Compound Type Identification in Sanskrit

Sriram Krishnan[1], Pavankumar Satuluri[2], Amruta Barbadikar[1], T S Prasanna Venkatesh[3] and Amba Kulkarni[1]
[1] Department of Sanskrit Studies, University of Hyderabad
[2] IIT Roorkie [3] Vivekananda College, University of Madras
sriramk8@gmail.com, apksh.uoh@nic.in

Abstract

Compounds are formed based on the principle of *sāmarthya* (semantic compatibility to compose components). Sanskrit compounds are binary compositions which can be composed further to form a nested constituency of compound structure. Analysing a compound requires us to find the correct structure as well the compound types. Compound analysis has long been approached in two stages: constituency analysis and compound type identification. In this work, we propose a methodology to analyse the compounds by first identifying the compound types based on morphological, syntactic and semantic clues from *Aṣṭādhyāyī*. We implement it within the dependency parser framework of *Saṃsādhanī*, where the types are identified first and then the constituency structure is established using a constraint solver.

1 Introduction

The following footnote without marker is needed for the camera-ready version of the paper.

One of the most distinguishing features of Sanskrit is its profound use of compounds (*samāsa*), which allows concise expression of complex ideas. Compounds are formed by combining two or more words into a single lexical unit, often omitting case markers and other grammatical indicators, thus creating a compact structure to convey meaning. These are categorized into four major types on the basis of the semantic relationships between their components. The description of these semantic relations is found in *Aṣṭādhyāyī*, the seminal grammatical text of Sanskrit. The four major types of compounds are *Avyayībhāva*, *Tatpuruṣa*, *Bahuvrīhi* and *Dvandva*. These compound types have further sub-types defined based on both the syntactic and semantic nature of their components. In total, 59 sub-types of these four major types have been observed in Kumar (2012).

Aṣṭādhyāyī meticulously codifies the process of compound generation, providing precise rules for their semantic classification and syntactic construction. The semantic and syntactic compatibility between the components of a compound, referred to as '*Sāmarthya*', ensures that the words forming a compound are meaningfully connected and contribute to a unified sense. The category of the compound-word thus formed can be determined based on its meaning and the grammatical rules that specify the syntactic arrangement of the component words, as well as the case, gender, and number of the final compound. The procedure for generating compounds is clearly outlined, ensuring uniformity in their formation.

When it comes to the analysis of a compound word, there are several challenges due to the inherent complexity of their structure. The first problem we face in analyzing a compound is the segmentation. The compound in Sanskrit is always written as a single word without any space or hyphen between its constituents. Further, the close proximities of the phonemes of two consecutive components also lead to phonological changes in them introducing ambiguities during splitting, thereby worsening the situation further for analysis.

The second problem is the ambiguity at the constituency level when the number of components is more than two. When there are more than two components in a compound, the number of ways in which a compound with $n+1$ components can be subgrouped is a Catalan number C_n (Huet, 2009), which is defined as:

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

The Catalan number for a 4-component compound is 5. This indicates that there are 5 distinct ways to form valid nested structures for a 4-component compound. For example, in a compound 'a-b-c-d', the five possible constituencies are:

$$(1) << ab > c > d \quad (2) < a < bc >> d \quad (3) < ab >< cd > \quad (4) a << bc > d > \quad (5) a < b < cd >>$$

In Sanskrit we observe such lengthy compounds in ample amount. Since C_n grows exponentially, determining the accurate constituency for multi-component compounds throws a challenge.

The last problem is to identify the semantic relation between the components, i.e. the compound type identification. A primary difficulty arises from the ambiguity in interpretation of the relation, as a single compound can often have multiple plausible meanings. For instance, the compound *śivapriyaḥ* could mean "dear to Śiva" (*śivasya priyaḥ*) or "one who loves Śiva" (*śivaṁ priyaḥ yasya*), depending on the context. Without sufficient contextual information, determining the precise compound type becomes challenging.

In this paper, we present an approach to handle the task of compound type identification. Section 2 summarizes the efforts towards developing Sanskrit compound type identifiers and towards the end provides details about the approach followed in this paper. Section 3 discusses how the dependency parsed structure differs from the constituency parsed structure with respect to compound analysis and also provides the reasons why we shift towards the dependency-based approach. Section 4 provides details on how the various lexical, syntactic and morphological clues obtained from *Aṣṭādhyāyī* and other resources, have been incorporated in our type identifier, along with the justification for considering the *dvandva* analysis separately. Our implementation of the compound type identifier is provided in section 5 and the observations on the evaluation of the type identifier is in section 6. Finally, section 7 discusses the issues and possible future directions.

2 Existing Approaches

Recent advancements in automatic compound type analysis for Sanskrit have led to the development of various methodologies aimed at addressing the complexity of Sanskrit compounds. This section highlights some of the notable works in this domain, discussing their methodologies and the type of analysis employed.

Kumar (2012) employed a combination of Pāṇinian grammatical rules and statistical methods for compound type identification. As part of this approach, a statistical constituency parser was developed to generate constituency structures, which serve as input for the type identification module. This module integrates both rule-based and statistical methods to classify compounds. However, the model is limited to handle only two-component compounds and struggles with more complex types, such as *dvandva* compounds. For the statistical constituency parser and the compound type identification tasks, the SHMT dataset was used.[1] This dataset contains compounds from various texts like *Bhagavad Gītā, Caraka-saṃhitā, Pañca-tantra*, etc. that were annotated with the constituency and compound type information.

Satuluri (2015) discusses various semantic, ontological and other information needed for the generation of compounds. This study and the observations there in are very much useful from the analysis point of view as well.

[1]Sanskrit Hindi Machine Translation Consortium Dataset developed under the funding from DeiTY (Department of Electronics and Information Technology), Govt. of India (2008-12), available at `https://sanskrit.uohyd.ac.in/scl/GOLD_DATA/tagged_data.html`.

Krishna et al. (2016) proposed a classification framework for the compound type identification task. It combines features extracted from rules in *Aṣṭādhyāyī*, taxonomy information and semantic relations inferred from *Amarakośa* ontology (Nair and Kulkarni, 2010), and linguistic structural information from the data using Adaptor grammar (Johnson et al., 2006). The rules from *Aṣṭādhyāyī* were divided into four types: rules with lexical lists, rules with morphological properties, rules with semantic properties of the components, rules with semantic relations between the components. The ontological relations between various words from *Amarakośa* were extracted. The SHMT dataset (32,000 compounds) was used for the classification task.

Sandhan et al. (2019) proposes a neural approach that classifies compounds without considering broader context. It employs deep learning techniques, utilizing features extracted from Sanskrit text. The model, trained on annotated corpora, relies on sequence-based architectures such as recurrent neural networks (RNNs) and transformers. By focusing on the internal structure of compounds and word embeddings, this approach offers scalability and adaptability to large datasets. However, it faces challenges in disambiguation, particularly when contextual understanding is essential. The same SHMT dataset as above was used for this task.

Sandhan et al. (2022) proposed a hybrid approach (SACTI), integrating rule-based linguistic features with deep learning techniques to improve compound classification. The system combines handcrafted linguistic features—such as compound segmentation and phonetic similarity—with a novel multi-task learning architecture to enhance accuracy. Additionally, it partially incorporates constituency-based analysis to capture hierarchical relationships within compounds, improving the identification of compositional structures. This approach conducted their experiments on the 4 coarse-grained types and 15 fine-grained types, restricting to binary compounds (upto 48,132 compounds) collected from the SHMT dataset. This dataset was revised additionally to include the context.

