Dependency Analysis of Ṛgveda-Saṃhitā: An alignment of the Universal Dependencies framework with the Śābdabodha-based Saṃsādhanī framework

Johan Paul[1], Gayathri Sepuri[1], Sriram Krishnan[2,3] and Amba Kulkarni[4]
[1] Soulax Software Pvt.
[2] Central Sanskrit University, Prayagraj [3] IIIT, Hyderabad
[4] Department of Sanskrit Studies, University of Hyderabad
17johan@gmail.com, sriramk8@gmail.com, ambakulkarni@uohyd.ac.in

Abstract

Integrating Universal Dependencies-based parsing with the traditional grammatical framework of *Śābdabodha* remains a significant challenge in the development of high-quality annotated corpora for Vedic Sanskrit. In this paper, we propose a hybrid methodology for the dependency analysis of the *Ṛgveda-Saṃhitā*, aligning the state-of-the-art ByT5-Sanskrit parser with the Saṃsādhanī (SCL) tagset. We introduce a deterministic transformation ruleset that converts UD graphs into SCL representations through label mapping and graph reduction techniques. Our contributions include the development of a Vedic treebank following the SCL guidelines and a linguistic analysis of the ambiguities encountered when mapping Universal Dependencies labels to the Śābdabodha-based relations. This work establishes a framework for interoperability between diverse annotation schemas in the Vedic domain.

1 Introduction

The creation of high-quality annotated corpora–or **Treebanks**–has become a fundamental prerequisite for computational linguistics tasks such as sentential dependency analysis, discourse analysis, and question answering. Treebanks essentially record the morphological, syntactic, semantic, and discourse elements of sentences and larger textual units. Their applications include the quantitative and qualitative analysis of the linguistic features, diachronic analysis, typological comparison, and the development of sentential and discourse parses.

Having found their use in linguistic, computational, and pedagogical applications, we observe a gradual increase in the size and quality of the treebanks being developed over the years. While this has been the case for global languages like English and some widely-spoken Indian languages like Hindi, Sanskrit remains relatively resource-poor. Currently, there are two primary treebanks that provide lexical, morphological, and dependency annotations: the **Sanskrit Treebank Corpus (STBC)** and the **Vedic Treebank (VTB)**. The STBC (Kulkarni et al., 2020) is a corpus with around 3000 sentences from Classical Sanskrit and modern sources such as textbooks, annotated using the theories of Śābdabodha. The VTB (Hellwig et al., 2020) is dedicated exclusively to Vedic Sanskrit, comprising 4000 sentences drawn from the Vedic literature, annotated using the Universal Dependencies framework. The Digital Corpus of Sanskrit (DCS) (Hellwig, 2010–2024) is another effort towards the development of a large-scale corpus, rich in lexical and morphological annotations, containing approximately 700,000 sentences from over 300 texts across various domains. In recent years, dependency annotations have also been recorded for some of its texts.[1]

It is quite surprising that, despite having a rich cultural and knowledge tradition, with texts ranging across multiple domains, genres, and types, we find very few treebanks for Sanskrit. However, this has more to do with the rigorous nature of treebank creation process involving corpus extraction and collection, validation, normalization, and most importantly high-quality annotations at various levels in morphology, syntax, semantics, discourse, etc. In recent years,

[1] http://www.sanskrit-linguistics.org/dcs/

while each of these stages has been assisted with the advancements in technology, reliability on the machine-generated results is a major concern, especially for the annotations.

Another major concern is related to the design decisions of the treebank creators–specifically, the **treebank guidelines**. These decisions are influenced by several factors, such as language-specific features, the nature of the tasks involved, the purpose served by the annotations, the domain, the grammatical tradition, and the goal of universality. These guidelines also affect later stages of treebank (or corpus) alignment and extension, particularly concerning interoperability among multiple treebanks. For instance, the VTB primarily uses the **Universal Dependencies (UD)** framework to annotate morphology and dependency. In contrast, the STBC follows the Pāṇinian Kāraka theory and the theories of **Śābdabodha**. Although UD allows language-specific features to be recorded, there is a partial overlap between the two frameworks, leading to both one-to-many and many-to-one mappings.

When compounded by the extensive human effort required from corpus compilers, annotators, and validators, these challenges make the entire treebank creation process highly demanding. In this regard, contributions towards annotation platforms and frameworks are crucial, as they establish clear task pipelines through user-friendly interfaces that are designed primarily to support these teams.

Additionally, several general-purpose dependency parsers have been developed for Sanskrit, ranging from rule-based (Huet, 2006; Goyal et al., 2009; Kulkarni et al., 2010; Kulkarni, 2013) to data-driven approaches (Hellwig, 2009; Krishna et al., 2020; Krishna et al., 2023; Sandhan et al., 2023). More recently, Vedic-specific dependency parsers have also emerged (Hellwig et al., 2023; Nehrdich et al., 2024; Vinjamuri and Sun, 2025), among which Nehrdich et al. (2024) is considered the state-of-the-art Vedic dependency parser.[2] This parser utilizes the VTB alongside DCS annotations as its base dataset.

On the other hand, one of the primary rule-based dependency parsers—the Saṃsādhanī parser,[3] developed by (Kulkarni et al., 2019; Kulkarni, 2021) using the theories of Śābdabodha, is widely used for annotation, research, and pedagogical activities. While its performance has been evaluated mainly on classical texts and modern literature, its applicability to Vedic Sanskrit is yet to be explored.

Thus, we address these challenges through three contributions:

1. **Gold Dataset Creation:** Building a Vedic treebank using the Saṃsādhanī guidelines and dependency tags. The present work focuses on the Ṛgveda-Saṃhitā mantras.[4]
2. **UD to SCL conversion:** Developing a formal conversion from UD-tags to SCL-tags to allow interoperability between the two representations.
3. **Silver Data Generation:** Generating UD-based annotations using the state-of-the-art ByT5-Sanskrit parser and converting them to the SCL representation for further validation.

Section 2 provides a summary of each of the Sanskrit treebanks, detailing their annotation schemas and platforms. It also gives an account of the various dependency parsers developed for Sanskrit in chronological order, including those dedicated exclusively to Vedic Sanskrit. Section 3 explains in detail the process of annotating the Ṛgveda-Saṃhitā using SCL tags, presenting an overview of the associated challenges alongside examples and their resolutions. Section 4 provides the methodology for converting UD-based representations to SCL representations. Finally, Section 5 presents comprehensive results and an analysis of the transformations obtained through this process.

[2]This will be referred to as ByT5-Sanskrit.

[3]The dependency parser will be referred to as the SCL-parser while the dependency framework and tagset will be referred to as the SCL-tags.

[4]Belonging to the Śākala-Śākhā of the Ṛgveda.

2 Existing treebanks and Annotation Schema

2.1 Vedic Dependency Treebank (VTB)

Hellwig et al. (2020) released the first treebank of Vedic Sanskrit, consisting of 4004 sentences from the following texts: Ṛgveda Saṃhitā, Atharvaveda Śaunaka Saṃhitā, Maitrāyaṇī Saṃhitā, Aitareya Brāhmaṇa and Śatapatha Brāhmaṇa. The annotation follows the CoNLL-U representation marking the features: word number, unsandhied word (if available), lemma, POS tag, morphological analysis, head position (0 for root), and dependency label. The Universal Dependencies (UD)[5] tags are used for the relations, but minor deviations from the UD were required, especially for the annotations of compound components, preverbs separated from verbs (labeled as *advmod*), head (ellipsis), the quotation marker *iti*, etc. In order to assist the annotators and speed up the annotation process, a neural network based syntactic labeler was built to automatically predict the relation when the annotator chooses the syntactic dependent. The authors also discuss how the manually tagged morphological annotations from the DCS additionally help in predicting the labels.

Digital Corpus of Sanskrit (DCS): The VTB is integrated into the Digital Corpus of Sanskrit, where recently more texts have been added to this treebank, which include the Taittirīya-śākhā texts, Paippalāda-Śākhā-Saṃhitā of Atharvaveda, various Brāhmaṇas, Āraṇyakas and Upaniṣads, and some of the sūtra texts as well. In the recent versions, additionally a few miscellaneous features like dictionary ID, word sense ID, reconstructed word forms from lemma and morphological annotations, punctuation, and annotator information have been added. Table 1 shows the differences in the size of the VTB since its release.

