

IKML: A Markup Language for Collaborative Semantic Annotation of Indic Texts

Chaitanya S Lakkundi
chaitanya.lakkundi@gmail.com

Gopalakrishnan Rajaraman
gopal.rajaraman@gmail.com

Sai Rama Krishna Susarla
sai.susarla@gmail.com

Siddhanta Knowledge Foundation, Chennai

Abstract

Bhāratīya śāstra texts employ highly structured and well-defined patterns of discourse derived from Nyāya and Mīmāṃsā concepts. Though they are written in a flat text style, making their knowledge structure explicit greatly helps in understanding and interpreting their meaning. It also helps in building automated tools to mine these texts for insights, and in building computational models of śāstras. However, transforming śāstra texts into knowledge structures cannot yet be automated, as there is not enough annotated data to train machine-learning tools.

In this paper, we describe a novel markup language to help this process, called **IKML** (Indic Knowledge Markup Language) and a web user interface for collaborative semantic annotation of content using IKML, called **e-Bhashya**.

IKML offers a new way to represent śāstra texts and annotate them with knowledge metadata at multiple levels of abstraction in a collaborative manner. IKML is designed for easy collaborative editing by śāstra scholars, version control, graphical visualization and scripted processing. It employs best practices of popular languages such as YAML, XML and JSON and is auto-convertible to these languages. This makes IKML representation of śāstra texts amenable for processing by scripting and visualization tools for large-scale knowledge mining. The key guiding principles for IKML design are brevity, readability, extensibility, and minability.

IKML supports simultaneously representing an Indic language book as a scanned image document, OCR-extracted text, proof-corrected text, sentence and word-split version, grammar-tagged version, discourse-tagged version and a knowledge network using tantrayuktis, tātparya liṅgas, saṅgatis and nyāya sambandhas as well as augmentation with user notes, translations and comments. IKML supports tagging semantic relationships at multiple levels: between granthas/treatises, vibhāgas/sections, vākyas/sentences and viśayas/concepts.

1 Introduction

Bhāratīya knowledge is embedded in millions of manuscripts, some printed and mostly handwritten, in diverse fields, technical, religious, social. To mine such knowledge using the latest AI technologies such as ChatGPT, a big hurdle is making them available in the form of processable text in Indic languages. Indic languages are very different grammatically from English, but have a common, rich grammatical structure derived from Sanskrit. Annotating these texts grammatically (splitting the sandhis and samāsas) will greatly help in automating the further linguistic process and semantic knowledge extraction. However, this step is highly ambiguous and cannot be fully automated. The second major hurdle is the unfamiliarity of today's technologists with the language and discursive style used in these texts. This can be overcome only by involving traditional śāstra experts in the interpretation, and enabling them to annotate texts with authentic śāstra-compliant notes.

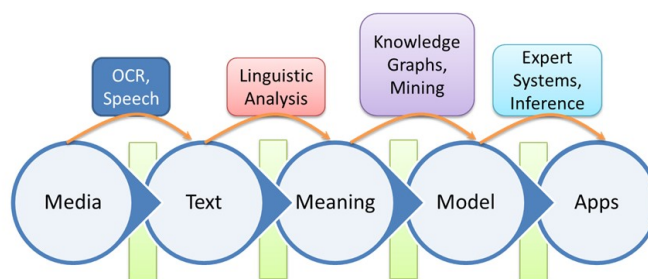
True utilization of Indic knowledge for modern innovation is possible only after making Indic textual heritage accessible and intelligible as above.

2 Outline of the Paper

The rest of the paper is organized as follows. First, we formally describe the problem we are trying to address, followed by a set of requirements that any solution must meet. Then we examine existing work to see how well they address the problem. Next, we describe the architecture of our solution including its data model and service organization which are our main contributions to the field. Next, we give an overview of our new Indic Knowledge Markup Language (IKML) language that we designed for the purpose along with example snippets. We describe the security and access control architecture, followed by our current implementation of the architecture. Next, we discuss specific end-user services in Indic knowledge education and research that our platform enables. We then present our evaluation of how the architecture meets the requirements stipulated earlier. Finally, we outline the value of the platform for various players in Indian Knowledge System (IKS) space, and conclude.