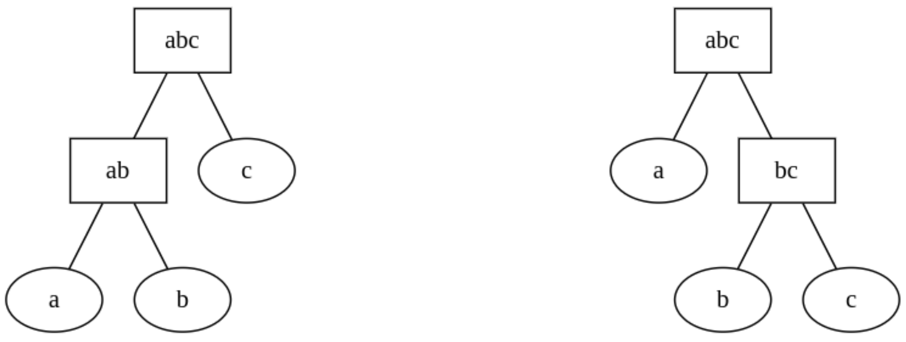
'DepNeCTI' (Sandhan et al., 2023) employs a dependency-based framework to analyze compound structures, incorporating syntactic dependency parsing alongside neural classifiers. Its dependency-based approach is particularly efficient in representing grammatical relationships, making it well-suited for analyzing the structural dependencies of Sanskrit compounds. A novel approach towards Nested Compound Type Identification was introduced focusing on identifying nested spans within a multi-component compound and interpreting their implicit semantic relationships. While the same SHMT dataset was used here, the primary focus was on compounds with more than two components, and on binary compounds with context. The dataset contains for each of the compounds, the segmented components, nested spans, context and semantic relations among the nested spans. Here 86 fine grained compound types were used along with the usual 4 broad types. A total of 17,656 compounds with context from philosophical, literary and *Āyurvedic* texts were collected for the task, and 1,189 compounds from *Purāṇas* were considered for an out of vocabulary dataset.

Our research approach integrates some of the elements from the aforementioned methods. We primarily rely on *Aṣṭādhyāyī* rules as the foundational framework, supplementing them with a supportive database and linguistic heuristics wherever necessary. Our goal is to handle all types of multi-component compounds, including *dvandva* compounds. We propose a compound type identifier which is integrated with the dependency parser of *Saṃsādhanī.* The type identifier is implemented to handle all the types except *dvandva* for which a standalone *dvandva* analysis module is incorporated. We prioritize dependency parsed structure over the constituency for various reasons, which are discussed in detail in the next section.

3 Dependency-based Compound Analysis

Sanskrit compounds exhibit intricate internal structures where multiple components combine to form complex words. The underlying principle to form compounds is the semantic compatibility between components, called *sāmarthya.* The sense in which the composition happens across two semantically compatible components is expressed through the semantic relations or compound types. While compound formations are predominantly binary compositions (except for *dvandva* and *bahupada-bahuvrīhi*), due to the productive nature of compounds, we can build a nested structure of compounds by successive binary composition of the components with these relations. The semantic relations are thus crucial to identify the correct nested structure (Sandhan et al., 2023).

Analyzing these structures requires a systematic approach to identify relationships among components. Two primary frameworks—constituency and dependency—offer different perspectives on compound analysis. In multi-component compounds, there can be more the one possible spans according to various combinations of intermediate nested compounds. Figure 1 shows two possible spans for a three-component compound in constituency analysis.



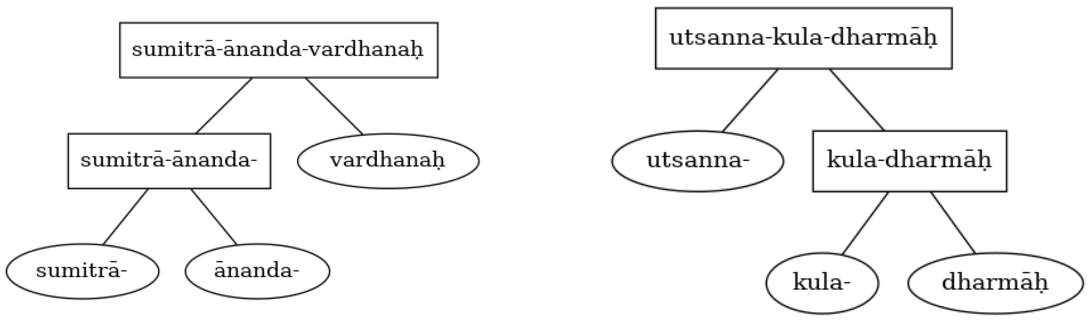
(a) < < a - b > - c >

(b) < a - < b - c > >

Figure 1: Possible constituency spans for a three-component compound a-b-c

3.1 Constituency Parsed Structure

In this approach, compounds are analyzed by constructing a hierarchical structure that represents their internal syntactic composition. Each constituent (sub-compound) is represented as an intermediate node, capturing the nested relationships within the compound. However, this method introduces additional nodes that may not be necessary for compound type identification, increasing computational complexity. For example, in figure 2a, the compound *sumitrā-ānanda-vardhanaḥ* (one who increases the happiness of Sumitrā) is represented with a noun phrase structure (having a left associative parse), introducing hierarchical levels that may not contribute directly to type identification. Similarly, figure 2b shows the structure of the right-associative three-component compound '*utsanna-kula-dharmāḥ*' (deprived of ancestral duties and traditions).



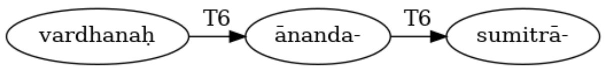
(a) Span 1: < <sumitrā-ānanda>T6-vardhanaḥ>T6

(b) Span 2: <utsanna-<kula-dharmāḥ>Bs3>T6

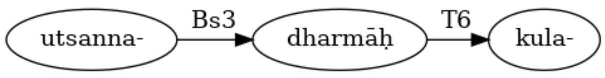
Figure 2: Possible spans for Constituency analysis with examples

3.2 Dependency Parsed Structure

Dependency-based analysis, in contrast, establishes direct relationships between components without the need for additional intermediate nodes. Each component of the compound is linked to the head based on grammatical / syntactic / semantic relations, making it easier to mark the type of compound based on dependency-like relations. For instance, in figure 3b, for the compound '*utsanna-kula-dharmāḥ*' (deprived of ancestral duties and traditions), the components *'dharma'* and *'kula'* are directly linked to their heads, and the relationship can be labeled according to its semantic function. Also, figure 3a shows the left associative span. The directed labels (T6, Bs3, etc.)[2] help us in identifying the heads of the intermediate and the external compounds.



(a) Span 1: < <sumitrā-ānanda>T6-vardhanaḥ>T6

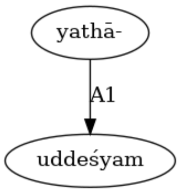


(b) Span 2: <utsanna-<kula-dharmāḥ>Bs3>T6

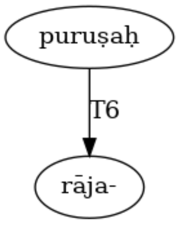
Figure 3: Dependency analysis of compounds with examples

3.3 Equivalence between Constituencey and Dependency Parsed Structures

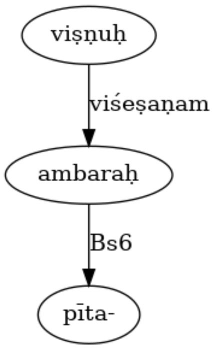
Typically, it is observed that the *avyayībhāva* compounds are left-headed (*pūrvapada-pradhāna*).[3] *tatpuruṣa* compounds are right-headed (*uttarapada-pradhāna*) and in the case of *bahuvrīhi*, none of the components is the head. And finally, in the case of *dvandva* compounds, both (all) components are of prime importance. So in dependency structure, we mark the relations between these components as in figure 4. The *dvandva* compounds are represented as a plain string joining all the components such as *rāma_lakṣmaṇau*, with the underscore (_), since both of them have equal status.[4]



(a) <yathā-uddeśyam>A1



(b) <rāja-puruṣaḥ>T6



(c) <pīta-ambaraḥ>Bs6

Figure 4: Dependency-based representations of compound types with the arrows marking the head.

