for learners and readers and complicates computational analysis, making it imperative to develop systems that can address these challenges related to interpreting *Upamā alaṅkāra* effectively.

The key steps in the analysis of *Upamā alaṅkāra* are:

1. Creation of construe (*anvaya*) to derive meaning.
2. *Upamā alaṅkāra* component identification.
3. Identification of sub-type of *Upamā alaṅkāra*: *Pūrṇopamā* or *Luptopamā*.
4. Identification of sub-type of *Pūrṇopamā* and identification of sub-type of *Luptopamā*

[1]Link to code and data: https://github.com/himanshu-dutta/sanskrit-upma-alankar-analysis.git

Currently, analyzing such texts requires manual efforts, where domain experts are required to invest considerable time and effort to study the literature which employs *Upamā alaṅkāra*. The poetry/prose belonging to the *Pūrṇopamā* subclass has all four components lexically present, making analysis of such texts through computational methodologies feasible. Further, employing computational methodologies such as Large Language Models (LLMs) allows us to scale analysis of *Upamā alaṅkāra* to the vast and rich Sanskrit literature. This serves as a motivation for us to undertake a study which employs LLMs for the different tasks related to the analysis of *Upamā alaṅkāra* in Sanskrit literature. More specifically, we identify two key tasks in the process of analysis of *Upamā alaṅkāra*. We focus on:

- *Upamā* Subclass Classification: Distinguishing between *Pūrṇopamā*, *Luptopamā* or *None*.
- *Pūrṇopamā* Component Identification: Identifying the components of texts belonging to *Pūrṇopamā* subclass.

By automating these processes, we devise an approach that not only enhances a reader's ability to comprehend Sanskrit literary texts but also serves as a valuable tool for independent learning and research. Furthermore, this work contributes to the broader Sanskrit Computational Linguistics (SCL) field by providing scalable methodologies for analyzing literary constructs.

Our main contributions are:

- We present the first computational study, to the best of our knowledge, on the identification and classification of Upamā alaṅkāra in Sanskrit literature. Our approach utilizes an LLM-based pipeline to address both tasks, as detailed in Section 3.
- We introduce a curated evaluation dataset comprising of poetic excerpts from the *Rāmāyaṇa* and *Raghuvaṃśa* (Kale (1957), Nandargikar (1957)). The dataset is specifically designed for subtype classification of *Upamā alaṅkāra* and component identification in *Pūrṇopamā alaṅkāra*, consisting of **128** examples annotated by domain experts.
- We evaluate state-of-the-art LLMs on these tasks, highlighting their capabilities and limitations. Our work establishes a baseline for future research in computational analysis of *Upamā alaṅkāra*.

2 Background and Literature Survey

2.1 *Upamā alaṅkāra*

Upamā alaṅkāra, one of the oldest and most prominent figures of speech in Sanskrit, traces its earliest mentions in the *Ṛgveda*[2]. Its systematic study as a literary embellishment begins with the *Nāṭyaśāstra* and is further developed by classical rhetoricians such as Bhāmaha, Daṇḍin, Udbhaṭa, Rudraṭa, Mammaṭa, Panditarāja Jagannātha, etc. In this figure of speech, the central idea is to compare one object with another due to both possessing a common characteristic. Herein, a comparison takes place by measuring the object of comparison closely with the standard of comparison. Such a comparison serves various purposes such as familiarizing with the unknown entity, giving a nuanced explanation, narrating a situation as is, appreciation or degradation of an object, etc. For instance,

1. *sa devyā vyavasāyaṃ ca ghoraṃ ca śapathaṃ kṛtam/dhyātvā rāmeti niśśvasya chinnastarurivāpatat//* [3]

Translation- Reflecting on the determination of the queen and her dreadful vow, the king sighed and cried, 'O Rama' and then fell down like a tree severed. [4]

[2]By conservative estimate the time period of the oldest text in Sanskrit i.e., Ṛgveda is considered as 1000 B.C.

[3]Valmīkīyarāmāyaṇa (2025) 2.12.54

[4]https://www.valmiki.iitk.ac.in/content?language=dv&field_kanda_tid=2&field_sarga_value=12&field_sloka_value=54

2. *sa rājyaṃ guruṇā dattaṃ pratipadyādhikaṃ babhau/ dinānte nihitaṃ tejaḥ savitreva hutāśanaḥ//* [5]
Translation- On receiving kingship from his father Raghu shone more brightly than before like Firegod who shines forth brightly when the Sun invests his radiance in him at the close of a day. [6] The structure of *Upamā alaṅkāra* comprises four components:

1. ***Upameya*** – The object of comparison or topic.

2. ***Upamāna*** – The standard of comparison or vehicle.

3. ***Sādhāraṇadharma*** – The common property that is the basis for the comparison. Also known as event or state.

4. ***Upamādyotaka*** – The word or marker that indicates the similarity or comparator.