3 Problem Statement

Our objective is to develop a scalable architecture for multi-layered semantic annotation and knowledge mapping of Indic language content.



An End-to-end Indic Knowledge Engineering Ecosystem

Figure 1: Knowledge Transformation Pipeline

The above figure illustrates an end-to-end knowledge transformation pipeline needed to fully unleash the value of Indic textual heritage. This involves converting scanned images of Indic books to processable text, its linguistic analysis, conversion to knowledge models and finally deploying those models in consumable applications.

3.1 Design Requirements

In this section, we outline a set of requirements that an annotation solution must meet to address the problem effectively. This will help in evaluating existing solutions and in identifying gaps to be filled.

3.1.1 Versatility

The solution should accommodate a variety of use cases spanning the entire knowledge processing spectrum - media, text, syntax, concepts, discourse, knowledge.

- Image-to-text mapping
- Syntax and grammatical tagging
- Concept tagging and networking

- Discourse analysis from the perspective of multiple śāstras
- Knowledge mining
- Community commentary

3.1.2 Portability

Due to its need to be used over long-term over decades in the face of technology churn, the solution must allow portability of its data as well as tools as they evolve. It cannot be locked into specific data formats or tools.

3.1.3 Extensibility

The solution must allow the data to be manipulated by both humans and machines seamlessly. For flexibility in adoption of technology, the data should be auto-convertible to/from popular formats, and allow the use of powerful database and visualization tools. The solution must provide an open platform to facilitate third-party app development and integration.

3.1.4 Security

The solution must offer social media user authentication, multi-layered access control to allow multiple organizations to participate while protecting and monetizing their assets.

3.1.5 Scalability

The solution must scale to millions of users, thousands of content curators, billions of documents, hundreds of organizations, dozens of languages and scripts.

4 Related Work

We discuss three types of existing work related to Indic knowledge processing: Indic book repositories, Linguistic processing tools and Indic knowledge management platforms. Our effort belongs to the third category and adds value to existing content as well as accommodating other tools.

There are numerous online repositories of heritage Indic language content including digitally scanned books and texts (archive.org, scribd, sanskritdocuments.org, Gita Supersite¹, ebharati-sampat.in, Wikisource, Digital Library of India², Jain e-library³ to name a few). There are also manuscript collections yet to be deciphered.

In the second category, numerous computational linguistics tools have been developed for Indic language processing⁴⁵⁶⁷⁸⁹¹⁰. The bulk of these tools deal with grammatical analysis but not with higher order knowledge tagging, representation and mining. Very few frameworks exist for applying these tools on large content repositories such as the above.

There also exist efforts at devising content encoding format to capture the semantics of Indic content. One of the prominent formats is the Text Encoding Initiative (TEI) (Ide, 1994)(Ide and Véronis, 1995)(Cummings, 2013). It was established in 1988 to standardize the process of annotating resources. In 1994, the TEI proposed a set of guidelines deriving some of its principles from the Standard Generalized Markup Language (SGML). Most of the texts available

¹<https://www.gitasupersite.iitk.ac.in/>

²<https://ndl.iitkgp.ac.in/>

³<https://jainelibrary.org/>

⁴<https://sanskrit.uohyd.ac.in/>

⁵<https://sanskrit.inria.fr/>

⁶<http://sanskrit.jnu.ac.in/index.jsp>

⁷<https://www.cfilt.iitb.ac.in/wordnet/webswn/wn.php>

⁸<https://sanskritlibrary.org/>

⁹<https://sanskrit.iitk.ac.in/>

¹⁰<https://sambhasha.ksu.ac.in/>

on websites such as GRETEL¹¹, SARIT¹², TITUS¹³ use this encoding format. At its inception, it significantly facilitated NLP tasks by providing clear annotations for paragraphs, segments, links, references, and more. However, as the standard gained traction and new features were introduced, the number of tags expanded, with the current total reaching 588 elements¹⁴ and 275 distinctly-named attributes.