[2] T6 stands for *ṣaṣṭhī-tatpuruṣa* and Bs3 stands for *tṛtīyārtha-bahuvrīhi*. The list of all the labels (compound types) along with the tagging guidelines are available at: `https://sanskrit.uohyd.ac.in/scl/GOLD_DATA/Tagging_Guidelines/samaasa_tagging16mar12-modified.pdf`.

[3] Here *pūrvapada* refers to *pūrvapada-artha*. Similarly for *uttarapada* and *anyapada*.

[4] We do not prove the equivalence between the constituency and dependency structures here, since it is outside the scope of this paper. A separate manuscript is getting ready proving the equivalence of these structures and how to convert one to the other automatically.

The advantage of the dependency representation is that intermediate nodes are no more needed. That is, there will be exactly as many nodes as there are components. Hence one can use the same algorithm of the dependency parser to parse compound structures as well. The second advantage is that one can now use the morphological and other constraints specified in the grammar rules of compound formation to identify the potential components that can be related and the possible compound types. Then use a constraint solver to prune out the types which do not satisfy the *yogyatā* and *sannidhi*. Thus, the *sāmarthya* between components can be established from the compound types which were obtained using the principles of *ākāṅkṣā, yogyatā and sannidhi*. Finally, a ranking algorithm should be incorporated to rank the various constituencies obtained. In this paper, we confine ourselves to only the *ākāṅkṣā* module.

4 Clues for Dependency-based Compound Analysis

Pāṇini's Aṣṭādhyāyī provides both semantic and syntactic scenarios for the construction of compounds from words. These scenarios can be considered as clues or constraints during the analysis of a compound. The syntactic clues pertain to the order of the components, deletion of case, assignment of *svara*, etc. This also includes the nature of syntactic combinations of the components, primarily attributed to the rule *sup supā (2.1.4)* with later grammarians introducing the possibilities of other syntactic combinations like (*sup-tiṅ, sup-nāma, sup-dhātu*, etc.).[5] On the other hand, a majority of the semantic clues are available across the first two parts of the second *adhyāya* of *Aṣṭādhyāyī*. Kumar (2012) extracted all the possible semantic clues from these rules and elaborated them in detail providing in each case whether the clues can be incorporated for the task of compound type identification or not.

Kumar (2012) developed first a statistical constituency parser which generates the constituency structure. This structure is fed to the compound type identification module where both the rule-based and statistical identifiers help in predicting the type of the compound. In the present work, since both constituency analysis and compound type identification are addressed simultaneously in the same step, this calls for a revised set of clues that addresses both these tasks. Thus we first revisit all these clues extracted from *Aṣṭādhyāyī* and perform a validation check with various examples along with observations on possible clues for constituency analysis as well. In this process, the clues are categorized into groups similar to Krishna et al. (2016) where the categorization was based on the syntactic and semantic nature of the clues. In our approach, the clues are reordered in such a way that the syntactically similar clues are put together in a group. And within each group, the clues which have more precise information are addressed first and those with more generic information are pushed to the last.

Our approach differs from Kumar (2012) in two aspects:

1. The constituency structure in Kumar (2012) was established based on co-occurrences of the components to figure out the affinity of a component to be composable with another component. The compound types were identified only for the compounds observed in the constituency structure. In our case, we establish all possible compound types between each pair of the components, and then using constraints prune out the incompatible types. In this process, the dependency structure is constructed.

2. The clues from *Aṣṭādhyāyī* were categorized according to the compound types in Kumar (2012). These are now re-organized into categories based on both syntactic and semantic nature of the clues.

4.1 Revisiting the clues from *Aṣṭādhyāyī*

Kumar (2012) proposed 55 semantic clues corresponding to each *sūtra* from *Aṣṭādhyāyī*. Clues from 21 *sūtras* were not implemented as extra semantic information was required. And it was

[5] *supāṃ supā tiṅā nāmnā dhātunā'tha tiṅāṃ tiṅā*
subanteneti vijñeyaḥ samāsaḥ ṣaḍvidho budhaḥ – *Kārikā* 28 of *Vaiyākaraṇabhūṣaṇasāra*

noted that for 7 rules, an implementation was not possible at all for various other reasons. We took all of these cases and prepared a revised set of 88 conditions from the 83 rules of Kumar (2012).

The clues are first divided into two types based on the number of constituents: two or three. Compound formations are typically binary with some exceptions like *itaretara-dvandva* and *bahupada-bahuvrīhi.* But some of the clues give specific information pertaining to both constituency and the compound type, for certain compounds with three components. 7 of the 88 conditions require three components and these are addressed together in a group.[6] The clues pertaining to binary combinations are divided into two: those containing semantic conditions (76), and conditions for exceptional compounds (5). The remaining 76 conditions are further divided into groups based on either the morphosyntactic nature of their constituents or the type of the compound:

1. *ktānta* (where either the first or the second or both components have the *kta* suffix),
2. *non-ktānta tatpuruṣa* (where neither component has the *kta* suffix),
3. *karmadhāraya* (which are not covered in the first two cases),
4. *avyayībhāva*,
5. *dvigu-tatpuruṣa*,
6. *residual tatpuruṣa*,
7. *bahuvrīhi*,
8. *nañ-tatpuruṣa*, and
9. *ṣaṣṭhī-tatpuruṣa*

Within each of the groups, the clues are ordered in such a way that the more precise information is checked first and the less precise information later. Clues which prescribe a specific set of words in either iic (*in initio composite*) or ifc (*in fini composite*) or both are given higher preference. Then the syntactic and morphological clues like *kta, yat*, gender, number, *avyaya*, etc. This is followed by semantically and ontologically tagged lexicon grouped into various lists such as *kālavācī, varṇavācī, nadīvācī, jātivācaka*, etc. These are collected as and when an example is encountered. Finally, the rest of the conditions follow. The morphosyntactic clues deploy the *ākāṅkṣā* constraint and help in establishing the possible compound types. *Yogyatā* has not yet been extensively dealt with but the above semantically tagged word lists denoting a particular entity or concept, provide some help in determining whether *yogyatā* is present or not. Thus, the first level of establishing *sāmarthya* is done using these constraints and to disambiguate between the observed types, we need a dedicated constraint solver in the second level.

4.2 Clues from the Heritage Segmenter

Sanskrit Heritage Segmenter (SH)[7] is a lexicon-directed segmentation engine that produces the segmentation along with the morphological analysis and the part of speech category for each of the possible segments produced. The part of speech categories, called *phases*, follow closely the *Pāṇinian* system of derivation and inflection mechanisms. For instance, according to *Aṣṭādhyāyī*,[8] a word is either a *subanta* (noun) or a *tiṅanta* (verb). *Subantas* are *prātipadikas* (stems) inflected with *sup* suffixes and *tiṅantas* are *dhātus* inflected with the *tiṅ* suffixes. The phases are addressed based on the inflectional and derivational suffixes used, type of the stem

[6]All the groups are presented in a tabular format in Appendix A.
[7]`https://sanskrit.inria.fr`
[8]*suptiṅtam padam* 1.4.14

or root (noun, pronoun, verb), compound components, preverbs, etc. (Huet, 2024). The phases are represented using various colors in the graphical interface. For example, deep sky blue for nouns, light blue for pronouns, red for verbs, mauve for indeclinables, yellow for iics, etc. Out of the 54 phases constructed in SH, 15 are of interest for the current task of compound analysis and are enlisted in Table 1.