While this figure of speech is expressed in multiple ways in various *Alaṅkāraśastra* texts, we rely on the classification pattern proposed by Mammaṭa in *Kāvyaprakāśa.* We chose this pattern of classification because it is in accordance with the theory of grammarians. The classification is straightforward as it is done on morpho-semantic grounds. This classification divides *Upamā* into two primary sub-types:

1. **Pūrṇā (Complete)** – The type which consists of all four components of *Upamā* mentioned explicitly.

2. ***Luptā*** **(Elliptical)** – The type which omits explicit mention of one, two or three components of *Upamā*.

The computational analysis of *Upamā alaṅkāra* is particularly challenging due to:

- Component Identification: Extracting the four components (*Upamāna, Upameya, Sādhāraṇadharma, and Upamādyotaka*) from sentences, especially in cases where components are implied or omitted.

- Subtype Classification: Distinguishing between *Pūrṇopamā* and *Luptopamā* based on the presence or absence of components.

- Contextual Dependencies: Understanding cultural and contextual nuances that influence the interpretation of the comparison.

- Grammatical Ambiguities: Disambiguating between Upamāna and Upameya, which often share the same grammatical case in Sanskrit's free word order.

- These challenges necessitate a computational framework that integrates syntactic, semantic, and contextual analysis to accurately process Sanskrit texts consisting of *Upamā alaṅkāra.*

2.2 Computational Background

In recent years, advancements in computational methods have significantly impacted Sanskrit language analysis, seamlessly blending traditional linguistic frameworks with modern Natural Language Processing (NLP) techniques. Early computational efforts, such as the Sanskrit Heritage Platform (Huet, 2005), utilized rule-based systems inspired by Pāṇinian grammar for morphological analysis and syntactic parsing.

The introduction of Transformer models (Vaswani, 2017) marked a paradigm shift in NLP by enabling the modeling of complex contextual relationships within the text through self-attention

[5]Raghuvaṃśa 4.1
[6]https://sanskritdocuments.org/sites/giirvaani/giirvaani/rv/sargas/04_rv.htm

mechanisms. Transformers are deep learning architectures that use a mechanism called self-attention to weigh the importance of different words in a sentence, allowing for nuanced understanding of context, even across long text sequences. This capability makes them particularly effective for linguistic analysis.

Large Language Models (LLMs), built upon Transformers, are pre-trained on massive amounts of textual data from various sources, enabling them to generate human-like text and perform a wide range of language-related tasks. Their ability to understand and produce text makes them valuable for analyzing complex linguistic phenomena, including those in classical languages like Sanskrit.

LLM Prompting (Brown et al. (2020), Sahoo et al. (2024)) is a technique for guiding LLMs to perform specific tasks by crafting input queries or statements that define the context and objectives. For example, a prompt can specify that the model should complete a translation, summarize a text, or answer a question. Few-shot prompting builds on this by including a small number of task-specific examples within the prompt itself, demonstrating the desired output format. This approach is particularly useful for linguists who may lack extensive labeled datasets, as it leverages the model's ability to generalize from minimal input.

These innovations have begun to influence Sanskrit computational linguistics, opening new avenues for integrating traditional linguistic principles with state-of-the-art machine learning techniques.

2.3 Literature Survey

Linguistic Studies on Upamā Alaṅkāra: *Upamā alaṅkāra* has been a subject of discussion in Sanskrit rhetorics from around 1st century until the present 21st century. The rhetoricians have put forth their perspective on the perception of this particular figure of speech. Paṇḍitarāja Jagannatha propounds a scholastic exposition of this figure of speech by discussing the verbal understanding that is the result of various linguistic formations expressing *Upamā*. Modern research focuses on the aesthetic analysis of Upamā as in Parik (2020). There are researches that highlight comparative analysis of proposition of Upamā of various rhetoricians as in Tiwari(2023), Joshi (2015). Traditional studies on *Upamā alaṅkāra* have focused on its role as an aesthetic and rhetorical device.

Computational Approaches to Sanskrit Alaṅkāra Analysis: The complexities of Sanskrit, including sandhi (euphonic combination) and compound formations, have driven specialized computational approaches.

Efforts to integrate syntactic and semantic features into Sanskrit analysis include early works like (Goyal et al., 2009), which combined rule-based and semantic-driven parsing, and (Kulkarni and Das, 2012), introducing a discourse analysis framework to understand contextual dependencies. (Goyal and Huet, 2013) further examined completeness in a Sanskrit reader for holistic text comprehension.

Recent studies merge traditional linguistic frameworks with neural models. (Jadhav and Kulkarni, 2024) utilized Immediate Constituent analysis for *Upamā alaṇkāra* in lexicography. (Chaudhari et al., 2024) fine-tuned a generative pre-trained model for simile element extraction, identifying *Upameya*, *Upamāna*, and *Upamādyotaka*. Similarly, (Jadhav et al., 2023) employed dependency tree structures to analyze *Upamāna*, *Upameya*, *Upamādyotaka*, and *Sādhāraṇadharma* in examples from *Kāvyaprakāṣa*.

Research on Sanskrit figures of sound, such as *Anuprāsa* and *Yamaka alaṇkāra*, includes tools by (Barbadikar and Kulkarni, 2024) and (Barbadikar and Kulkarni, 2023) for their identification and classification. To the best of our knowledge no such tools exist for *Upamā alaṇkāra*.