This extensive list of tags can make it challenging for users to remember and recall specific tags. Specialized taggers (Scharf, 2018) have also been developed with the aim of addressing this issue. While the taxonomy provided by TEI offers valuable documentation for derivative works, we believe that its current complexity can be overwhelming for newcomers.

Additionally, for large-scale, collaborative annotation of the thousands of available Sanskrit texts by śāstra scholars, it is essential to offer a compact, readable and intuitive syntax closer to their conventions.

The START¹⁵ (Sanskrit Teaching, Annotation, and Research Tool), developed by researchers at the University of Hyderabad (Nelakanti et al., 2024), integrates various tools including a morphological analyzer, segmenter, and dependency parser, to create an E-Reader for accessing Sanskrit literature. Users can import texts and annotate them with information such as anvaya, sandhi splits, and morphological tags. Future releases of the tool are expected to support data export in formats such as CONLL-U¹⁶.

To the best of our knowledge, the primary objective of the START tool is to establish a platform focused on E-Reading through data annotation. However, it does not adhere to a specific text-based format, instead relies on MongoDB for storing data, emphasizing instead the use of its platform interface. This approach may complicate usability and accessibility for users and researchers. Additionally, the platform currently lacks the capability for users to add knowledge-level tags such as tantrayuktis and represent it in a meaningful way for interaction with the text.

Neural approaches are also being used by many researchers for annotation and other NLP tasks (Gupta et al., 2020)(Krishna et al., 2023). Efforts such as SanskritShala (Sandhan et al., 2023) use neural networks to handle annotation tasks and offer an interface that enables annotators to correct the system’s predictions. Lately, newer AI models are also being fine-tuned for Sanskrit NLP tasks (Nehrdich et al., 2024).

E-bharatisampat¹⁷ is an online digital library that hosts various Sanskrit texts and enables OCR and proof-reading using its platform. However, the platform is limited to proof-reading. It does not support annotating the text.

Ambuda.org¹⁸ is a new addition to the E-Reader platform landscape, featuring a clean and user-friendly interface for reading Sanskrit texts. It includes sandhi split and parse data from Digital Corpus of Sanskrit¹⁹ (Hellwig, 2010 2021) and integrates multiple dictionaries for quick reference. The platform also supports OCR, proofreading and tagging (POS) capabilities. Users can also export data in plain text or XML format. However, as of now, Ambuda.org does not support specialized annotations.

Sangrahaka (Terdalkar and Bhattacharya, 2021) is a tool for annotating and querying knowledge graphs using the Neo4j DB backend (Terdalkar et al., 2023). This platform supports multiple types of annotations at knowledge level and has a detailed user interface. Although

¹¹<https://gretel.sub.uni-goettingen.de/>

¹²<https://sarit.indology.info/>

¹³<https://titus.uni-frankfurt.de/indexe.htm>

¹⁴<https://tei-c.org/release/doc/tei-p5-doc/en/html/REF-ELEMENTS.html>

¹⁵<https://sanskrit.uohyd.ac.in/start/>

¹⁶<https://universaldependencies.org/format.html>

¹⁷<https://www.ebharatisampat.in/>

¹⁸<https://ambuda.org/>

¹⁹<http://www.sanskrit-linguistics.org/dcs/index.php>

any specific annotation format isn't supported, the platform supports directly querying Neo4j DB using custom query templates. The relations can also be visualized using graphs. Antarlekhaka (Terdalkar and Bhattacharya, 2023) is another tool by the same authors to exclusively perform annotation tasks in their custom format.