Phase	Type	Color	Example
Iic	first part of compounds	yellow	*rāma*-alayaḥ
Iiv, Iivc, Iivv	inchoatives (cvi verbal compounds)	orange	*śuklī*-karoti
Iiif	ifc of iic	kaki	kumbha-*kāra*-putraḥ
Ifc	second part of compounds[1]	cyan	kumbha-*kāraḥ*
Indifc	indeclinable forms usable as ifcs	mauve	bhīṣma-droṇa-*pramukhataḥ*
A, An	privative *nañ*-compound formations	yellow	*a*-prāpya
Ai, Ani	initial privative *nañ*-compounds	yellow	*a*-kīrti-karam
Iik, Iikc, Iikv	*kṛdanta* iics	yellow	*kaṣṭa*-śritaḥ
Iiy	*avyayībhāvas*	pink	*upa*-kṛṣṇam

Table 1: Phases of Sanskrit Heritage Segmenter usable for compound analysis

The following inferences can be observed from the phases provided by SH:

1. The *avyayībhāva* iics can be detected using the phase Iiy.
2. Given a compound, we can detect whether it can have a *bahuvurīhi* interpretation using the phase Ifc. But we would require the context to confirm if it is *bahuvrīhi* or not. Generally, the final part of the compounds are inflected with the sup-suffix and would come under the regular Noun phase. The phase (Ifc) corresponds to specific cases of inflections on the final components when compounds like *pīta-ambaraḥ, kumbha-kāraḥ* are formed. *ambaraḥ* is not a valid form of the neuter stem *ambara*, but the gender is inherited from the surrounding noun phrase head. Generated stems like *-kāra* are restricted to the role of ifcs. In case of compounds like *pīta-ambaram*, the machine will mark three possible relations viz. *tatpuruṣa* or a *bahuvrīhi* reading with *ambaram* as neuter in either nominative or accusative case, and a *bahuvrīhi* reading with *ambaram* as masculine accusative. The presence of a substantive with the same case-gender-number information will propose the *bahuvrīhi* solution with higher confidence. The dependency parser will rank these solutions considering the complete sentence.
3. In a multi-component compound setup, the phase Iiif helps in addressing the constituency of the compound.
4. With the phase Iiv, the cvi-compounds can be detected.
5. Ai and Ani help in disambiguating whether the initial compound is a privative or not in a multi-component compound setup with the initial component being the negative compound particle "a" or "an".

Additionally, the privatives are mostly lexicalized except when they form absolutive compounds. And some of the *aluk* compounds (like *pātresamita*) and compounds with retroflexion effects (for eg. *dūrvāvaṇa*) are also lexicalized in SH.

4.3 Analysing *Dvandva* Compounds

The *dvandva* (or copulative) is a special type of compound, where all the components compose together to denote a collection. This is an exception to the general notion of binary composition during compound formation. In *Aṣṭādhyāyī*, *dvandva* compounds are defined by the rule *cārthe dvandvaḥ*.[9] *Dvandva* can be classified into two types: *itaretara* and *samāhāra*.

1. **itaretara:** when a conjunction of mutually compatible entities is intended (*plakṣaśca nyagrodhaśca plakṣa-nyagrodhau*),[10]

2. **samāhāra:** when an aggregation of similar entities is intended (*sañjñā ca paribhāṣā ca tayoḥ samāhāraḥ sañjñā-paribhāṣam*).[11]

In the case of *itaretara-dvandva*, all the components are considered equally important and they are considered individually. The number of the overall compound depends on the number of entities present and the gender depends on the gender of the last component. For example, *rāma-lakṣmaṇa-bharata-śatrughnāḥ*. In the case of *samāhāra-dvandva*, the components are considered as a group and not individually. The overall gender of the compound is in neuter. And the number is singular. For example, *pāṇi-pādam*. There is another type of *dvandva* called *ekaśeṣa-dvandva* compounds which are special cases where only one of the components will remain in the final compound, with gender and number addressed similarly to *itaretara*. For example, *pitarau (māta ca pitā ca)*. Some grammarians consider *ekaśeṣa* as a separate *vṛtti*,[12] and not under the purview of *samāsa*.

We observe that morphological analysis, specifically the gender and number of the overall compound, helps in detecting a *dvandva* compound. Sometimes, the final component could have certain indicators like the stem *ādi*. For example *kāka-kūrma-ādīnām*. With such a stem, the set of previous entities can be considered as a *dvandva* compound. Additionally, the *sāmarthya* in the *dvandva* components lies with their similar ontological structure:

kāka $\rightarrow$ *(padārtha, dravyam, pṛthvī, calasajīva, manuṣyetara, jantu, pakṣī)*[13]
kūrma $\rightarrow$ *(padārtha, dravyam, pṛthvī, calasajīva, manuṣyetara, jantu, ubhayacara)*

Both have common parent nodes until *jantu* and thus have a higher tendency of mutual compatibility to become *dvandva*. For this, we need a semantically and ontologically annotated data with more specific features to handle *dvandva* compounds.

Morphological clues do help to an extent, but with this difficulty in incorporating ontological clues for identifying the *sāmarthya* for *dvandva* between the components, an alternate approach is required to address *dvandva* compounds. We can thus build sets of words where each set contains stems which have the *sāmarthya* to form *dvandva* compounds with any other stem in the same set. We create two such lists: (1) sets of words where the order needs to be preserved (frozen compounds or *nitya-samāsas*) and (2) sets of words where order does not matter. Thus stems from a similar domain are collected together from a list of *dvandva* compounds extracted from the SHMT dataset.

Finally, *dvandva* analysis has to be done separately and prior to the analysis of other compound types because of three reasons:

- The *n*-ary nature of *dvandva* as opposed to the binary,

- The approach involves comparison with a collection of stems in a particular domain and does not cater to either the morphological or syntactic clues, and thus cannot be integrated with the clues mentioned earlier for other compounds,

- In a nested compound structure, *dvandva* compounds can be found more frequently in the inner nested structure, and the *dvandva* composition of other compounds like *tatpuruṣa, bahuvrīhi* is very rare in usage.

[9] *Aṣṭādhyāyī* 2.2.29
[10] *militānām* anvayaḥ
[11] *samūhaḥ*
[12] *kṛt-taddhita-samāsa-ekaśeṣa-sanādyanta* are the five *vṛttis*.
[13] Extracted from the *Amarakośa* knowledge web: `https://sanskrit.uohyd.ac.in/scl/amarakosha/index.html`

5 Implementation of the Compound Analyser

In the present work, as the compound analysis task is integrated with the dependency analysis environment, it adheres to the same procedure deployed in the dependency parser with the following modifications:

1. Segmentation (with compound boundaries marked)

2. Morphological analysis for all words and compound components

3. Dvandva Analysis

4. Identifying the relations

5. Constraint solver (partially implemented)

The segmentation is obtained from SH, which also marks the compound boundaries. In the current setup, phase-level clues from SH are not taken into account. The segmented sentence is then passed to *Saṃsādhanī's* morphological analysis engine that produces all possible morphological analysis for each of the segments. It marks all the iics as *samāsa-pūrva-pada*.