While figurative language analysis is extensive in high-resource languages like English (Chakrabarty et al. (2022), Shutova (2011), Lai and Nissim (2024), Qadir et al. (2016), He et al. (2022), Liu et al. (2018)), comprehensive studies on simile subtype or component identification in Sanskrit remain limited.

Current computational research on Sanskrit *alaṇkāra*-s faces challenges due to the lack of annotated corpora and standardized tools. However, leveraging LLMs presents a promising direction, as highlighted in the International Sanskrit Computational Linguistics Symposia (Bhattacharya, 2024).

3 Methodology

In this section, we present our computational approach to identifying and analyzing *Upamā alaṅkāra* in Sanskrit sentences. Our methodology is divided into two main phases: (1) Classification of *Upamā alaṅkāra* subtypes, and (2) Identification of components for *Pūrṇopamā*. Figure 1 illustrates the overall pipeline of our approach. The input to our system, sentence S, is considered to be any poetry/prose in Sanskrit literature.

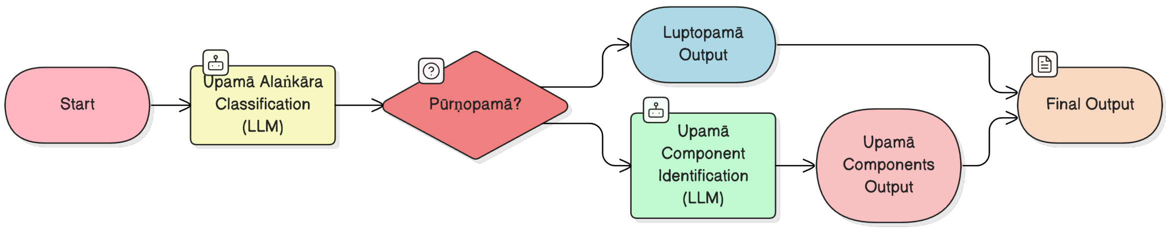

Figure 1: Pipeline Overview for *Upamā alaṅkāra* Extraction.

3.1 Phase 1: Upamā Alaṅkāra Classification

The first phase involves classifying the input sentence as *Pūrṇopamā*, *Luptopamā*, or *None*. We utilize a few-shot prompting technique on an instruction-tuned LLM for this stage. The prompt is constructed based on the description of *Upamā alaṅkāra* from *Kāvyaprakāśa*.

3.1.1 Algorithm for Classification

The classification process utilizes a systematic algorithm to determine the type of *Upamā alaṅkāra*. A custom prompt is constructed to include definitions and annotated examples of *Pūrṇopamā* and *Luptopamā*. This prompt is input into the LLM alongside the source Sanskrit sentence S. The algorithm evaluates the presence of the four key components—*Upameya*, *Upamāna*, *Sādhāraṇadharma*, and *Upamādyotaka*—to make a classification. Sentences containing all four components are labeled as *Pūrṇopamā*, while those missing at least one component (other than *Upameya*) are categorized as *Luptopamā*. Sentences that fail to meet these criteria are classified as None. The classification process follows these steps:

Algorithm 1 Upamā Alaṅkāra Classification Algorithm

Require: Sanskrit sentence S in romanized form.
Ensure: Classification result: *Pūrṇopamā*, *Luptopamā*, or None.

Construct a prompt with:
- Definitions of *Pūrṇopamā* and *Luptopamā*.
- Annotated examples.

Input S into the LLM using the constructed prompt.
Extract the output classification label.
if all four components (*Upameya*, *Upamāna*, *Sādhāraṇadharma*, *Upamādyotaka*) are present **then**
    return *Pūrṇopamā*.
else if at least one component is missing (excluding *Upameya*) **then**
    return *Luptopamā*.
else
    return None.
end if

3.1.2 Prompt Template for Classification

We use the following prompt template to prepare the input for the LLMs with the input sentence for the classification task:

You are a highly knowledgeable language model specializing in classical Sanskrit poetics. Your task is to classify a given prose passage in Sanskrit (Romanized) into one of four categories based on the presence of the figure of speech called Upamā alaṅkāra.

Explanation of Upamā alaṅkāra:
Upamā alaṅkāra (simile) is a poetic device where a comparison is drawn between two entities. The essential elements of Upamā alaṅkāra are:
- Upameya (Object of comparison): The entity being described.
- Upamāna (Standard of comparison): The entity being compared to.
- Sādhāraṇadharma (Common property/State/Event): The common quality between the two. This common quality can be an object or an action. A noun or verb can denote this event in a verse or sentence.
- Upamādyotaka (Comparator): Words like iva, yathā, tulya that indicate the comparison are called comparators. For e.g. yathā, iva, vā, va, vat, sadṛśa, tulya, saṅkāśa, sannibha, upama, nīkāśa, sama, ābha, nibha, pratīkāśa, prakhya, pratinidhi, savarṇa.