Vedavaapi (Susarla and Challa, 2019) has developed an end-to-end architecture for Indic Knowledge processing. It made two design decisions: 1) using MongoDB as primary data store, and 2) embedding relations as attributes of objects instead of as first-class objects themselves. These decisions hurt its portability and extensibility. Vedavaapi had to build its own text search infrastructure instead of piggybacking on web search engines to index its content. Secondly, adding new knowledge tags by users wasn't possible as schema is baked into Vedavaapi software architecture. Several other frameworks also have these limitations.

Our solution is designed based on lessons learnt from these existing efforts.

5 Why a new Markup Language

There exist several languages that provide the expressive power needed for multi-layered knowledge representation that we envision. They include XML, JSON, GraphML etc. Just like high-level programming languages offer compact and natural abstractions for programmers compared to assembly language, the purpose of IKML is to offer convenient higher level abstractions for Indic Knowledge representation than existing markup languages. Its purpose is to augment rather than replace other languages. We evaluate the suitability of various languages for our task, using the following criteria.

5.1 Compact, Readable and Intuitive Syntax

We need a language that is easy to read and edit by humans who may not be programmers. XML is too verbose and cluttered as it employs significant metadata text that is non-user content. So are its variants such as GraphML, TEI, and to a lesser extent, JSON. YAML is more compact and readable as it employs indentation for nesting and avoids beginning and end tag markers.

IKML adopts YAML's syntax style but also avoids quoting user content and tag names for compactness and readability. Consequently, the content formatted in IKML is smaller in size compared to the same text in JSON, YAML, and XML.

5.2 Customizable and Validated Schema

Since we cannot anticipate all annotations ahead of time, we need the ability to specify schema and content in the same language to make content self-describing. To ensure development of tools for visualization and processing of content, we need builtin-support for schema validation as well. YAML falls short in this feature. IKML supports schema description in the same syntax as annotations.

5.3 Convenient Śāstric Abstractions for Experts

We need the ability to predefine śāstra-specific concepts as annotations for śāstra experts to use. IKML's schema definition with predefined values for tag attributes helps in this regard.

5.4 Non-hierarchical Knowledge Structures

Most markup languages only support hierarchical nesting of tags. However, knowledge structures tend to be non-hierarchical where relations between entities can impose a graph structure including cycles. The relation types are also diverse. To support them, IKML offers two features: 1) auto-assignment of globally unique ID to every annotation, and 2) relation to be treated as a first-class annotation (with its own globally unique ID) that refers to its endpoints via their IDs. Together, these features enable arbitrary annotations or text segments to be linked to express non-hierarchical relations. While currently, we use only pairwise relations, IKML allows more than 2 entities per relation to support hypergraphs.

6 E-Bhashya Architecture

In this section, we present E-bhashya, our proposed solution for collaborative human+machine annotation of Indic documents meeting the requirements outlined earlier.

6.1 Features at a Glance

- Google search for the entire library content for free.
- Book access based on subscription.
- Users can make comments and notes that others can see and respond to. Enables online communities around libraries.
- Śāstric analysis of content.

6.2 Data Model

At the core of e-Bhashya architecture is a data model for unified representation of both data and metadata at multiple levels of abstraction. In this model, entities are represented as a hierarchy of nested **tags** with **attributes** to capture tag properties. The schema of the tag hierarchy can also be described in the same data model, making it self-describing and extensible by users for multiple purposes.

A tag can have multiple instances of a subtag to express collections. But a tag can have only one instance of an attribute. If the same attribute is specified multiple times for a tag, the last value prevails.

To keep our data format tool- and database-neutral and enable web search, we chose to go for a flat text representation of all data and metadata. While we could have used any popular language for this representation such as JSON, YAML or XML, we devised a new markup language for specific reasons: our data representation should be readable, compact, editable by humans and amenable to intuitive change-tracking and version control using off-the-shelf tools like Git.