Generally, the words with the possible morphological analyses are sent to the dependency parser, which builds a set of dependency relations between every word, based on the *śābdabodha* theories. But we insert the *dvandva* analysis module here which takes in the words and their corresponding morphological analysis and whenever a series of components match any of the existing *dvandva* compound components lists collected earlier, these components are merged together (with an *underscore*) to represent a single entity. This resolves the representation issues of the *dvandva* compounds and also makes sure that in the next stage, the constraint solver does not overgenerate the relations with the components of the *dvandva* compounds. The results of the *dvandva* module are passed onto the parser. The clues for detecting and analysing compound types and the constituency are integrated into this parsing engine which involves picking up the components based on their syntactic category followed by running through the clues to identify the possible compound types. The graph marking the relations between various components is generated as follows.

1. For every triplet of compound components, it runs through the clues for deciding the constituency and identifying the compound type and assigns relations wherever possible,

2. For every pair of compound components:

   (a) if the first component is *a* or *an*, then it assigns *nañ-tatpuruṣa*,
   (b) the list of exceptional compounds is checked (this contains the special cases like *aluk* compounds, closed and open sets of compounds like *mayūravyaṃsaka*),
   (c) conditions pertaining to the clues are checked in the following order: *avyayībhāva*, *ktānta*, *non-ktānta tatpuruṣa*, *karmadhāraya*, *dvigu-tatpuruṣa*, residual *tatpuruṣa*, *bahuvrīhi* and finally *ṣaṣṭhī-tatpuruṣa*
   (d) if none of the relations is obtained, then *ṣaṣṭhī-tatpuruṣa* is assigned by default

The set of relations is then passed to a constraint solver that resolves various conflicts that occur between the predicted relations. In the current setup, for the constituency analysis, it is assumed by default that the compounds with more than two components have left associativity. The compound types are ranked and simple constraints of proximity and every node having only one incoming arrow are applied.

6 Evaluation

For the present implementation, we collected a list of compounds proposed as examples for each of the *samāsa-vidhāyaka-sūtras* from *Kāśikā* and *Siddhāntakaumudi.* These were predominantly two-component compounds with a few three component compounds. These examples were used as development sets along with the rules. Also, all the compounds of *Bhagavad Gītā* were collected and used for testing. Here we describe the performance of the dependency-based compound type identification module on these two test sets. We elaborate on where the identifier fails and what needs to be done for improving the analyser.

As the compound type identifier is integrated into the dependency parser, it requires a complete sentence so that an overall dependency tree is obtained. But this implementation does not consider context into account. This helps us understand where exactly context plays an important role and for what type of compounds, context is critical. To account for this, each of the examples is provided with an auxiliary verb like (*asti, bhavati, etc.*) depending on the case and number of the overall compound. The ground truth and the results are obtained as JSON objects for ease of comparison.

Metrics: The macro-averaged label score (LS), labeled attachment score (LAS) and unlabeled attachment score (UAS) are considered for evaluating all the compounds. Additionally, the precision recall and f-score values are recorded for each of the coarse compound types.

6.1 Error Analysis on *Aṣṭādhyāyī* examples

311 compounds from *Aṣṭādhyāyī* were considered for evaluation. The relations were predicted correctly for 194 compounds, with the macro-averaged LAS being 62.38%. A manual error analysis was done on the remaining 117 examples and here are the observations. There were five kinds of issues present:

1. **multiple relations:** This is the most common issue where multiple relations are assigned based on several conditions and the constraint solver does not know how to disambiguate between them. 56 examples are affected because of this issue. For example, *go-hitam* is a *caturthī-tatpuruṣa* compound but the possible relations are *ṣaṣṭhī-tatpuruṣa, caturthī-tatpuruṣa, tṛtīyā-tatpuruṣa, karmadhāraya*, but *tṛtīyā-tatpuruṣa* is produced as the final relation. There are two possible solutions to resolve such cases: (1) stricter conditions are to be placed based on observations to disambiguate at the level of type identification, or (2) the constraint solver should contain some measure to rank the possible relations obtained.

2. **no morphological analysis:** This corresponds to 30 examples where *Saṃsādhanī's* morphological analysis fails to analyse one of either iic or ifc and the parser does not produce any result. For example, (*kaṣṭa-samīkṛtam*). One solution is to update the morphological analyser to accept such missing forms.

3. **incorrect morphological analysis:** 19 cases have this issue where the expected morphological analysis is not available in the possible morphological analyses. For example, the compound *sukha-apetaḥ* is a *pañcamī-tatpuruṣaḥ* compound in the ground truth but it is analysed as a *ṣaṣṭhī-tatpuruṣaḥ* compound because *apeta* is not analysed as a *ktānta.* Even here, the morphological analyser can be improved by incorporating such forms.

4. **missing data:** The semantically and ontologically tagged lexicon is of limited size. For example, for the compound *pañca-nadam*, *nadam* was missing in the *nadī_vācī* list. These correspond to 10 examples and there is a need to update such lexicon.

5. **no condition:** In some rare cases, the set of conditions put forth based on the clues are insufficient and we have to bring in more clues for addressing this issue. 2 compounds have been affected by this issue. For example, *uccaiḥ-kṛtya* needs a condition that checks that the ifc has a *lyap* suffix instead of a *kta* suffix. This condition needs to be added.

6.2 *Bhagavad Gītā* Examples

Bhagavad Gītā possesses a huge number of compounds across its 700 verses. We collected a total of 1,580 compounds from the SHMT dataset, averaging two compounds per verse. An analysis was done on the distribution of the compound types and it was observed that there are 2,052 compound formations, including the inner compounds of a nested compound structure. Of these, 526 are *bahuvrīhi* and 349 are *ṣaṣṭhī-tatpuruṣa* resulting in a combined 885 instances.[14] The distribution of the number of components per compound is shown in table 2.

Number of components	**Number of compounds**
2	1,229
3	253
4	84
5	9
6	1
7	4

Table 2: Component-wise distribution of *Bhagavad Gītā* compounds

Out of these, 88 compounds had issues with morphological analysis, i.e., at least one of the components could not be recognized by the morphological analyser. Of the remaining 1,492 compounds, 550 were predicted with the correct compound types with the correct constituency. 756 compounds had a partial match, i.e, at least one of the relations was correctly matched. In the ground truth analysis of the compounds, only the coarse grained annotations were present for *avyayībhāva* and *karmadhāraya* types and the evaluation was done on these types alone. The macro-averaged LS, UAS and LAS scores are recorded in table 3. The confusion matrix for the overall coarse classification is presented in table 4 and the precision, recall and f-score for each of the types is shown in table 5.

	Morph-issue	Compounds	**LS**	**UAS**	**LAS**
2-component compounds	78	1,151	0.43	-	0.43
3 to 7 component compounds	10	341	0.49	0.88	0.45
all compounds	88	1,492	0.41	0.92	0.40

Table 3: Label Scores (LS), Unlabeled Attachment Scores (UAS) and Labeled Attachment Scores (LAS) of *Bhagavad Gītā* compounds

A sample set of unanalysed compounds were considered for error analysis and here are the observations:

- Since the conditions for *bahuvrīhi* compounds in the implementation were only for certain exceptional subtypes of *bahuvrīhi*, most of these went unanalysed. In addition to these exceptional compounds, *bahuvrīhi* was also identified when a change in the (inherent) gender is observed in the final component.

- 41 *dvandva* compounds were not analysed because of two reasons:

  - Some of the *dvandva* compounds had a *bahuvrīhi or tatpuruṣa* compound embedded within.

[14]In the SHMT dataset, the compound types were manually annotated based on the context but in this implementation, we are not considering context while detecting a particular type.