Classification Categories of Upamā alaṅkāra:

- Pūrṇopamā (Complete Simile): All four elements are present in the prose or poetry.
- Luptopamā (Ellided Simile): One or more elements, namely, the Upamāna, Upameya, Upamādyotaka or sādhāraṇadharma are missing, but the comparison is implied.
- None: No elements of Upamā alaṅkāra are present.

Input-Output Format:
Input:
- A Romanized Sanskrit prose or poetry excerpt.

Output:
- reason: A text explanation of how the elements of Upamā alaṅkāra are identified or absent.
- label: One of the categories: Pūrṇopamā, Luptopamā, None.

Output Format:
{"reason": "<reason>", "label": "<label>"}

Example 1:
Input: "Bhrātarau aśvinau iva rūpeṇa samupasthitayauvanau "
Output: {
"reason": "All elements are present: Upameya: bhrātarau, Upamāna: aśvinau, Sādhāraṇadharma: rūpeṇa, Upamādyotaka: iva.",
"label": "Pūrṇopamā"
}

Example 2:
Input: ”Kāminīgaṇḍapāṇḍunā candreṇa prācīdik alaṅkṛtā”
Output: {
”reason”: ”The comparative word (Upamādyotaka) is missing, but the comparison is implied.”,
”label”: ”Luptopamā”
}

Example 3:
Input: ”Vṛkṣaḥ sthiraḥ tiṣṭhati.”
Output: {
”reason”: ”No comparison elements are present.”,
”label”: ”None”
}

Give only the output in the specified format and nothing else.

Input: <Sentence S>
Output:

3.2 Phase 2: Upamā Component Identification

Algorithm 2 Component Identification Algorithm

Require: Sentence S classified as *Pūrṇopamā*.
Ensure: Identification of *Upameya*, *Upamāna*, *Sādhāraṇadharma*, *Upamādyotaka*.

1: Construct a prompt with:
   - Definitions of each component.
   - Annotated examples.

2: Input S into the LLM using the constructed prompt.
3: Extract component labels from the LLM output.
4: **return** Identified components in the format:
   - *Upameya:* Extracted object of comparison.
   - *Upamāna:* Extracted standard of comparison.
   - *Sādhāraṇadharma:* Extracted shared attribute.
   - *Upamādyotaka:* Extracted comparator word.

For sentences classified as *Pūrṇopamā*, the second phase focuses on identifying the four key components of *Upamā alaṅkāra*: *Upameya*, *Upamāna*, *Sādhāraṇadharma*, and *Upamādyotaka*. This phase is critical for understanding the structural and semantic intricacies of the figure of speech, as outlined in classical Sanskrit literature. Similar to the classification phase, we again utilize a few-shot prompting technique on an instruction-tuned LLM for this stage as well.

3.2.1 Algorithm for Component Identification

The identification process is guided by a structured algorithm that leverages the capabilities of the LLM. A carefully constructed prompt, containing precise definitions and annotated examples of each component, is used to guide the model in analyzing the input sentence. The algorithm systematically extracts each component by aligning the model's output with predefined roles. For example, *Upameya* is identified as the object of comparison, while *Upamāna* represents the standard of comparison. Similarly, *Sādhāraṇadharma* corresponds to the shared attribute, and *Upamādyotaka* is the comparator word connecting the other elements. This systematic approach ensures that the model provides a detailed breakdown of the components, contributing to a nuanced analysis of the *Upamā alaṅkāra.* The identification process follows the steps presented in algorithm 2.

3.2.2 Prompt Template for Component Identification

We use the following prompt template to prepare the input for the LLMs with the input sentence for the component identification task:

You are a highly knowledgeable language model specializing in classical Sanskrit poetics.You will be given a prose/poetry excerpt in Sanskrit (Romanized) which has presence of the figure of speech called Upamā alaṅkāra. Your task is to identify the essential elements of Upamā alaṅkāra: Upameya, Upamāna, Sādhāraṇadharma, and Upamādyotaka. Upamā alaṅkāra and its elements are described below.

Explanation of Upamā alaṅkāra:
Upamā alaṅkāra (simile) is a poetic device where a comparison is drawn between two entities. The essential elements of Upamā alaṅkāra are:

- Upameya (Object of comparison): The entity being described.
- Upamāna (Standard of comparison): The entity being compared to.
- Sādhāraṇadharma (Common property/State/Event): The common quality between the two. This common quality can be an object or an action. A noun or verb can denote this event in a verse or sentence.
- Upamādyotaka (Comparator): Words like iva, yathā, tulya that indicate the comparison are called comparators. For e.g. yathā, iva, vā, va, vat, sadṛśa, tulya, saṅkāśa, sannibha, upama, nīkāśa, sama, ābha, nibha, pratīkāśa, prakhya, pratinidhi, savarṇa.

Classification Categories of Upamā alaṅkāra:

- Pūrṇopamā (Complete Simile): All four elements are present in the prose or poetry.
- Luptopamā (Elided Simile): One or more elements, namely, the Upamāna, Upameya, Upamādyotaka or sādhāraṇadharma are missing, but the comparison is implied.
- None: No elements of Upamā alaṅkāra are present.