Version	Number of texts	Number of Sentences	Number of Sentences from Ṛgveda
1 (2020)	5	4004	298
2 (2022)	38	18,958	3,462
3 (2023)	57	27,255	4,352

Table 1: Size of the Vedic Dependency Treebank

DCS' representation differs from the traditional representation of the Vedic data due to its design decisions (Krishnan et al., 2025). DCS annotations are based on sentences and not at the level of a mantra, thus ignoring the context of the mantra. And DCS directly annotates the word forms and does not link it to the padapāṭha. The padapāṭha is a device to wrap the underlying word in the segmented form with the features such as original accents of the words, compound component boundaries marked (avagraha), identifying the words which do not undergo sandhi with the subsequent word using the *itikaraṇa* mechanism, etc. Sometimes DCS does not resolve the terminal sandhis. For example, sa (for saḥ), viśvatas (for viśvataḥ). In some rare occasions, two different entries of the padapāṭha are joined together as a single word (for example, *parehi* in DCS vs *parā* / *ihi* in padapāṭha (R.V 10.18.1)).

2.2 Samsādhanī dependency annotation

The development of the Saṃsādhanī (SCL) dependency tagset started with the collection and classification of possible relations (around 100), along with the formulation of annotation guidelines, based on the theories of the Śābdabodha systems (Ramkrishnamacharyulu, 2009). As these relations were highly fine-grained, a subset of these (around 31) were chosen (Kulkarni and Ramkrishnamacharyulu, 2013) and a treebank of 3000 sentences (from modern literature) was developed following these guidelines under the SHMT consortium.[6]

[5] `https://universaldependencies.org/`

[6] This was the first treebank of dependency analysis for Sanskrit, developed under the Sanskrit-Hindi Machine Translation Consortium executing the project entitled 'Development of Sanskrit Computational Tools and

Kulkarni et al. (2020) observed that some of these relations were not semantic in nature, but were either syntactic or referred to the morphological requirement. Thus a revised set of 54 relations along with a new classification of the relations were released. And a new treebank (STBC) was developed from various sources viz. grammar books, NCERT 9th grade Sanskrit text book, various books on Sanskrit learning, and some modern stories. Kulkarni et al. (2020) also discuss several ambiguities observed during the annotation process, especially with non-finite verbs, causative forms, ellipsis, etc. This treebank thus consists of prose sentences majorly from modern texts.

The same tagset and guidelines were used to annotate texts from the classical literature viz. Bhagavad Gītā, Saṅkṣepa-Rāmāyaṇa and Śiśupālavadha, and these have been published online.[7] However, there has not been any attempt to annotate texts in Vedic Sanskrit using the SCL tags. This has been one of the main motivations for the present work to build a treebank of the Vedic corpus using the SCL dependency relations. To assist the annotation process, 'START' (Kumar et al., 2024) has been used as the annotation interface where given a sentence, the tool processes sandhi segmentation, morphological analysis and dependency parsing to produce a tabulated version of the results (similar to the CoNLL-U format, but with different features). The annotators then manually modify the morphology and/or the relations wherever the parser has gone wrong.

2.3 Dependency Parser Implementations

There have been significant efforts to develop dependency parsers for Sanskrit in the past few years. Huet (2006) developed a shallow syntactic parser using semantic nets constraint, which was followed by Goyal et al. (2009) developing a dependency parser using the utsarga-apavāda (rules and exceptions) approach for relation analysis. Hellwig (2009) proposes a hybrid dependency parser that improves upon purely lexical models by incorporating simple syntactic rules and grammatical information. It uses a modified version of Yuret's unsupervised learning algorithm, enhanced with information about grammar, valences, smoothing probabilities, and fixed syntactic rules to reduce improbable analyses. Kulkarni et al. (2019) propose a new rule-based parser designed for both prose and verse that employs an Edge-Centric binary join method. It models the theories of Śābdabodha, considering the conditions based on ākāṅkṣā, yogyatā and sannidhi. The conditions are used to establish relations (edges) between different words (nodes) and the resultant graph is converted to a directed tree using a constraint solver.

Krishna et al. (2020) propose a framework using energy-based models for multiple structured prediction tasks that includes dependency parsing as well. Krishna et al. (2023) evaluates four different data-driven machine learning models (graph-based and transition-based) on the existing Sanskrit treebank, and shows that self-training and contextualized representations benefit the low-resource settings, although the previous energy-based model with all the linguistic features outperformed all the other models. Sandhan et al. (2023) experiment with five low-resource strategies—data augmentation, sequential transfer learning, pretraining, multi-task learning, and self-training across low resource languages and demonstrate a successful ensemble system for Sanskrit which outperforms previous hybrid and data-driven state-of-the-art models in terms of UAS. The evaluation is done on both STBC and VTB.

All of these were built as general-purpose parsers, focusing on Classical or modern Sanskrit texts, their performance on the Vedic texts being either poor or untested. The following are the only three approaches developed exclusively for Vedic Sanskrit.

Data-driven parser for Vedic Sanskrit: Hellwig et al. (2023) propose the first data-driven parser for Vedic Sanskrit using the state-of-the-art contextualized word embeddings and the gold annotations from DCS. A graph-based biaffine parser combined with the character-level CNN, and a deep contextualized self training (DCST), were utilized and the experiments

Sanskrit-Hindi Machine Translation System' sponsored by TDIL Programme, Ministry of Information Technology, Government of India, 2008-12.

[7] `https://start.samsaadhanii.in/`

were carried out with various word representations, including static embeddings (Word2Vec, fasText) and contextualized embeddings (ELMo, RoBERTa, XLM-R). The comparisons revealed that manually validated morpho-syntactic gold data is more effective than large contextualized models given the limited corpus size. Specifically, static fastText embeddings with full gold linguistic information outperformed contextual models. In its optimal configuration using Deep Contextualized Self-Training (DCST), the model achieved 81.84 LAS and 87.61 UAS on held-out test data.

ByT5-Sanskrit: Nehrdich et al. (2024) proposed the **ByT5-Sanskrit** pre-trained language model and evaluated it on word segmentation, Vedic Sanskrit dependency parsing, and OCR post-correction, achieving state-of-the-art results in all the three tasks. The Vedic Sanskrit dependency parsing task was reformulated as a sequence generation task. 24,807 sentences were extracted from DCS along with their gold dependency, part of speech, and morphosyntactic annotation. The evaluation was done in two settings: (1) using only the surface form of the word, without any additional linguistic information, and (2) with all the linguistic features. A comparison with the previous best results (Hellwig et al., 2023) showed an improvement by 2.18 in UAS and 2.60 in LAS. And even without gold linguistic features, ByT5-Sanskrit (LAS 81.54) nearly matched the performance of previous biaffine models that did use gold data (LAS 81.98).

Cross-lingual Transfer learning framework: Vinjamuri and Sun (2025) proposed a cross-lingual transfer learning framework for the dependency parsing task on Vedic Sanskrit and demonstrates that the Transformer-based architecture significantly outperforms the BiLSTM baseline, and the cross-lingual transfer learning framework is more effective and data efficient than the DCST paradigm. The model was pretrained on typologically related high-resource languages viz, Ancient Greek, Latin, and Classical Sanskrit before fine-tuning on Vedic Sanskrit, which proved to be a powerful inductive bias that substantially improved the parsing accuracy, even in rigorous few shot settings.

3 Ṛgveda dependency annotations using Saṃsādhanī tags

The Saṃsādhanī parser follows the traditional preprocessing pipeline of segmentation, morphological analysis and dependency parsing. Given the availability of the Ṛgveda-padapāṭha, which provides segmented text, the initial segmentation stage was bypassed. However, the padapāṭha encodes phonological, morphological, and compound information alongside the underlying words. Since our specific task requires only the underlying words, we extracted them using a heuristic-based pattern-matching mechanism (Krishnan et al., 2025). The built-in morphological analyzer exhaustively generates all possible morphological analyses for each word and forwards them to the parser to mark dependency relations. These relations are established using the *khaṇḍa-anvaya* approach, wherein the main verb is identified first, followed by the assignment of relations based on the rules of Śābdabodha. The parsing pipeline is susceptible to errors at the following stages:

1. failing to generate the correct morphological analysis within the set of possibilities,
2. disambiguating the correct morphological analysis in context,
3. identifying the correct dependency relation contextually, and
4. assigning the correct head for the relation.

While 2, 3, and 4 are interdependent, 1 relies entirely on the coverage of the morphological analyzer. Because it is a lexicon- and paradigm-based analyzer, certain lemmas exclusive to the Vedic period may be unavailable in the lexicon. Consequently, the parser may fail by either selecting an incorrect morphological analysis in context or assigning incorrect relations. The parser could also fail if the main verb is not (overtly) expressed—a common phenomenon in the RV—and it does not automatically detect sentence boundaries. Therefore, the onus lies on the annotator to refer to the commentaries to select the correct morphological analysis, detect (or supply) the main verb if it is not present, mark the sentence boundaries, and correct the errors in the dependency relations. The manual annotation process conducted on the first Maṇḍala of

the Ṛgveda-Saṃhitā is detailed in the following sections.

3.1 Manual annotation of 30 sūktas of the first Maṇḍala

The annotation process comprised three stages: (1) automatic parsing, (2) manual correction and annotation, and (3) expert validation. In the first stage, the SCL-parser was employed to generate a preliminary dependency analysis for each mantra.