-	None	AB	BV	D	KD	K	TP
None	0	0	0	0	0	0	0
AB[1]	0	17	0	0	0	0	6
BV[2]	37	0	126	2	31	0	329
D[3]	1	0	8	205	12	0	20
KD[4]	1	0	28	0	61	0	121
K[5]	0	0	0	0	0	0	4
TP[6]	49	0	193	12	22	0	765

[1] AB - avyayībhāvaḥ

[2] BV - bahuvrīhiḥ

[3] D - dvandvaḥ

[4] KD - karmadhārayaḥ

[5] K - kevala-samāsaḥ

[6] TP - tatpuruṣaḥ

Table 4: Coarse-level confusion matrix for *Bhagavad Gītā* compounds

	Precision	Recall	F1-score	Support
avyayībhāvaḥ	1.00	0.74	0.85	23
bahuvrīhiḥ	0.35	0.24	0.29	526
dvandvaḥ	0.93	0.83	0.88	246
karmadhārayaḥ	0.48	0.29	0.36	212
kevala-samāsaḥ	0.00	0.00	0.00	4
tatpuruṣaḥ	0.61	0.73	0.67	1041
micro avg	0.60	0.57	0.58	2052
macro avg	0.56	0.47	0.51	2052
weighted avg	0.57	0.57	0.56	2052

Table 5: Coarse-level Precision, Recall and F-Score for *Bhagavad Gītā* compounds

  – In the case of multiple morphological analyses for a component, the first stem was considered.[15]

- Similar to the *Aṣṭādhyāyī* examples, the analyser failed to disambiguate between various predicted types due to the lack of a good constraint solver. The conflict mainly lies between *tṛtīyā-tatpuruṣa, ṣaṣṭhī-tatpuruṣa* and *karmadhāraya.*

- For the *karmadhāraya* compounds, we need a good lexicon with their ontological categories such as *jāti, guṇa, dravya,* etc.

- In some cases, the gold annotations were found to be incorrect, especially for nested compounds. For example, *an-eka-janma-saṃsiddaḥ* should have been annotated as:
  < < < an - eka > Tn > - janma > K - saṃsiddhaḥ > T7,
  but is annotated in the gold as:
  < < < an - eka > K > - janma > K - saṃsiddhaḥ > T7.

[15]This is mainly to avoid over-generation of the *dvandva* possibilities. For all other compound types, all possible morphological analyses were considered.

6.3 Evaluation of SH results

Based on the clues from SH discussed earlier, it can be observed that *avyayībhāva* and *bahuvrīhi* compounds can be detected using the phases of SH. In order to check the performance, a list of *avyayībhāva* (136) and *bahuvrīhi* (2,126) compounds was collected from the SHMT dataset. Each of the compounds was run on the SH engine to produce the segments along with their phase and morphological analyses. The segments and the phases were compared with the ground truth segments and phases and the observations are recorded in table 6. In addition to this test set, the *avyayībhāva* (21) and *bahuvrīhi* (18) compounds from the *Aṣṭādhyāyī* examples were also run on the SH engine and the observations are recorded.

	SHMT		*Aṣṭādhyāyī*	
	Avyayībhāva	*Bahuvrīhi*	*Avyayībhāva*	*Bahuvrīhi*
Total Compounds	136	2,126	21	18
Unrecognized	14	213	4	3
Correct Segmentation	83	1,358	11	11
Correct Phase	58	637	8	5
Incorrect Segmentation	39	555	6	4
Incorrect Phase	25	721	3	6

Table 6: Performance of SH on *Avyayībhāva* and *Bahuvrīhi* compounds from SHMT dataset and *Aṣṭādhyāyī* examples

Considering the 25 *avyayībhāva* compounds from SHMT dataset, where SH was able to segment it correctly but couldn't detect it as an *avyayībhāva*, we observed that some of the compounds with *yathā* or *prati* in the iic are misjudged. Some of the other examples are *nānā-vidham, sama-akṣam, daśa-ānanaḥ, madhya-yātram*, etc. Considering the *Aṣṭādhyāyī* examples, the compounds *dvi-muni, sapta-gaṅgam, pañca-nadam* were segmented correctly but not detected as *avyayībhāva*, suggesting that the *saṅkhyāpūrva-nadyuttarapada* and *saṅkhyāpūrva-vaṃśyottarapada* types of *avyayībhāva* compounds are not detected by SH.

Considering the 8 *bahuvrīhi* compounds from the *Aṣṭādhyāyī* examples where SH could segment it correctly but not decide whether it can be *bahuvrīhi* or not, we observed that the compounds having words indicating directions in both the components (*digvācaka-bahuvrīhi*) are not detected. For example, *dakṣiṇa-pūrvā, pūrva-uttarā*, etc. although the gender of the ifc has been identified as feminine correctly. And another set of compounds which have numerals in either of the components are not detected. For example, *tricaturāḥ, āsanna-viṃśāḥ, adhika-viṃśāḥ*, etc. An error analysis on the SHMT compounds is still pending.

7 Inferences and Discussion

The present work is an attempt to build a rule-based compound analyser, where instead of doing constituency analysis first followed by the type identification we identify all possible types of compounds between the components and then apply the constraints to get the possible analyses that are finally ranked based on the ranks of various types. These ranks are obtained based on the involved ontological constraints and also the statistics of the annotated tagged data available.

Contextual Analysis: The current implementation does not consider the context at all. Hence many *bahuvrīhi* compounds were not detected at all, as they are required to be in a modifier-modified (*viśeṣaṇa-viśeṣya*) relation with another word in the same context, while the sentences we gave as an input had a single compound with a verb. With the help of contextual analysis, we will be able to define possible relations between the overall compound and other words in the sentence. For example, the *avyayībhāva* compounds are predominantly *kriyāviśeṣaṇa* for their corresponding verbs in the sentence, with a few exceptions like *pare-gaṅgāt, madhye-gaṅgam*, etc. where the corresponding *vibhakti* can be used for assigning the dependency relation with the verb. For mostly all other compounds the depencency relation is assigned based on the morphological clues from the last component of the compound. In the present context we have not yet implemented the *asamartha* compounds and thus cases such as *śāpena astaṅgamitamahimā* would not be analysed properly.

Constraints: The evaluation results showed that majority of the compounds had multiple types predicted and a constraint solver is required to disambiguate them. The inferences from the examples of *Bhagavad Gītā* observed in section 6.2 can be used here for disambiguating mainly between *bahuvrīhi, ṣaṣṭhī-tatpurṣa and karmadhāraya.* More observation is required on various other examples to bring about such inferences. For instance, the *ṣaṣṭhī-tatpurṣa* compounds require a specific *sambandha* like *pitā-putra-sambandha* for the compound *pāṇḍu-putrāṇām.* Such relations can be obtained from the ontology and Named Entity Recognizers. Also, for narrowing down to the expected type, we can consider the predominance of relations based on the occurrences like *ṣaṣṭhī-tatpurṣa* > *bahuvrīhi* > *karmadhāraya.* A constraint solver thus has two tasks: (1) to disambiguate between multiple types for a compound, and (2) to disambiguate between different spans of a nested structure.

References

V. V. Bhandare. 1995. Structural and semantic aspects of the dvandva compound. *Annals of the Bhandarkar Oriental Research Institute*, 76(1/4):89–96.

Sushant Dave, Arun Kumar Singh, Prathosh A. P., and Brejesh Lall. 2021. Neural compound-word (sandhi) generation and splitting in sanskrit language. In *CODS-COMAD 2021: 8th ACM IKDD CODS and 26th COMAD, Virtual Event, Bangalore, India, January 2-4, 2021*, pages 171–177. ACM.

Pawan Goyal and Gérard Huet. 2016. Design and analysis of a lean interface for Sanskrit corpus annotation. *Journal of Linguistic Modeling*, 4(2):117–126.

Gérard Huet and Amba Kulkarni. 2014. Sanskrit linguistics web services. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 48–51.