Input-Output Format:

Input:

- A Romanized Sanskrit prose or poetry excerpt.

Output:

- Explanation: Reasoning based on which, the four elements are identified.
- The four elements: upameya, upamāna, sādhāraṇadharma, and upamādyotaka, in the specified format.

Output Format:

{
”upameya”: ”<upameya>”,
”upamāna”: ”<upamāna>”,
”sādhāraṇadharma”: ”<sādhāraṇadharma>”,
”upamādyotaka”: ”<upamādyotaka>”
}

Examples:

Example 1:
Input: ”rāmaḥ kālāgnisadṛśaḥ krodhe ”
Explanation: The comparison here is between ‘rāmaḥ’ and ‘kālāgni’, where ‘rāma’ is the ‘upameya’ and ‘kālāgni’ is the ‘upamāna’. The common property is anger, indicated by the word ‘krodhe’. The upamādyotaka used here is ‘sadṛśaḥ’. Since all four components namely, Upameya, Upamāna, sādhāraṇadharma and upamādyotaka are present, this is an example of Pūrṇopamā.
Output: {
”upameya”: ”rāma”,
”upamāna”: ”kālāgni”,
”sādhāraṇadharma”: ”krodhe”,
”upamādyotaka”: ”sadṛśaḥ”
}

Example 2:
Input: ”sītā api anugatā rāmaṃ śaśinaṃ rohiṇī yathā ”
Explanation: Here, a comparison is being made between ‘sītā’ and ‘rohiṇī’ who is the wife of Candra (moon). The sādhāraṇadharma is indicated by the word ‘anugatā’ which one who follows. The Upamā alaṅkāra is indicated by the upamādyotaka/ comparator ‘yathā’. Since, all

the four components namely, Upameya, Upamāna, sādhāraṇadharma and upamādyotaka are present, this is an example of Pūrṇopamā.
Output: {
    ”upameya”: ”sītā”,
    ”upamāna”: ”rohiṇī”,
    ”sādhāraṇadharma”: ”anugatā”,
    ”upamādyotaka”: ”yatha”
}

Example 3:
Input: ”salabdhamānairvinayānvitairnṛpaiḥ purālayairjānapadaiśca mānavaiḥ upopaviṣṭairnṛpatirvṛto babhau sahasracakṣurbhagavāniva amaraiḥ
Explanation: Here, a comparison is being made between ‘mānavaiḥ’ which means men and amaraiḥ which means Gods. The sādhāraṇadharma is ‘vṛtaḥ’ which means the common property is to encircle. The Upamā alaṅkāra is indicated by the comparator ‘iva’.
Since, all the four components namely, Upameya, Upamāna, sādhāraṇadharma and upamādyotaka are present, this is Pūrṇopamā.
Output: {
    ”upameya”: ”mānavaiḥ”,
    ”upamāna”: ”amaraiḥ”,
    ”sādhāraṇadharma”: ”vṛtaḥ”,
    ”upamādyotaka”: ”via”
}

Give only the output in the specified format and nothing else.

Input: <Sentence S>
Output:

3.3 Integration and Final Output

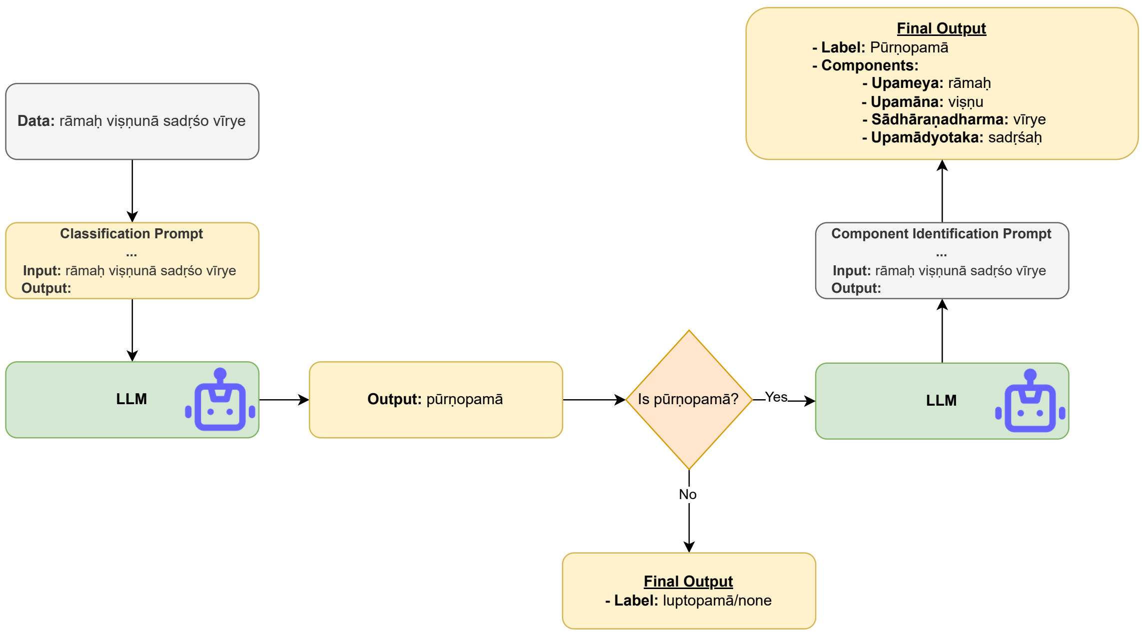

Figure 2: Example run of our pipeline on a datapoint. This shows both the classification and component identification processes.