During the second stage, the automated SCL dependency outputs were manually corrected. One of the authors—holding a Ph.D. in Sanskrit (Vyākaraṇa) with over six years of research experience—performed these corrections and supplied missing annotations wherever the parser failed. This workflow was primarily conducted using the 'START' platform, which integrates the SCL-parser and presents the preliminary results within an interactive interface, allowing for dynamic validation, correction, and dependency tree generation. Through this procedure, 347 mantras from the first Maṇḍala were fully annotated according to the SCL dependency guidelines.

In the third stage, two Vedic Sanskrit experts[8] were consulted to further validate the annotations and assist in designing modified tagging and representation guidelines tailored specifically for Vedic Sanskrit.[9]

3.2 Challenges in dependency parsing in the Vedic context

The following section outlines the primary challenges encountered across all three phases of the annotation process.

3.2.1 Accent (svara)

The distinguishing features of Vedic Sanskrit can be observed at all levels: *sandhi*, *svara* (accent), morphology, syntax, vocabulary, and usage. A primary distinction from Classical Sanskrit is the usage of accents. Although these accents are crucial for determining word senses and identifying compound types, they were temporarily omitted from the input text because existing engines (segmentation, morphological analyzers, and parsers) do not yet support accent processing. Nevertheless, accent information was useful during the manual validation phase (Stage 3), as it helped disambiguate multiple possible senses based on the context and insights from the traditional commentaries.

3.2.2 Preverbs (*upasargas*)

The padapāṭha entries separate *upasargas* (preverbs) from their corresponding verbs, and these preverbs are often present away from the verbs within the sentence (a phenomenon known as *tmesis*). This syntactic feature poses a representational problem: should these preverbs be treated as free morphemes establishing independent dependency relations with the corresponding verb, or should they be considered bound morphemes requiring them to be merged with their verbs? While the former approach requires assigning a context-specific semantic role to the preverb—thereby diverging from standard dependency tagging guidelines—the latter disrupts the original linear word order of the mantra.

It was observed that both scenarios are prevalent in the Vedic context. Certain preverbs must be merged with the verb, whereas others function as *karmapravacanīya*, denoting an action when an explicit verbal form is absent. Contextual cues are essential to address this ambiguity. Figure 1 depicts both scenarios within a single mantra. Currently, the SCL guidelines do not permit the independent representation of preverbs that get realized along with their verbs. Consequently, they are combined with the verbs, with an underscore indicating the preverb-verb combination.[10]

Alternatively, to maintain a uniform representation across both scenarios, a special tag (**kriyāyoga**) was introduced. This tag marks all preverbs associated with their correspond-

[8] Both experts possess over 40 years of experience in teaching and research in Vedic Sanskrit.

[9] At the time of writing, the validation phase had covered sūktas 1.1 through 1.11, consisting of 110 mantras.

[10] It is to be noted that all the examples are presented without marking the accents.

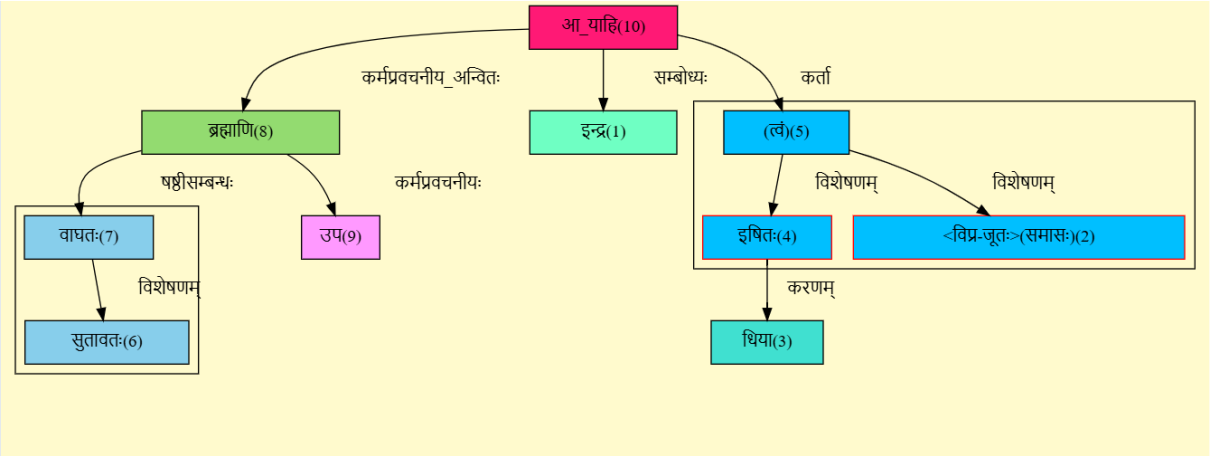

Figure 1: **upasargas**

Mantra: indrā yāhi dhiyeṣito viprajūtaḥ sutāvataḥ | upa brahmāṇi vāghataḥ ||
Padapāṭha: indra | ā | yāhi | dhiyā | iṣitaḥ | vipra'jūtaḥ | sutāvataḥ | upa | brahmāṇi | vāghataḥ || (RV 1.3.5)
Explanation: The *upasarga* **ā** combines with the verb **yāhi** to form **āyāhi** (to come or arrive). The preverb **upa**, indicating the sense of 'to be near' (samīpe), denotes an action even though there is no explict verbal form present. In this case it indicates 'near to brahmāṇi' (brahma here refering to the Veda or all kinds of knowledge). Thus, *upa* is marked as a karmapravacanīya to brahmāṇi, and brahmāṇi is marked as the **karmapravacanīya__anvitaḥ** of the verb **ā__yāhi**.[11]

ing verb, thereby preserving the original mantra word order, while explicitly indicating the preverb's syntactic connection to the *kriyā* (action).

3.2.3 Avagraha

The Padapāṭha employs an *avagraha* (') to indicate the separation of:

1. compound components (e.g., *ratna'dhātamam*, *dive'dive*),
2. stems from suffixes (e.g., *ṛṣi'bhiḥ*, *vīravat'tamam*), and
3. a word from an adjacent *iva*.

The SCL parser expects compound components to be delimited by a hyphen ('-'), whereas stem-suffix pairs must be euphonically combined (i.e., subjected to *sandhi*). Consequently, uniformly replacing the *avagraha* with a hyphen creates structural errors for stem-suffix pairs, resulting in the words being unrecognized by the morphological analyzer. Determining whether to insert a hyphen (for compounds) or to present the combined form of the *pada* (for suffixes) relies heavily on context. Currently, this disambiguation is performed manually during the second stage (annotation and correction).[12]

Figure 2 illustrates an example of compound annotation. At present, the dependency parser's compound analysis module is optimized primarily for Classical Sanskrit; and an analysis on whether Vedic compounds also have the same features is a part of future research.

3.2.4 Irregularities

The padapāṭha contains instances where regular case (vibhakti) suffixes are substituted with irregular endings. Pāṇini and other traditional grammarians provide explicit lists of such occurrences.[13] For example, the nominative dual suffix *au* is frequently substituted with *ā*. Figure 3 illustrates this phenomenon:

[12] While it is theoretically possible to compile a list of all separable suffixes to automate this *sandhi* process, a dedicated Vedic *sandhi* engine is not yet available.
[13] *supāṃ sulukpūrvasavarṇā"ccheyāḍāḍyāyājālaḥ* – Aṣṭādhyāyī 7.1.39.

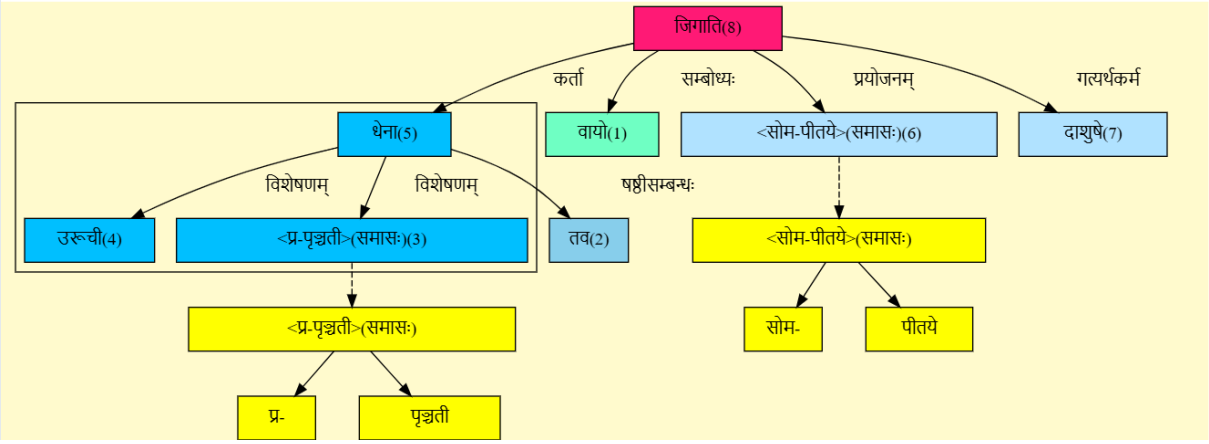

Figure 2: Avagraha for compound-component separation

Mantra: ṛtena mitrāvaruṇāvṛtāvṛdhāvṛtaspṛśā | ṛtuṃ bṛhantamāśāthe ||
Padapāṭha: ṛtena | mitrāvaruṇau | ṛta'vṛdhau | ṛta'spṛśā | ṛtum | bṛhantam | āśāthe iti || (RV 1.2.8)