Gérard Huet. 2003. Lexicon-directed segmentation and tagging of Sanskrit. In *XIIth World Sanskrit Conference, Helsinki, Finland. Final version in Themes and Tasks in Old and Middle Indo-Aryan Linguistics, Eds. Bertil Tikkanen and Heinrich Hettrich.*, pages 307–325, Delhi, August. Motilal Banarsidass.

Gérard Huet. 2009. Sanskrit Segmentation. In *Proceedings of the South Asian Languages Analysis Roundtable XXVIII*, October.

Gérard Huet. 2024. Hoisting the colors of Sanskrit. In Arnab Bhattacharya, editor, *Proceedings of the 7th International Sanskrit Computational Linguistics Symposium*, pages 39–51, Auroville, Puducherry, India, February. Association for Computational Linguistics.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press.

S. D. Joshi. 1968. Patañjali's vyākaraṇa-mahābhādsya. samarthāhnika (p 2.1.1).

Amrith Krishna, Pavankumar Satuluri, Shubham Sharma, Apurv Kumar, and Pawan Goyal. 2016. Compound type identification in Sanskrit: What roles do the corpus and grammar play? In Dekai Wu and Pushpak Bhattacharyya, editors, *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pages 1–10, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Amba Kulkarni and Devanand Shukl. 2009. Sanskrit morphological analyser: Some issues. *Indian Linguistics*, 70(1-4):169–177.

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):1–38, April.

Anil Kumar. 2012. *Sanskrit Compound Processor*. Ph.D. thesis, University of Hyderabad, Hyderabad.

Sivaja S. Nair and Amba Kulkarni. 2010. The knowledge structure in amarakośa. In Girish Nath Jha, editor, *Sanskrit Computational Linguistics*, pages 173–189, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bhagyalata Pataskar. 1996. Some observations about the compound structure of *Aṣṭādhyāyī*. *Annals of the Bhandarkar Oriental Research Institute*, 77(1/4):121–131.

Jivnesh Sandhan, Amrith Krishna, Pawan Goyal, and Laxmidhar Behera. 2019. Revisiting the role of feature engineering for compound type identification in Sanskrit. In Pawan Goyal, editor, *Proceedings of the 6th International Sanskrit Computational Linguistics Symposium*, pages 28–44, IIT Kharagpur, India, October. Association for Computational Linguistics.

Jivnesh Sandhan, Ashish Gupta, Hrishikesh Terdalkar, Tushar Sandhan, Suvendu Samanta, Laxmidhar Behera, and Pawan Goyal. 2022. A novel multi-task learning approach for context-sensitive compound type identification in Sanskrit. In Nicoletta Calzolari, Chu-Ren Huang, Hansaem Kim, James Pustejovsky, Leo Wanner, Key-Sun Choi, Pum-Mo Ryu, Hsin-Hsi Chen, Lucia Donatelli, Heng Ji, Sadao Kurohashi, Patrizia Paggio, Nianwen Xue, Seokhwan Kim, Younggyun Hahm, Zhong He, Tony Kyungil Lee, Enrico Santus, Francis Bond, and Seung-Hoon Na, editors, *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4071–4083, Gyeongju, Republic of Korea, October. International Committee on Computational Linguistics.

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

Pavan Kumar Satuluri. 2015. *Sanskrit Compound Generation: With a Focus on the Order of Operations*. Ph.D. thesis, University of Hyderabad, Hyderabad.

Preeti Shukla, Amba Kulkarni, and Devanand Shukl. 2013. Geeta: Gold Standard Annotated Data, Analysis and its Application. In *Proceedings of the 10th International Conference on Natural Language Processing*, CDAC, Noida, India, December. NLP Association of India.

Appendices

A Clues for identifying the types

Num	**iic1** (a)	**iic2** (b)	**ifc** (c)	**Extra Information**	**Type**	**Rule**	**Examples**
1	*ktānta*	*a / an*	*ktānta*	-	$< a- < b - c > Tn > K2$	2.1.60	*kṛta-a-kṛtam*
2	*ktānta*	*prādi*	*ktānta*	-	$< a- < b - c > Tp > K2$	2.1.60	*kṛta-apa-kṛtam*
3	*avayava-vācī*	*ahorātra*	*ktānta*	*avayava-vācī aho-rātra*	$<< a - b > K - c > T7$	2.1.60	*pūrva-ahna-kṛtam, apara-rātra-kṛtam*
4	-	*a / an*	-	-	$< a- < b - c > Tn > K2$	2.1.60	*kraya-a-krayikā*
5	-	*a / an*	-	-	$< a- < b - c > Tn > K2$	2.1.60	*māna-un-mānikā*
6	*dik-vācī*	-	-	-	$<< a - b > T - c > B$	2.1.51	*pūrva-śālā-priyaḥ*
7	*saṅkhyā-vācī*	-	-	-	$<< a - b > Td - c > B$	2.1.51	*pañca-gava-dhanaḥ*

Table 7: Conditions for deciding constituency and assigning types for 3-component compounds

List	**Type**	**Rule**	**Examples**
1	T7	2.1.47	*udake-viśīrṇam, bhasmani-hutam*
2	T7	2.1.48	*pātre-samitāḥ, udumbara-maśakāḥ*
3	T7	2.1.44	*araṇye-tilakāḥ, vane-kaserukāḥ*
4	T	2.1.72	*mayūra-vyaṃsakaḥ, chātra-vyaṃsakaḥ*
5	A3	2.1.17	*tiṣṭhadgu, āyatī-gavam*

Table 8: Conditions for exceptional compounds

Num	**iic**	**ifc**	**Extra Information**	**Type**	**Rule**	**Examples**
1	*śreṇyādi*	*ktānta*	-	K	2.1.59	*śreṇi-kṛtāḥ, eka-kṛtāḥ*
2	*ktānta*	*ktānta*	-	K1	2.1.49	*snāta-anuliptaḥ*
3	-	*patita*	-	T2	2.1.24	*naraka-patitaḥ*
4	-	*patita*	-	T5	2.1.38	*svarga-patitaḥ*
5	-	*śritādi*[1]	-	T2	2.1.24	*kaṣṭa-śritaḥ*
6	-	*hita, rakṣita*	-	T4	2.1.36	*go-rakṣitam*
7	-	*bhīta*	-	T5	2.1.37	*vṛka-bhītam*
8	-	*apeta,* etc.[2]	-	T5	2.1.38	*sukha-apetaḥ*
9	*svayam*	*ktānta*	-	T2	2.1.25	*svayam-vilīnam*
10	*khaṭvā*	*ktānta*	-	T2	2.1.26	*khaṭvā-rūḍhaḥ*
11	*sāmi*	*ktānta*	-	T2	2.1.27	*sāmi-kṛtam*
12	*tatra*	*ktānta*	-	T7	2.1.46	*tatra-kṛtam*
13	*kālavācī*	*ktānta*	-	T2	2.1.28	*ahaḥ-saṅkrāntāḥ*
14	*stokādi*[3]	*ktānta*	*dūrārtha* words	T5	2.1.39	*śreṇi-kṛtāḥ*
15	*prāpta, āpanna*	-	-	T2	2.2.04	*prāpta-jīvikaḥ*
16	-	*ktānta* root from *karaṇa* verbs	*karaṇa* verbs	T5	2.1.32	*ahi-hataḥ paraśu-chinnaḥ*
17	-	words ending in *yat, ṇyat* root from *karaṇa* verbs	*karaṇa* verbs	T3	2.1.33	*kāka-peyā*