Figure 2 shows the execution of our pipeline on a datapoint. The outputs from both phases are combined into the final structured result:

- Classification result (*Pūrṇopamā*, *Luptopamā*, or None).
- For *Pūrṇopamā*, the four extracted components.

Task	Dataset	Metric	Value
Classification	*Rāmāyaṇa*	Accuracy	59%
		F1 Score	0.53
Classification	*Raghuvaṃśa*	Accuracy	56%
		F1 Score	0.49
Component Identification	*Rāmāyaṇa*	Exact Match	47%
		Full Match	41%
Component Identification	*Raghuvaṃśa*	Exact Match	44%
		Full Match	39%

Table 1: Results for Classification and Component Identification

4 Experiments and Results

This section describes the experimental setup, evaluation methodology, and results for the proposed LLM-based approach to analyzing *Upamā alaṅkāra* in Sanskrit texts. Our study focuses on two main tasks: the classification of sentences into subtypes of *Upamā alaṅkāra*: *Pūrṇopamā* or *Luptopamā* or *None* and the identification of the components of sentences belonging to *Pūrṇopamā alaṅkāra*.

4.1 Dataset

The datasets for the experiments are derived from two Sanskrit texts that are rich in Upamā, the *Vālmīkīyarāmāyaṇa* and *Raghuvaṃśa*. The selection of texts ensured that we select examples of Upamā alaṅkāra from both simplified and complex Sanskrit language. The datasets are manually annotated to mark the sentences as either *Pūrṇopamā*, *Luptopamā*, or *None*, along with the identification of components for sentences belonging to *Pūrṇopamā alaṅkāra*. The *Raghuvaṃśa* text consists of 129 examples, with 74 examples of *Pūrṇopamā*, 27 examples of *Luptopamā*, and 28 examples of *None*. Similarly, the *Vālmīkīyarāmāyaṇa* text consists of 116 examples, with 70 examples of *Pūrṇopamā*, 39 examples of *Luptopamā*, and 7 examples of *None*. Further, both the datasets have all the four components: *Upameya*, *Upamāna*, *Sādhāraṇadharma*, and *Upamādyotaka*; identified for all the *Pūrṇopamā* examples.

Component	Count (*Rāmāyaṇa*)	% (*Rāmāyaṇa*)	Count (*Raghuvaṃśa*)	% (*Raghuvaṃśa*)
Upameya	14	31.82	11	23.40
Upamāna	20	45.45	10	21.28
Sādhāraṇadharma	11	25.00	6	12.77
Upamādyotaka	31	70.45	36	76.60

Table 2: Component Matches for *Rāmāyaṇa* and *Raghuvaṃśa*

4.2 Experimental Setup

We conduct our experiments with the Llama-3.1 8B model[7], a large language model based on the Transformer architecture, prompted for analysis of *Upamā alaṅkāra*. We utilize a few-shot prompting based strategy, and provide the LLM with the following information in the prompt: *<Context, Task Description, Few-shot Examples, Sentence S>*. The detailed prompts for each of the task in our pipeline have been provided in section 3. Hence, the prompts provide a detailed description of the task, including examples for few-shot learning.

[7]https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

4.3 Evaluation Metrics

The performance of the model is evaluated using classification and component identification metrics. For the classification task, which involves categorizing sentences as Purnopama, Luptopama, or None, we use Accuracy and F1 Score. Accuracy measures the proportion of sentences correctly classified into their respective categories and is defined as

$$\text{Accuracy} = \frac{\text{Correctly Classified Sentences}}{\text{Total Sentences}}. \tag{1}$$

The F1 Score, which provides a harmonic mean of Precision and Recall, is defined as

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{2}$$

where Precision represents the proportion of correctly classified sentences out of all sentences predicted in a particular category:

$$\text{Precision} = \frac{\text{Correctly Classified Sentences in Category}}{\text{Total Predicted Sentences in Category}}, \tag{3}$$

and Recall represents the proportion of correctly classified sentences out of all actual sentences in that category:

$$\text{Recall} = \frac{\text{Correctly Classified Sentences in Category}}{\text{Total Actual Sentences in Category}}. \tag{4}$$

For the component identification task, which involves identifying *Upameya*, *Upamāna*, *sādhāraṇadharma*, and *upamādyotaka*, we use Exact Match and Full Match metrics. Exact Match calculates the proportion of correctly identified components per example:

$$\text{Exact Match} = \frac{\text{Number of Correctly Identified Components}}{\text{Total Components Per Example}}. \tag{5}$$

Full Match evaluates the proportion of examples where all four components are perfectly identified between the system's output and the reference output. It is defined as

$$\text{Full Match} = \frac{\text{Number of Examples with All Components Correctly Identified}}{\text{Total Examples}}. \tag{6}$$

These metrics provide a precise and task-specific evaluation of the model's performance in both sentence classification and component identification tasks.