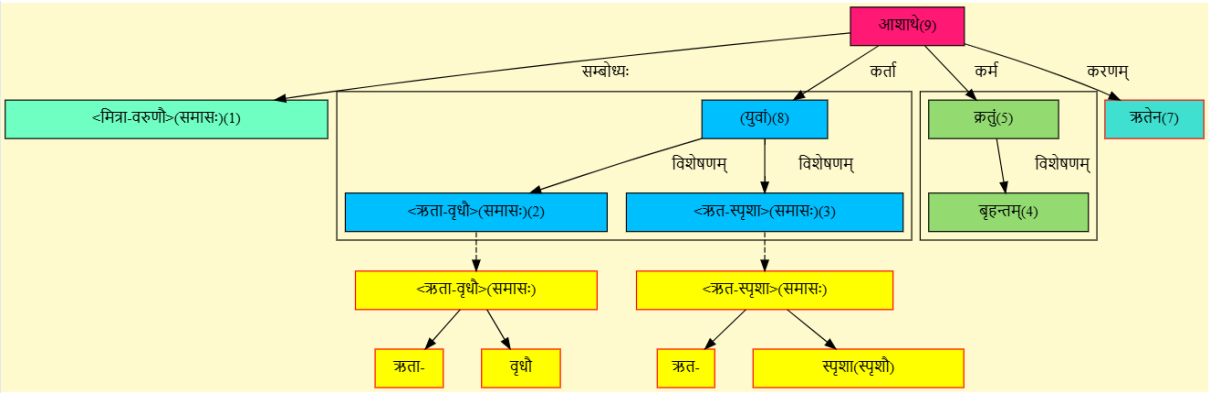

Figure 3: Suffix-based irregularities in Vedic Sanskrit

Two critical points arise here. First, the word *ṛta'spṛ'sā* (with the transformed suffix *ā*) neither satisfies the *ākā.nkṣā* of the main verb nor relates syntactically to any other contextual word in its current form. It must be interpreted as *ṛta-spṛ'sau* (using the standard *au* suffix). Because the Vedic context introduces this modification, the SCL-parser may fail to analyze it during the initial morphological processing. To address this at the treebank level, both versions are recorded on the node, placing the grammatically expected form in parentheses: *spṛ'sā(spṛ'sau)*.

Second, we observe a metrical elongation in the mantra for the word *ṛta-vṛdhau*, appearing as *ṛtā-vṛdhā*. Although the *saṃhitā-pāṭha* introduces such elongations, we strictly retain the *padapāṭha* representation. Consider another example:

Mantra: indramidgāthino bṛhadindramarkebhirarkiṇaḥ | indraṃ vāṇīranūṣata ||
Padapāṭha: indram | it | gāthinaḥ | bṛhat | indram | arkebhiḥ | arkiṇaḥ | indram | vāṇīḥ | anūṣata || (RV 1.7.1)

Here, *bṛhat* is annotated alongside its contextually and grammatically congruent form (*bṛhatā*), reinstating the suffix lost due to A 7.1.39. Similarly, *vāṇīḥ* is annotated with *vāgbhiḥ* to resolve semantic ambiguities.

Supplying the main verb: We frequently observe instances where a single verb (*anūṣata*), despite appearing at the end of the mantra, governs multiple preceding clauses. In such cases,

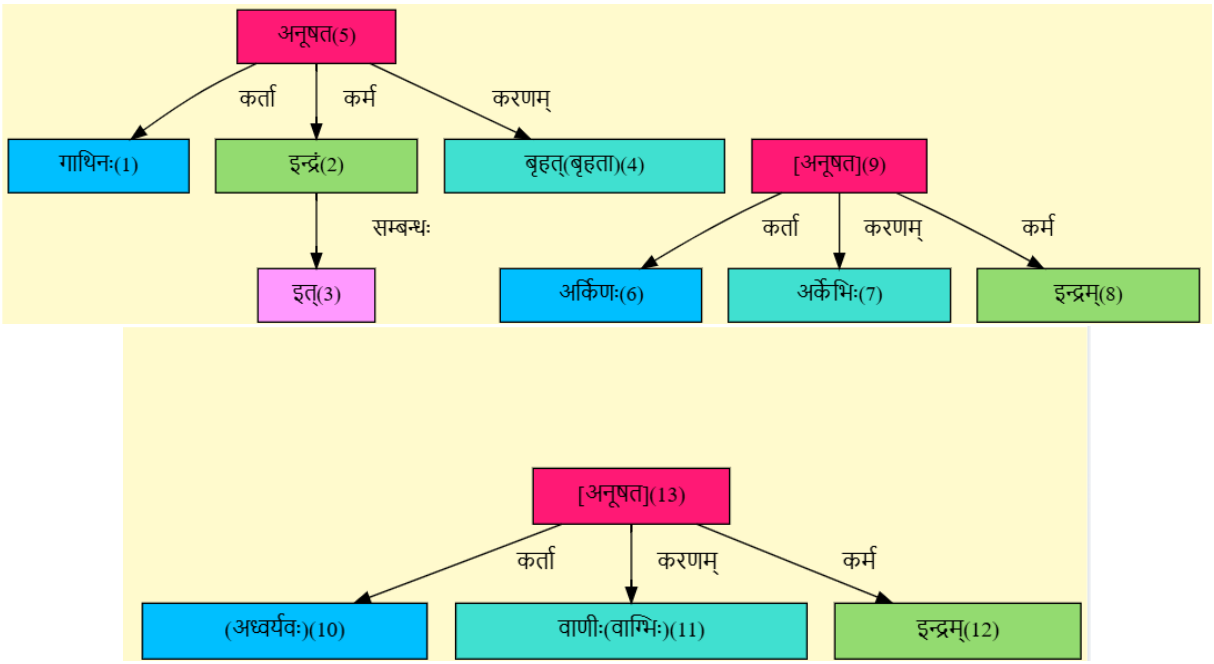

Figure 4: Using actual words in context within brackets

the verb is explicitly supplied to each clause. Because the Ṛgveda-Saṃhitā is a poetic text, verbal elision is common, necessitating this manual insertion. Furthermore, the third sentence lacks an inherent *kartā* (agent), which must be inferred contextually. Traditional commentaries supply *adhvaryavaḥ* as the *kartā* for *anūṣata*, precluding the agents from the mantra's other sentences. Consequently, *adhvaryavaḥ* is manually supplied as the *kartā* during annotation.

Sentence boundaries: A single mantra could have multiple sentences, with the sharing of words across sentences as observed in such cases. Detecting the main verbs for each of sub-sentence is challenging and determining the optimal points to split a mantra into individual sentences remains a critical annotation decision.

Upādhi: We observe specific syntactic behaviours concerning names of various deities across Vedic texts. Consider the phrase *viśve devāsaḥ* in the following mantra:

> **Mantra:** omāsaścarṣaṇīdhṛto viśve devāsa āgata | dāśvāṃso dāśuṣaḥ sutam ||
> **Padapāṭha:** omāsaḥ | carṣaṇi-dhṛtaḥ | viśve | devāsaḥ | ā | gata | dāśvāṃsaḥ | dāśuṣaḥ | sutam || (RV 1.3.7)

There are two traditional interpretations for the word *vi'sve*. According to Yāska,[14] it is a synonym for *sarva* (indicating "all" or "everything"). However, according to Sāyaṇācārya, *vi'sve* refers to a specific class of deities rather than acting as a mere synonym for *sarva*. Under this interpretation, it is not in syntactic agreement with *devāsaḥ* or *omāsaḥ*, and thus cannot be treated as a standard *vi'seṣaṇa* (adjective) for *devāsaḥ*. To handle such cases, a new tag, **upādhi**, was introduced. Figure 5 presents the resulting dependency annotations.

3.2.5 Mulitple interpretations

Being one of the world's oldest literary texts, the RV possesses a vast history of traditional commentaries spanning diverse cultural, regional, and temporal contexts. Among the numerous commentaries and translations available, we primarily relied on those of Sāyaṇācārya and Maharshi Dayanand Sarasvati (MDS) for comparative analysis. However, because the tradi-

[14] Yāska is the author of the Nighaṇṭu and the Nirukta, which provide etymological analyses of complex Vedic words.