[1] *śrita, atīta, gata, atyasta, prāpta, āpanna*

[2] *apeta, apoḍha, mukta, apatrasta*

[2] *stoka, antika, dūra, kṛcchra*

Table 9: Conditions for group 1 *ktānta*

Num	iic	ifc	Extra Information	Type	Rule	Examples
1	*kālavācī*	*yat*-ending word	in the sense of *ṛṇa*	T7	2.1.43	*māsa-deyam*
2	*kālavācī*	-		T2	2.1.29	*muhūrta-sukham*
3	-	*pūrvādi*[1]	*ūnārtha* words	T3	2.1.31	*māsa-pūrvaḥ*
4	-	*annavācī* words	*annavācī* words	T3	2.1.34	*dadhi-odanaḥ*
5	-	*bhakṣyavācī* words	*bhakṣyavācī* words	T3	2.1.35	*guḍ-dhānāḥ*
6	-	*bhaya, bhīti, bhī*		T5	2.1.37	*vṛka-bhayam*
7	-	*śouṇḍādi* words	*śouṇḍādi* words	T7	2.1.40	*akṣa-śouṇḍaḥ*
8	-	*siddha, śuṣka, pakva, bandha*	-	T7	2.1.41	*cakra-bandhaḥ*
9	-	*dhvāṅkṣavācī* words	*dhvāṅkṣavācī* words	T7	2.1.42	*tīrtha-dhvāṅkṣaḥ*
10	-	*guṇavacana* words	*guṇavacana* words	T3	2.1.30	*śaṅkulā-khaṇḍaḥ*
10	-	*tadartha, artha, bali, sukha*	tadartha words	T4	2.1.30	*yūpa-dāru*

[1] *pūrva, sadṛśa, sama, ūnārtha, kalaha, nipuṇa, miśra, ślakṣṇa*

Table 10: Conditions for group 2 *non-ktānta*

Num	iic	ifc	Extra Information	Type	Rule	Examples
1	*yuvan, yuvati*	*khalati*[1]	-	K2	2.1.67	*yuva-khalatiḥ, yuva-jaratī*
2	*kumāra*	*śramaṇādi*	-	K	2.1.70	*kumāra-śramaṇā*
3	*eka*, etc.[2]	-	-	T	2.1.49	*eka-śāṭī, sarva-devāḥ*
4	*pūrvādi*[3]	-	-	K1	2.1.58	*pūrva-puruṣaḥ*
5	*sat*, etc.[4]	-	-	K1	2.1.61	*mahā-puruṣaḥ*
6	*katara, katama*	-	-	K1	2.1.63	*katara-kaṭhaḥ*
7	*kim*	-	-	K1	2.1.64	*kim-rājā*
8	*pāpa, aṇaka* *kutsa* words	-	-	K2	2.1.54	*pāpa-nāpitaḥ*
9	-	*vṛndāraka* *nāga, kuñjara*	-	K2	2.1.62	*go-vṛndāraka*
10	*catuṣpāda-jātivācaka*	*garbhiṇī*	-	K2	2.1.71	*go-garbhiṇī*
11	*varṇavācaka*	*varṇavācaka*	-	K3	2.1.69	*lohita-sāraṅgaḥ*
12	*jātivācaka*	*poṭā*, etc.[5]	-	K2	2.1.65	*agni-stokaḥ*
13	*jātivācaka*	*praśaṃsāvācaka*	-	K2	2.1.66	*go-matallikā*
14	*kṛtya, tulyārtha*	*ajātivācaka*	*tulyārtha* words	K	2.1.68	*tulya-śvetaḥ, sadṛśa-mahān*
15	-	*vyāghrādi*	-	K5	2.1.56	*puruṣa-vyāghraḥ*
16	*dravyavācī*	*guṇa*	-	K4	2-1-55	*śastrī-śyāmā*
17	*guṇa, saññā, kriyā*	*jātivācaka*	-	K1	2-1-57	*nīla-utpalam*
18	-	*kutsita* words	-	K2	2-1-53	*vaiyākaraṇa-khasūciḥ*

[1] *khalati, palita, valina, jarati, jaran*

[2] *eka, sarva, jarat, purāṇa, nava, kevala*

[3] *pūrva, apara, prathama, carama, jaghanya, samāna,madhya, madhyama, vīra*

[4] *sat, mahat, parama, uttama, utkṛṣṭa*

[5] *poṭā, yuvati, stoka, katipaya, ghṛṣti, dhenu, vaṣā, vehat, baṣkayaṇī, pravaktṛ, śrotriya, adhyāpaka, dhūrta*

Table 11: Conditions for group 3 *karmadhāraya*

Num	iic	ifc	Extra Information	Type	Rule	Examples
1	-	*prati*	-	A2	2.1.09	*sūpa-prati*
2	*akṣa, śalākā* numerals	*pari*	-	A2	2.1.10	*akṣa-pari*
3	*prādi*	-	-	A1	2.1.06	*adhi-strī, upa-kumbham*
4	*yāvat*	-	-	A1	2.1.07	*yāvat-amatram*
5	*apa, pari, bahis*	-	-	A1	2.1.12	*bahir-grāmam*
6	*ā*	-	-	A1	2.1.13	*ā-kumāram*
7	*abhi, prati*	-	-	A1	2.1.14	*prati-agni*
8	*anu*	-	-	A1	2.1.16, 2.1.16	*anu-vanam, anu-gaṅgam*
9	*pāre, madhye*	-	-	A7	2.1.62	*pāre-gaṅgam*
10	numerals	*vaṃśyavācī*	-	A6	2.1.19	*dvi-muni*
11	numerals	*nadīvācī*	-	A6	2.1.20	*sapta-gaṅgam*

Table 12: Conditions for group 4 *avyayībhāva*

Num	iic	ifc	Extra Information	Type	Rule	Examples
1	*ardha*	-	-	T1	2.2.02	*ardha-pippalī*
2	*dvitīya, tṛtīya caturtha, turya, turīya*	-	-	T	2.1.13	*dvitīya-bhikṣā*
3	-	*dvitīya, tṛtīya caturtha, turya, turīya*	-	T6	2.2.03	*bhikṣā-dvitīyam*
4	*ku*	-	-	T	2.2.18	*ku-puruṣaḥ*
5	*prādi*	-	-	T	2.2.18	*pra-ācāryaḥ*
6	*pūrvādi*[1]	-	-	T	2.2.01	*pūrva-kāyaḥ*
7	-	*yājakādi*	-	T6	2.2.09	*brāhmaṇa-yājakaḥ*
8	*digvācī*	-	named entity	T	2.1.50	*pūrva-iṣukāmaśamī*
9	*taddhitānta-digvācī*[2]	-	-	T	2.1.51	*paurva-śālaḥ*
10	*kālavācī*	-	-	T	2.2.05	*māsa-jātaḥ*
11	*īṣat*	*guṇavacana*	-	T	2.2.07	*īṣat-kaḍāraḥ*
12	words denoted by *gati*	-	-	T	2.2.18	*urarī-kṛtam*
13	-	bound morphemes[3]	-	U	2.2.19	*kumnbha-kāraḥ*
14	-	*krīḍā-jīvika*	-	T6	2.2.17	*danta-lekhakaḥ*

[1] *pūrva, apara, adhara, uttara*

[2] *digvācī* with *taddhita* suffix

[3] like *kāra*

Table 13: Conditions for group 6 *tatpuruṣaḥ*

Num	iic	ifc	Extra Information	Type	Rule	Examples
1	*āsanna, dūra, adhika*	*saṅkhyāvācī*	-	B	2.2.25	*āsanna-daśāḥ*
2	*digvācī*	*digvācī*	-	B	2.2.26	*dakṣiṇa-pūrvā*
3	*sa*	non-neuter gender	-	B	2.2.28	*sa-putraḥ*
4	indeclinables	*saṅkhyāvācī*	-	B	2.2.25	*upadaśāḥ*

Table 14: Conditions for group 7 *bahuvrīhi*