4.4 Results and Analysis

The results for both tasks on the *Rāmāyaṇa* and *Raghuvaṃśa* datasets are summarized in Table 1. Additionally, Table 2 provides detailed statistics of component matches for *Upamā alaṅkāra* across both datasets.

Classification Performance: The model achieves moderate accuracy (59% and 56%) and F1 scores (0.53 and 0.49) for *Rāmāyaṇa* and *Raghuvaṃśa*, respectively. These results indicate that while the model can distinguish between *Pūrṇopamā*, *Luptopamā*, and non-*Upamā* sentences to some extent, it struggles with finer distinctions, particularly when implicit or ambiguous components are present. Misclassifications often occur in cases where sentences contain all components but are labeled as *Luptopamā* or vice versa, revealing gaps in the model's understanding of contextual nuances.

Task	Ave. Annotator Accuracy	Ave. System Accuracy	Time Taken By Annotator-to-System
Upamā Subclass Classification	63.33%	63.33%	24:1
Pūrṇopamā Component Identification	70.00%	60.00%	10:1

Table 3: Comparison with Human Annotators

Component Identification Performance: The identification task shows varying levels of success for different components. *Upamādyotaka*, being explicitly marked by words like "iva" and "yathā," is the easiest to identify, with match percentages exceeding 70% across both datasets. In contrast, *Sādhāraṇadharma* consistently shows the lowest match percentages (25% and 12.77%), reflecting the difficulty of recognizing shared attributes that are often implicit or context-dependent.

The confusion between *Upameya* and *Upamāna* is a recurring issue, suggesting that the model struggles with semantic roles, particularly when adjectives or phrases overlap these components. Similarly, phrases containing both the comparator and the standard of comparison are sometimes misinterpreted as the common property (*Sādhāraṇadharma*), further complicating the task.

Error Patterns and Linguistic Challenges: The analysis reveals several persistent error patterns that hinder the model's performance in both classification and component identification tasks. Sentences with missing or ambiguous components are often misclassified, indicating the model's difficulty in dealing with partial or implicit information. Moreover, the inflectional nature of Sanskrit poses challenges, particularly in handling *Sandhi*-s (euphonic combinations), which often result in fragmented or incorrect interpretation of words. This issue disrupts the lexical and syntactic coherence required for accurate analysis. Additionally, adjectives describing the *Upameya* are frequently misclassified as part of the component itself, highlighting the model's inability in distinguishing the modifier from the modified. The machine finds it challenging to identify the components particularly when there is variance in the expression of Upamā of various poets such as *Rāmāyaṇa* and *Raghuvaṃśa.* The complexity of expression is one of the key factors that impact the inconsistencies in the identification and labeling of components. Combined with the limited availability of annotated datasets, these challenges underscore the complexities of applying computational techniques to classical Sanskrit texts.

Comparison with Human Annotators: To asses how our system compares with linguists who are acquainted with analysis of *Upamā alaṅkāra*, we conduct a comparison of our proposed approach. This comparison is done on the basis of accuracy for the particular task, and the efforts required are quantified on the basis of average time taken to analyze and annotate a single instance of poetry/prose. The results in table 3 show that our proposed approach performs comparably with the performance of linguists undertaking the aforementioned two tasks, while taking significantly less time to analyze and annotate an instance. For the task of *Upamā* Subclass Classification, both linguists and our proposed approach achieve an accuracy of 63.33% while our system performing the task 24 times faster than a human. For the task of *Pūrṇopamā* Component Identification, linguists achieve an average accuracy of 70%, while our system achieves an average accuracy of 60%, while taking 10 times lesser time than human.

5 Conclusion and Future Work

We study the potential of large language models (LLMs) for analyzing *Upamā alaṅkāra* in Sanskrit literature, specifically focusing on the classification of subtypes of *Upamā alaṅkāra*: *Pūrṇopamā* and *Luptopamā*, and identifying components of poetry/prose belonging to *Pūrṇopamā alaṅkāra*: *Upameya*, *Upamāna*, *Sādhāraṇadharma*, and *Upamādyotaka*.

Our experiments highlight the efficacy of the proposed approach. We show that using LLMs and using few-shot prompting strategy achieves reasonable performance, with F1 scores exceeding 0.5 for classification and acceptable accuracy in component identification, despite the inherent linguistic complexities of Sanskrit. The findings underscore both the promise and challenges of LLMs in Sanskrit computational linguistics, particularly in resolving contextual dependencies and handling linguistic features like Sandhi and compounds. These limitations suggest avenues for refinement. We further show that computational methodologies such as LLMs can achieve comparable accuracy with linguists in terms of achieving the task, while the proposed approach being considerably faster. Compared to linguists, LLMs suffer from hallucinations and show inability to provide proper reasoning for the provided output. We consider this to be a topic for a future study. As we experiment with datasets from sources with different level of morpho-semantic complexity, with *Rāmāyaṇa* consisting of simpler language structures as compared to *Raghuvaṃśa* which exhibits a profound style of *Upamā* expression. Comapred to *Raghuvaṃśa*, our approach shows better performance on dataset extracted from *Rāmāyaṇa*, underlining the inability of LLMs to process complex language structures.