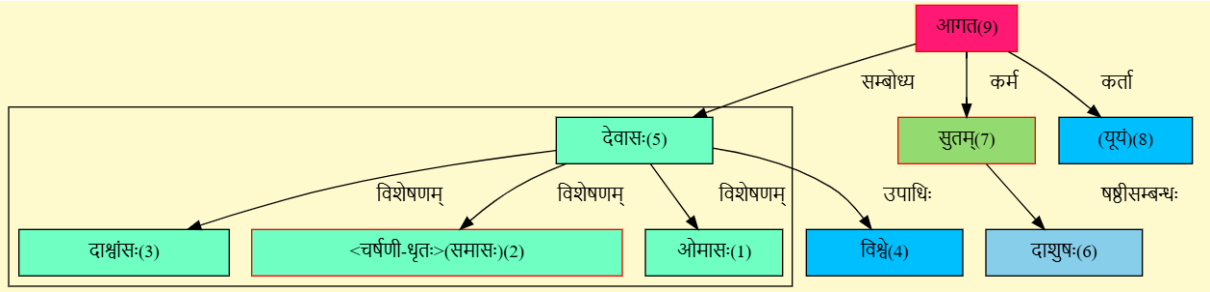

Figure 5: An example for *upādhi*

tion widely considers Sāyaṇācārya's interpretation as the most authoritative, we grounded our primary annotations in his commentary.

3.2.6 SCL Dependencies revision

Based on these challenges, we summarize the major modifications made to the existing tagset, guidelines, and representation schema for Vedic Sanskrit:

1. The padapāṭha's morpho-phonemic irregularities are accommodated by recording both the original padapāṭha form and its syntactically accurate counterpart.
2. Preverbs are either explicitly attached to their verbs (using an underscore to denote a non-terminating chunk) or marked as kriyāyoga when not functioning as a karmapravacanīya.
3. Three new dependency relations have been formally introduced: **kriyāyoga** (for preverbs attaching to verbs), **karmapravacanīya** (for preverbs independently indicating action), and **upādhi** (for class or title).

4 A Hybrid Dependency Parser

This section outlines the pipeline developed for the dependency parsing of Vedic Sanskrit. The process employs a hybrid approach that utilizes the strong structural parsing capabilities of a language model (ByT5-Sanskrit) alongside a deterministic, rule-based transformation layer. This layer enforces the specific semantic constraints of the Śābdabodha framework, effectively bridging the language-agnostic Universal Dependencies (UD) framework with the semantically rich SCL representation. The pipeline is presented in Figure 6 and detailed below.

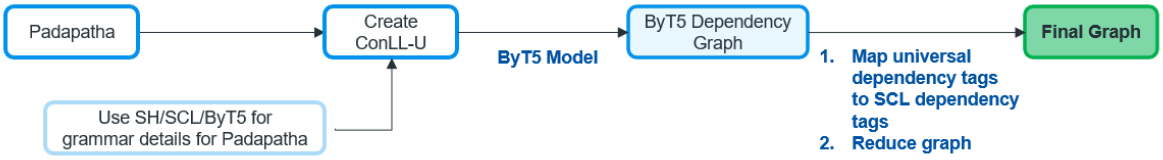

Figure 6: Pipeline of the Hybrid Dependency Parsing using ByT5-Sanskrit and the SCL tagset

4.1 System Pipeline

This two-stage pipeline transforms raw Vedic text into a final dependency graph annotated with SCL tags. Because the ByT5-Sanskrit parser requires segmented input in the CoNLL-U format accompanied by morphological analyses, an initial preprocessing layer is required.

4.1.1 Preprocessing

This stage extracts words from the *Ṛgveda-padapāṭha* alongside their morphological analyses generated by the Sanskrit Heritage (SH) and Saṃsādhanī (SCL) morphological analyzers.[15] If

[15]Because both SH and SCL produce multiple possible analyses, these were aligned with the ground truth analysis of the DCS to map to a single ground truth representing features from all systems.

both platforms fail to produce an analysis, outputs from the ByT5-Sanskrit model are utilized as a fallback. These combined analyses are then formatted into the CoNLL-U representation (see Figure 7) to be passed to the parser.

```
# text = agnim īḍe purohitam yajñasya devam ṛtvijam hotāram ratna dhātamam
#
1       agnim   agni    _       _       Case=Acc|Gender=Masc|Number=Sing        _       _       _       _
2       īḍe     īḍ      _       _       Mood=Ind|Number=Sing|Person=1|Tense=Pres        _       _       _       _
3       purohitam       purohita        _       _       Case=Acc|Gender=Masc|Number=Sing        _       _       _       _
4       yajñasya        yajña   _       _       Case=Gen|Gender=Masc|Number=Sing        _       _       _       _
5       devam   deva    _       _       Case=Acc|Gender=Masc|Number=Sing        _       _       _       _
6       ṛtvijam ṛtvij   _       _       Case=Acc|Gender=Masc|Number=Sing        _       _       _       _
7       hotāram hotṛ    _       _       Case=Acc|Gender=Masc|Number=Sing        _       _       _       _
8       ratna   ratna   _       _       Case=Cpd|Gender=Neut    _       _       _       _
9       dhātamam        dhātama _       _       Case=Acc|Gender=Masc|Number=Sing        _       _       _       _
```

Figure 7: CoNLL-U representation of the padapāṭha with the morphological analyses from SH, SCL or ByT5-Sanskrit

4.1.2 Stage 1: Universal Dependency Parsing

In the first stage, each mantra is processed by ByT5-Sanskrit, the current state-of-the-art language model for Vedic Sanskrit dependency parsing (Nehrdich et al., 2024). Built upon the token-free ByT5 architecture (Xue et al., 2021)—which evolved from the multilingual mT5 model (Raffel et al., 2020)—it operates directly at the byte level without relying on a pre-learned vocabulary. This model generates the initial dependency parse using standard UD labels.

4.1.3 Stage 2: UD-to-SCL Transformation

During the second phase, the UD-annotated parse tree undergoes the deterministic transformation. A custom-built, rule-based system translates the standard UD labels into specific SCL tags and structurally enhances the graph to strictly align with the SCL guidelines.

4.2 Universal Dependency to Samsādhanī Transformation Ruleset

The procedure for converting from the UD schema to the SCL framework is a two-phase process. The first phase focuses on converting individual dependency labels, while the second phase refines the overall graph structure.

4.2.1 Stage 1: Dependency Label Conversion

The initial stage involves mapping individual UD labels to their SCL equivalents. This is achieved through two distinct rule sets: (1) direct lexical mappings for unambiguous tags, and (2) intricate, morphology-dependent mappings for tags requiring grammatical context.

Direct Lexical Mappings: Context-free transformations as depicted in table 2 are applied to UD tags that have a clear and unambiguous counterpart in the SCL framework. Their semantic functions are consistent and do not require additional morphological information for correct interpretation.

Morphology-Dependent Mappings: For ambiguous UD tags like **obl** (oblique nominal) and **nsub** (nominal subject), direct mapping is insufficient. The system employs a secondary set of rules that leverages morphological information—such as the grammatical case of the dependent word or the voice of its verbal head—to assign a precise SCL relation.

- An **obl** relation is mapped to **Karma** if the dependent word is in the 2nd case (accusative).
- An **obl** relation is mapped to **Karaṇam** if the dependent word is in the 3rd case (instrumental).
- An **obl** relation is mapped to **Sampradānam** if the dependent word is in the 4th case (dative).
- An **obl** relation is mapped to **Apādānam** if the dependent word is in the 5th case (ablative).
- An **obl** relation is mapped to **Adhikaraṇam** if the dependent word is in the 7th case (locative).

Universal Dependency Tag	**Saṃsādhanī Tag**
obj (Object)	Karma
conj (Conjunct)	Samuccitaḥ
cc (Coordinating conjunction)	Samuccita_dyotakaḥ
voc (Vocative)	Sambodhyaḥ
iob (Indirect object)	Sampradānam
cas (Case marking)	Karmapravacanīyaḥ
dis (Discourse element)	Sambandhaḥ
adv (Adverbial modifier)	Kriyāviśeṣaṇam
nmod, nmod:appos, det, acl, amo, num	Viśeṣaṇam

Table 2: Mapping between Universal Dependency and Saṃsādhanī Tags

- An **nmod** (nominal modifier) relation is mapped to **ṣaṣṭhīsambandhaḥ** if the dependent word is in the 6th case (genitive).
- An **nsub** (nominal subject) relation is mapped to **Karma** if its verbal head possesses passive, *ṇyat*, *yat*, or *Gdv* (gerundive) properties; otherwise, it is mapped to **Kartā**.
- An **obl** relation is mapped to **Kartā** if the dependent word is in the instrumental case and its verbal head has passive, *Nyat*, *yat*, or *Gdv* properties.
- A **cc** (coordinating conjunction) relation is mapped to **anyatara_dyotaka** if it has a child node "vā".
- An **obl** relation is mapped to **karmapravacanīya-anvitaḥ** if the dependent is in the 2nd case and is followed by a **cas** (case marking) relation.