Future work could involve expanding annotated datasets, developing component-specific sub-models, and integrating advanced syntactic and semantic features. Addressing preprocessing challenges, such as Sandhi splitting, and extending the approach to other *alaṅkāras* would enhance applicability.

References

Amruta Barbadikar and Amba Kulkarni. 2023. Yamaka identifier and classifier: A computational tool for the analysis of sanskrit figure of sound. 08.

Amruta Vilas Barbadikar and Amba Kulkarni. 2024. Anuprāsa identifier and classifier: A computational tool to analyze Sanskrit figure of sound. In Arnab Bhattacharya, editor, *Proceedings of the 7th International Sanskrit Computational Linguistics Symposium*, pages 102–112, Auroville, Puducherry, India, February. Association for Computational Linguistics.

Arnab Bhattacharya, editor. 2024. *Proceedings of the 7th International Sanskrit Computational Linguistics Symposium*, Auroville, Puducherry, India. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tuhin Chakrabarty, Yejin Choi, and Vered Shwartz. 2022. It's not rocket science: Interpreting figurative language in narratives. *Transactions of the Association for Computational Linguistics*, 10:589–606, 05.

Rhugved Pankaj Chaudhari, Bhakti Jadhav, Pushpak Bhattacharyya, and Malhar Kulkarni. 2024. SansGPT: Advancing generative pre-training in Sanskrit. In Sobha Lalitha Devi and Karunesh Arora, editors, *Proceedings of the 21st International Conference on Natural Language Processing (ICON)*, pages 432–441, AU-KBC Research Centre, Chennai, India, December. NLP Association of India (NLPAI).

Pawan Goyal and Gérard Huet. 2013. Completeness analysis of a sanskrit reader. In *Proceedings, 5th International Symposium on Sanskrit Computational Linguistics.* DK Printworld (P) Ltd.

Pawan Goyal, Vipul Arora, and Laxmidhar Behera. 2009. Analysis of sanskrit text: Parsing and semantic relations. In *Sanskrit Computational Linguistics*, pages 200–218. Springer.

Qianyu He, Sijie Cheng, Zhixu Li, Rui Xie, and Yanghua Xiao. 2022. Can pre-trained language models interpret similes as smart as human? In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7875–7887, Dublin, Ireland, May. Association for Computational Linguistics.

Gérard Huet. 2005. Computational processing of sanskrit. In *First International Symposium on Sanskrit Computational Linguistics.*

Bhakti Jadhav and Malhar Kulkarni. 2024. Ic analysis in encyclopaedic dictionary of sanskrit on historical principles with special reference to upamā alaṅkāra. In *Journal of Indian Intellectual Traditions Volume XVII-II, June 2024: (1 to 9) (Forthcoming)*. Journal of Indian Intellectual Traditions, June.

Bhakti Jadhav, Amruta Barbadikar, Amba Kulkarni, and Malhar Kulkarni. 2023. Issues in the computational processing of upamāalaṅkāra. In Jyoti D. Pawar and Sobha Lalitha Devi, editors, *Proceedings of the 20th International Conference on Natural Language Processing (ICON)*, pages 180–190, Goa University, Goa, India, December. NLP Association of India (NLPAI).

M. R. Kale. 1957. *The Raghuvamsa Of Kalidasa*. Motilal Banarsidass.

Amba Kulkarni and Monali Das. 2012. Discourse analysis of sanskrit texts. In *Proceedings of the Workshop on Advances in Discourse Analysis and its Computational Aspects*, pages 1–16, Mumbai, India. COLING 2012 Organizing Committee.

Huiyuan Lai and Malvina Nissim. 2024. A survey on automatic generation of figurative language: From rule-based systems to large language models. *ACM Comput. Surv.*, 56(10), May.

Lizhen Liu, Xiao Hu, Wei Song, Ruiji Fu, Ting Liu, and Guoping Hu. 2018. Neural multitask learning for simile recognition. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1543–1553, Brussels, Belgium, October-November. Association for Computational Linguistics.

Gopal Raghynath Nandargikar. 1957. *The Raghuvamsa Of Kalidasa*. Motilal Banarsidass.

Ashequl Qadir, Ellen Riloff, and Marilyn A. Walker. 2016. Automatically inferring implicit properties in similes. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1223–1232, San Diego, California, June. Association for Computational Linguistics.

Pranab Sahoo, Ayush Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications, 02.

Ekaterina V. Shutova. 2011. Computational approaches to figurative language. Technical Report UCAM-CL-TR-803, University of Cambridge, Computer Laboratory, August.

Valmīkīyarāmāyaṇa. 2025. *Valmīkīyarāmāyaṇa*. A project by IIT Kanpur that gives translation and provides with various commentaries of Rāmāyaṇa.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.