4.2.2 Stage 2: Graph Reduction

After label conversion, a set of five precedence-ordered rules is applied to simplify and normalize the dependency graph. This restructuring is crucial for resolving complex conjunctions and collapsing redundant modifier chains into a flatter, semantically transparent representation.

- **Conjunct Promotion (Rule 1):** This fundamental rule distributes a grammatical relation across two or more conjuncts. If word B is a conjunct (*samuccita*) of A, and A is related to its head C with the relation r, the graph is transformed so that both A and B are individually related to C with the same relation r.

- **Repetition (*Vīpsā*) Identification (Rule 2):** This rule identifies *vīpsā*, a grammatical device in Sanskrit for expressing distributiveness or intensity through repetition. When two identical, adjacent words are linked by a conjunct relation, the label is converted to the more precise *vīpsā* tag.

- **Modifier Chain Collapse (Rule 3):** This rule flattens chains of *Viśeṣaṇam* (modifier) relations. If node A modifies B, and B modifies C, the structure is simplified by making both A and B direct modifiers of C.

- **Conjunctive Particle Re-attachment (Rule 4):** This rule addresses cases where a conjunctive particle (*samuccita_dyotakaḥ*) is attached to a conjunct rather than the main head of the phrase. It promotes the particle by re-attaching it directly to the main head, simplifying the hierarchy.

- **Complex Conjunction Resolution (Rule 5):** This rule addresses dependency chains containing a conjunctive marker (*samuccita_dyotakaḥ*) and a conjunct relation, eliminating intermediate dependencies to clarify the structure.
  - **Rule 5a:** If R1 is **not** a kāraka relation (kartā, karma, karaṇam, apādānam, or sampradānam), both conjunct elements are promoted to be direct dependents of the head under relation R1.
  - If R1 is a kāraka relation (kartā, karma, karaṇam, apādānam, or sampradānam), the primary word retains its relation to the head, while the conjunct is reattached as a modifier to preserve semantic hierarchy.

4.3 Illustrations

The following example illustrates the three stages of graph conversion for a sample mantra (RV 1.1.1). Refer figure 8 for a visual illustration.

- **Initial UD Graph:** The first graph represents the raw output from the ByT5-Sanskrit parser. It exhibits a deep, chained structure typical of UD parses. For example, *agnim* is the Object of the root *īḷe*, but it serves as the head of a long chain of appositional and conjunctive modifiers (*purohitam*, *devam*, *ṛtvijam*, *hotāram*, *ratna-dhātāmam*). This structure is syntactically correct but semantically indirect.

- **Intermediate Conversion Graph:** The second graph displays the results after applying the label conversion rules. The dependency structure remains identical to the initial parse, but all UD labels have been mapped to their SCL equivalents. *Object* is now *Karma*, *Conjunct* is *Samuccita*, and various modifier types have been unified into *Viśeṣaṇam* or specified as *ṣaṣṭhīsambandhaḥ*. The color-coding system in the second graph helps distinguish between *Magenta labels* (direct UD-to-SCL), and *Purple labels* (mappings dependent on morphological features).

- **Final Reduced Graph:** The third graph shows the final output after the graph reduction rules—specifically Rule 3 (Modifier Chain Collapse) and Rule 1 (Conjunct Promotion)—have been applied. The deep chain of modifiers has been collapsed, resulting in a flatter, semantically direct representation. The modifiers (*purohitam*, *devam*, *ṛtvijam*, *hotāram*, *ratna-dhātāmam*) are now all direct dependents of *agnim* with the *Viśeṣaṇam* relation. Local dependencies are re-evaluated within the new structure; for example, *purohitam* remains the head of *yajñasya*.

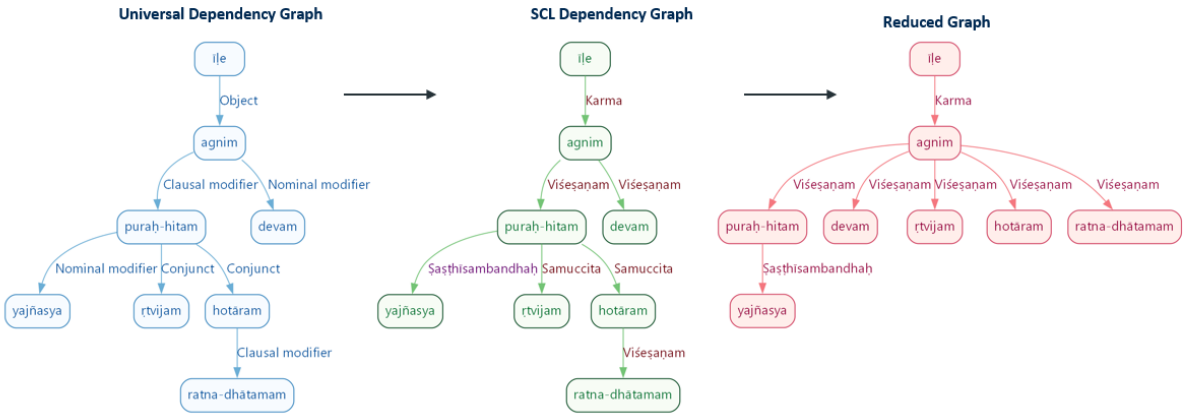

Figure 8: Illustration for UD to SCL conversion for R.V 1.1.1

4.4 Evaluation Corpus and Evaluation Protocol

The manually annotated dataset (consisting of 347 mantras) described in section 3.1 is used as the evaluation corpus. The performance of the complete parsing pipeline was assessed using two standard metrics from the dependency parsing literature, Unlabeled Attachment Score (UAS) and Labeled Attachment Score (LAS). These metrics provide a quantitative measure of both structural and semantic accuracy. While UAS evaluates the system's ability to correctly identify the grammatical structure of a sentence (independent of the dependency label), LAS provides a measure of complete parsing accuracy, assessing both structural identification and semantic classification performance.

5 Results and Analysis

This section presents an empirical evaluation of the system's performance on the 347-mantra test corpus from the Ṛgveda. This analysis directly evaluates the efficacy of the hybrid UD-to-SCL transformation ruleset, pinpointing which rules succeed and where the deterministic mapping proves insufficient for complex semantic disambiguation.

5.1 Overall Parsing Performance

The macro-average UAS and LAS correspond to **51.41%** and **42.38%**, respectively. These results indicate a moderate capacity to identify the correct syntactic structure of the Vedic verses. However, the substantial 9.03% gap between UAS and LAS serves as the first clear quantitative indicator that the primary system failure mode is semantic label classification rather than structural attachment.

5.2 Distributional Analysis of Parser Accuracy

An analysis of the performance distribution across the 347 mantras reveals significant variability, indicating that the system's effectiveness is highly dependent on the syntactic complexity of the input verse as shown in figure 9.

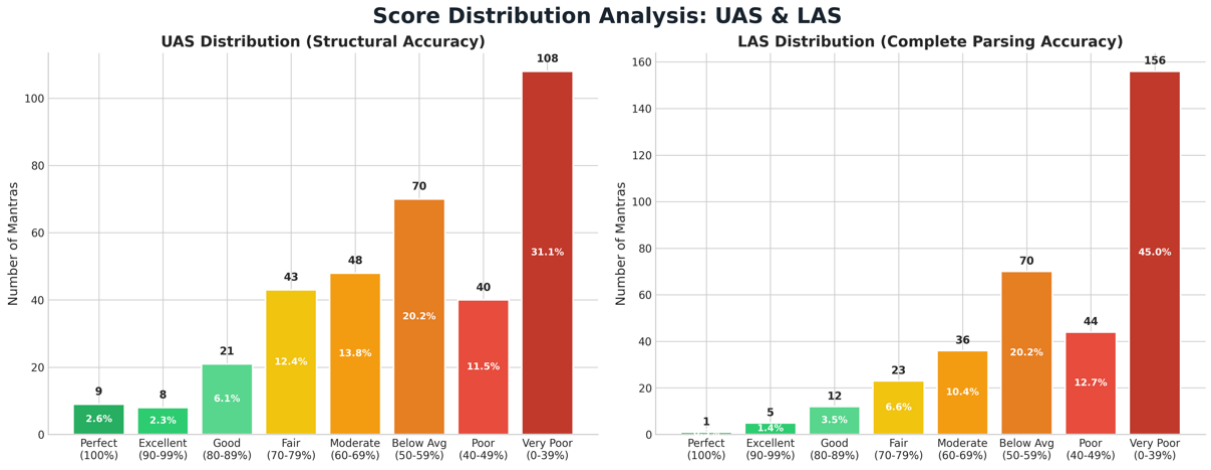

Figure 9: Distribution analysis of UAS and LAS

- For structural accuracy (UAS), 11.0% of the mantras achieve good-to-excellent scores (80% UAS). However, a large portion (42.7%) exhibits poor structural understanding, with UAS scores below 50%.
- The distribution for complete parsing accuracy (LAS) shows greater variance. A substantial 45.0% of mantras fall into the "very poor" category (<40% LAS), underscoring the system's inconsistent performance on semantically complex sentences.

Percentile analysis (figure 10) further confirms this trend. Top-performing mantras (90th percentile) achieve 70.00% LAS, while the median performance sits at 42.11% LAS. This indicates a system with moderate but highly variable capabilities.

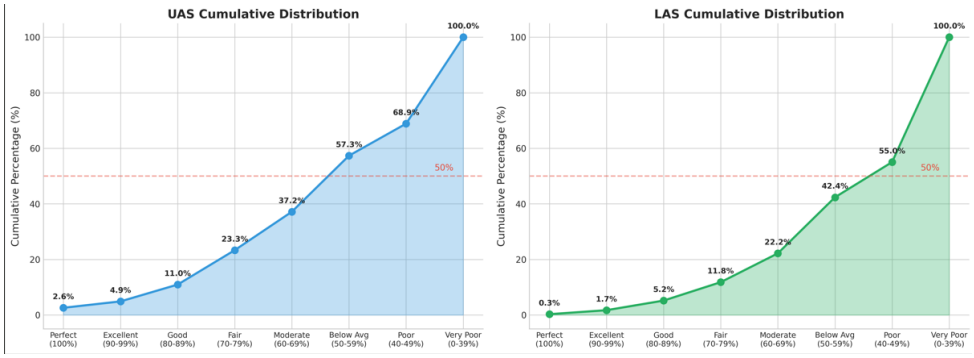

Figure 10: Cumulative distribution analysis of UAS and LAS

5.3 A Deep Dive into Link-Type Errors

To understand the root causes of the gap between structural and semantic accuracy, this analysis focuses on link-type errors–cases where the parser identifies the correct syntactic head but assigns an incorrect dependency label.

A total of 370 link-type errors were identified across 215 mantras, representing an error rate of 62.0% and comprising 114 unique error patterns. On average, each mantra contained 1.07 such errors. Figure 11 presents the error distribution based on the SCL-tags.

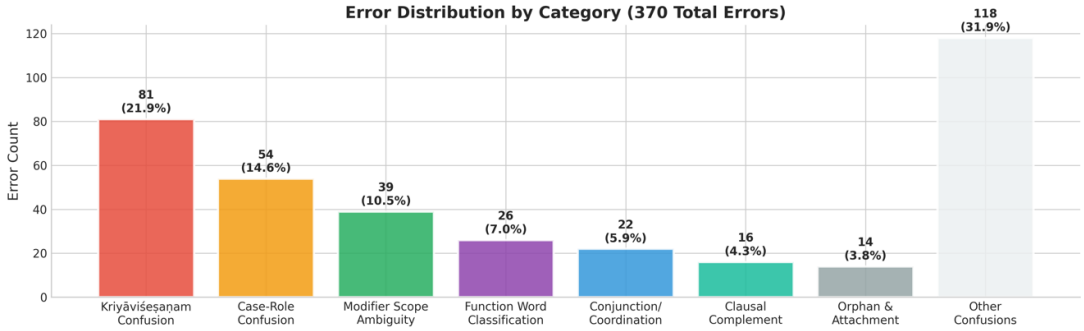

Figure 11: Error Distribution based on relations

The five most frequent error patterns, detailed in table 3, account for over a third of all mistakes and reveal systematic semantic confusion. Figure 12 illustrates this confusion specifically for *Kriyāviśeṣaṇam* and *Kāraka roles*.

Rank	Predicted Label	Correct Label	Count
1	KriyāViśeṣaṇam	Adhikaraṇam	36
2	KriyāViśeṣaṇam	prayojanam	33
3	Karaṇam	sahārthaḥ	26
4	Sampradānam	prayojanam	18
5	KriyāViśeṣaṇam	Sambandhaḥ	12

Table 3: Error Analysis of Predicted vs. Correct Labels

A linguistic review of the top three error patterns highlights the subtleties that challenge the system:

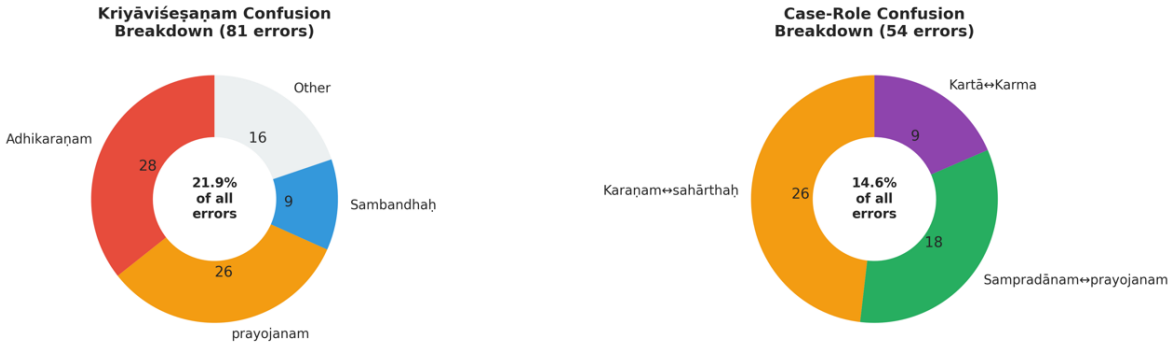

Figure 12: Confusion for *Kriyāviśeṣaṇam* and *Kāraka roles*

- **KriyāViśeṣaṇam → Adhikaraṇam:** This is a classic confusion between an adverbial role (denoting manner or time) and a locative role (denoting spatial location). This ambiguity often arises because both semantic roles can be expressed using similar morphological cases.
- **KriyāViśeṣaṇam → prayojanam:** This error reflects the fine semantic distinction between an adverbial of manner (how an action is performed) and a purposive role (why it is performed). This challenge is compounded by the use of similar grammatical forms (e.g., datives, infinitives) for both concepts.
- **Karaṇam → sahārthaḥ:** This error highlights the inherent ambiguity of the instrumental case, which can denote either instrumentality (*Karaṇam*, "by means of") or associativity (*sahārthaḥ*, "together with"). The model frequently mistakes accompaniment for instrumentality.

5.4 Systematic Error Categorization and Key Findings

The link-type errors are not random but cluster into distinct linguistic categories, revealing the system's primary weaknesses. A selected confusion scores are presented in figure 12.

- **KriyāViśeṣaṇam Confusion Cluster:** This is the single largest source of error, accounting for 21.9% of all instances. It includes all errors where KriyāViśeṣaṇam (adverbial) was either incorrectly predicted or missed in favor of another label.
- **Case-Role Confusion Cluster:** This category, driven by the morphological ambiguity of cases, accounts for 14.6% of all errors. Key examples include the confusion between instrumental for Karaṇam vs. sahārthaḥ and dative for Sampradānam vs. prayojanam.

This detailed error analysis confirms that the hybrid conversion-based approach is viable and, crucially, interpretable. The identified error patterns provide a clear and actionable roadmap for future improvements.

6 Conclusion & Future works

In this paper, we have presented a systematic approach to the dependency analysis of Vedic Sanskrit, bridging the gap between the modern Universal Dependencies framework and the traditional Saṃsādhanī (SCL) representation based on the theories of Śābdabodha. By developing a hybrid pipeline that leverages the results of the ByT5-Sanskrit model and a deterministic UD-to-SCL transformation ruleset, we have demonstrated a path towards automated, semantically-grounded analysis of the Ṛgveda-Saṃhitā.

Our empirical evaluation on a 347-mantra test set yielded a UAS of 51.41% and an LAS of 42.38%. While these scores reflect the inherent difficulty of Vedic dependency parsing, the qualitative analysis of error clusters–specifically the confusion between KriyāViśeṣaṇam (adverbials) and Adhikaraṇam (locatives)–provides insights for future refinement of the transformation rules.

Based on our error analysis and methodological observations, we identify four primary avenues for future discussion and research:

- **Morphological Tagset Alignment:** The morphological analyses from the SH and SCL platforms were provided to the ByT5-Sanskrit model. However, ByT5-Sanskrit relies on the DCS morphological analysis, which operates on UD-based morphological tags. Differences between the DCS and SH-SCL morphological analysis may affect ByT5 model's initial output.
- **Direct VTB Conversion and Comparison:** Although the Vedic Treebank (VTB) records the ground truth analysis for a subset of the Ṛgveda mantras, these have not yet been subjected to a direct conversion into the SCL framework.[16] Evaluating the UD-to-SCL conversion on VTB annotations and comparing it with our manually annotated SCL dataset will yield deeper insights to improve the pipeline and provide a qualitative contrast between the two frameworks.
- **Parser Benchmarking:** A direct comparison of the native Saṃsādhanī parser's dependency analysis against our hybrid approach and the raw ByT5-Sanskrit results will form a crucial part of future work to enhance the rule-based parser's capacity to handle Vedic Sanskrit.
- **Poetry vs. Prose Dynamics:** Both treebanks are hybrid collections of poetry and prose. A comparative analysis of parser performance and transformation mechanism across these distinct literary styles will provide valuable nuances regarding syntactic behavior.

Thus, as a result of this study, we provide three distinct datasets with dependency annotations:[17]

- **Silver Data:** Automatically generated UD-to-SCL parses for all available mantras of the *Ṛgveda-Saṃhitā*.
- **Gold-Annotated Data:** A set of 347 mantras from the first *Maṇḍala*, which is manually corrected and annotated.
- **Gold-Validated Data:** A high-precision subset of 102 mantras, further validated through consultation with senior Vedic scholars.

Acknowledgements

The authors would like to thank Dr. Bhagyalatha Pataskar and Dr. Jayashree Sathe for their invaluable expertise and guidance. Their insights into Vedic grammar, the structural nuances of the Vedas, the complexities of the padapāṭha, and their deep analysis of the Ṛgveda through the lens of the Śābdabodha theories, were instrumental in shaping the direction of this research.

The present work is a part of a larger initiative titled 'Svarupa'.[18] We wish to express our sincere gratitude to Shri Arimilli Ramana Rao for conceptualizing the project and providing the foundational vision that made this study possible.

References

Erica Biagetti, Chiara Zanchi, and Silvia Luraghi. 2023. Linking the Sanskrit WordNet to the Vedic dependency treebank: a pilot study. In German Rigau, Francis Bond, and Alexandre Rademaker, editors, *Proceedings of the 12th Global Wordnet Conference*, pages 77–83, University of the Basque Country, Donostia - San Sebastian, Basque Country, January. Global Wordnet Association.

Pawan Goyal, Vipul Arora, and Laxmidhar Behera. 2009. Analysis of sanskrit text: Parsing and semantic relations. In Gérard Huet, Amba Kulkarni, and Peter Scharf, editors, *Sanskrit Computational Linguistics*, pages 200–218, Berlin, Heidelberg. Springer Berlin Heidelberg.

[16] While the training of the ByT5-Sanskrit parser implicitly includes the VTB, there is an error rate of >10% for UAS and >15% for LAS in the dependency relations it produces. Because the VTB is also manually curated, analyzing the structural differences between the UD-based VTB and SCL-based treebanks is essential.

[17] All our data is released here: `https://github.com/17johan-paul/RigVeda-SCL-Dependency-Dataset`

[18] `https://svarupa.org/`

Oliver Hellwig, Salvatore Scarlata, Elia Ackermann, and Paul Widmer. 2020. The treebank of vedic Sanskrit. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5137–5146, Marseille, France, May. European Language Resources Association.

Oliver Hellwig, Sebastian Nehrdich, and Sven Sellmer. 2023. Data-driven dependency parsing of vedic sanskrit. *Language Resources and Evaluation*, 57(3):1173–1206, Sep.

Oliver Hellwig. 2009. Extracting dependency trees from sanskrit texts. In Amba Kulkarni and Gérard Huet, editors, *Sanskrit Computational Linguistics*, pages 106–115, Berlin, Heidelberg. Springer Berlin Heidelberg.

Oliver Hellwig, 2010–2024. *The Digital Corpus of Sanskrit (DCS)*.

Gérard Huet. 2006. Shallow syntax analysis in sanskrit guided by semantic nets constraints. In *Proceedings of the 2006 International Workshop on Research Issues in Digital Libraries*, IWRIDL '06, New York, NY, USA. Association for Computing Machinery.

Amrith Krishna, Bishal Santra, Ashim Gupta, Pavankumar Satuluri, and Pawan Goyal. 2020. A graph-based framework for structured prediction tasks in Sanskrit. *Computational Linguistics*, 46(4):785–845, December.

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

Sriram Krishnan, Sepuri Gayathri, and Amba Kulkarni. 2025. Challenges in processing Vedic Sanskrit: Towards creating a normalized dataset for the Ṛgveda-saṃhitā. In Amba Kulkarni and Oliver Hellwig, editors, *Computational Sanskrit and Digital Humanities - World Sanskrit Conference 2025*, pages 131–147, Kathmandu, Nepal, June. Association for Computational Lingustics.

Amba Kulkarni and K. V. Ramkrishnamacharyulu. 2013. Parsing sanskrit texts: Some relation specific issues. In Malhar Kulkarni, editor, *Proceedings of the Fifth International Sanskrit Computational Linguistics Symposium*. D. K. Printworld(P) Ltd.

Amba Kulkarni, Sheetal Pokar, and Devanand Shukl. 2010. Designing a constraint based parser for sanskrit. In Girish Nath Jha, editor, *Sanskrit Computational Linguistics*, pages 70–90, Berlin, Heidelberg. Springer Berlin Heidelberg.

Amba Kulkarni, Sanal Vikram, and Sriram K. 2019. Dependency parser for Sanskrit verses. In Pawan Goyal, editor, *Proceedings of the 6th International Sanskrit Computational Linguistics Symposium*, pages 14–27, IIT Kharagpur, India, October. Association for Computational Linguistics.

Amba Kulkarni, Pavankumar Satuluri, Sanjeev Panchal, Malay Maity, and Amruta Malvade. 2020. Dependency relations for Sanskrit parsing and treebank. In Kilian Evang, Laura Kallmeyer, Rafael Ehren, Simon Petitjean, Esther Seyffarth, and Djamé Seddah, editors, *Proceedings of the 19th International Workshop on Treebanks and Linguistic Theories*, pages 135–150, Düsseldorf, Germany, October. Association for Computational Linguistics.

Amba Kulkarni. 2013. A deterministic dependency parser with dynamic programming for Sanskrit. In Eva Hajičová, Kim Gerdes, and Leo Wanner, editors, *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*, pages 157–166, Prague, Czech Republic, August. Charles University in Prague, Matfyzpress, Prague, Czech Republic.

Amba Kulkarni. 2019. *Sanskrit Parsing based on the theories of Śābdabodha*. IIAS, Shimla and D K Printworld.

Amba Kulkarni. 2021. Sanskrit parsing following indian theories of verbal cognition. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 20(2), April.

Anil Kumar, Amba Kulkarni, and Nakka Shailaj. 2024. START: Sanskrit teaching; annotation; and research tool – bridging tradition and technology in scholarly exploration. In Arnab Bhattacharya, editor, *Proceedings of the 7th International Sanskrit Computational Linguistics Symposium*, pages 113–124, Auroville, Puducherry, India, February. Association for Computational Linguistics.

Sebastian Nehrdich, Oliver Hellwig, and Kurt Keutzer. 2024. One model is all you need: ByT5-Sanskrit, a unified model for Sanskrit NLP tasks. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13742–13751, Miami, Florida, USA, November. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), January.

K. V. Ramkrishnamacharyulu. 2009. Annotating sanskrit texts based on sabdabodha systems. In Amba Kulkarni and Gérard Huet, editors, *Proceedings of the Third International Sanskrit Computational Linguistics Symposium*, pages 26–39, Hyderabad, India, 01. Springer-Verlag.

Jivnesh Sandhan, Laxmidhar Behera, and Pawan Goyal. 2023. Systematic investigation of strategies tailored for low-resource settings for low-resource dependency parsing. In Andreas Vlachos and Isabelle Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2164–2171, Dubrovnik, Croatia, May. Association for Computational Linguistics.

Sven Sellmer and Oliver Hellwig. 2024. The Vedic compound dataset. In Archna Bhatia, Gosse Bouma, A. Seza Doğruöz, Kilian Evang, Marcos Garcia, Voula Giouli, Lifeng Han, Joakim Nivre, and Alexandre Rademaker, editors, *Proceedings of the Joint Workshop on Multiword Expressions and Universal Dependencies (MWE-UD) @ LREC-COLING 2024*, pages 50–55, Torino, Italia, May. ELRA and ICCL.

Abhiram Vinjamuri and Weiwei Sun. 2025. Transfer learning for dependency parsing of Vedic Sanskrit. In Chen Zhang, Emily Allaway, Hua Shen, Lesly Miculicich, Yinqiao Li, Meryem M'hamdi, Peerat Limkonchotiwat, Richard He Bai, Santosh T.y.s.s., Sophia Simeng Han, Surendrabikram Thapa, and Wiem Ben Rim, editors, *Proceedings of the 9th Widening NLP Workshop*, pages 50–55, Suzhou, China, November. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online, June. Association for Computational Linguistics.