We have devised a new markup language called the **Indic Knowledge Markup Language (IKML)**. IKML adopts best practices from multiple existing languages and is auto-convertible to them for portability and ease of processing. It adopts an indentation-based syntax to express nesting relations like YAML. It uses tags like XML but avoids its verbosity of explicit start and end delimiters by going for indentation instead. That way, a tag's name needs to be specified only once. We give an overview of IKML syntax in a subsequent section.

In the following sections, we describe e-Bhashya's objects, which can all be expressed using IKML tag hierarchy model.

6.3 Object Types

At a high level, our data architecture supports building digitized online libraries of Indic documents with role-based access control at multiple granularities. A **library** is a collection of *granthas* or books with common access control settings. A **grantha** is a multilingual document - either text or multimedia. It can be divided into multiple sections, each called a **vibhāga**. Vibhāgas are of two types: those that constitute a physical layout view (pages, blocks and paragraphs) and a logical layout view (chapters, sections, sentences and words). **Annotations** can be associated with any vibhāga. Annotations are logically grouped by śāstra, and represent concepts specific to that śāstra (e.g., vyākaraṇa, nyāya, mīmāṃsā, tantrayukti, sāhitya, nāṭya). All these objects are assigned globally unique ids (**GUID**), enabling them to be referenced, accessed and shared over social media individually via URL.

Objects can be connected via śāstrically labeled relations called **sambandhas** in addition to their inherent nesting (parent-child) relation. *Sambandha* is also a first-class object having its own GUID. We currently support pairwise relations, but can extend to multi-object relations later as need arises for hypergraphs.

6.4 Object Networking

Sambandhas enable networking objects into a variety of rich knowledge maps at multiple levels of abstraction. We support three specific maps. A **vākya map** shown in Fig. 2 connects the sentences of a discourse via semantic relations called *tantrayuktis* to reflect the grantha's thought flow as intended by its author (*vivakṣā*). It facilitates exploring and summarizing a text only to a desired level of detail. A **viṣaya map** as shown in Fig. 4 connects words representing technical concepts of a grantha via Nyāya-śāstra-recognized sambandhas. It represents the ontology network of a grantha. The cypher queries used to create a viṣaya map are illustrated below 6.4.1. A **vibhāga map** shown in Fig. 3 displays the physical and logical layout of an object via the nesting hierarchy of parent-child relations (e.g., adhyayas, shlokas and their padas).

Figure 2: Vākya Map of Tarkasāṅgraha

6.4.1 Cypher Queries to Create Viṣaya Map

Viṣaya map is a graph of interrelations between technical terms of a śāstra grantha. It is derived from user-supplied annotation of nyāya sambandhas between words of vākyas denoting those terms.

We extract smb tags denoting Nyāya sambandhas between vākyas, and build a new graph with their `src_phrase` and `target_phrase` values as nodes.

```
WITH "https://api.siddhantakosha.org/ikmldata?gpath=libraries/smap-granthas/Tarkasāṅgraha-Moola/sambandhas-ikml.txt&fmt=neo4j" AS URL,
```

```
"libraries/smap-granthas/Tarkasāṅgraha-Moola/sambandhas-ikml.txt" as GPATH
```

```
CALL apoc.load.json(URL) YIELD value WHERE value.`.srcid` IS NOT NULL AND value.`.nyaya_sambandha` IS NOT NULL AND value.`.targetid` IS NOT NULL
```

```
MERGE (src:VSN content:value.`.src_phrase`, gpath: GPATH) MERGE (tar:VSN content:value.`.target_phrase`, gpath: GPATH)
```

```
CREATE (src)-[smb:VISHAYA_SMB nyaya_sambandha: value.`.nyaya_sambandha`,
id:value.id]->(tar);
```

```
MATCH (n:VSN)-[smb:VISHAYA_SMB]->(m) WHERE n.gpath STARTS WITH
"libraries/smāp-granthas/Tarkasangraha-Moola" RETURN n,smb,m;
```

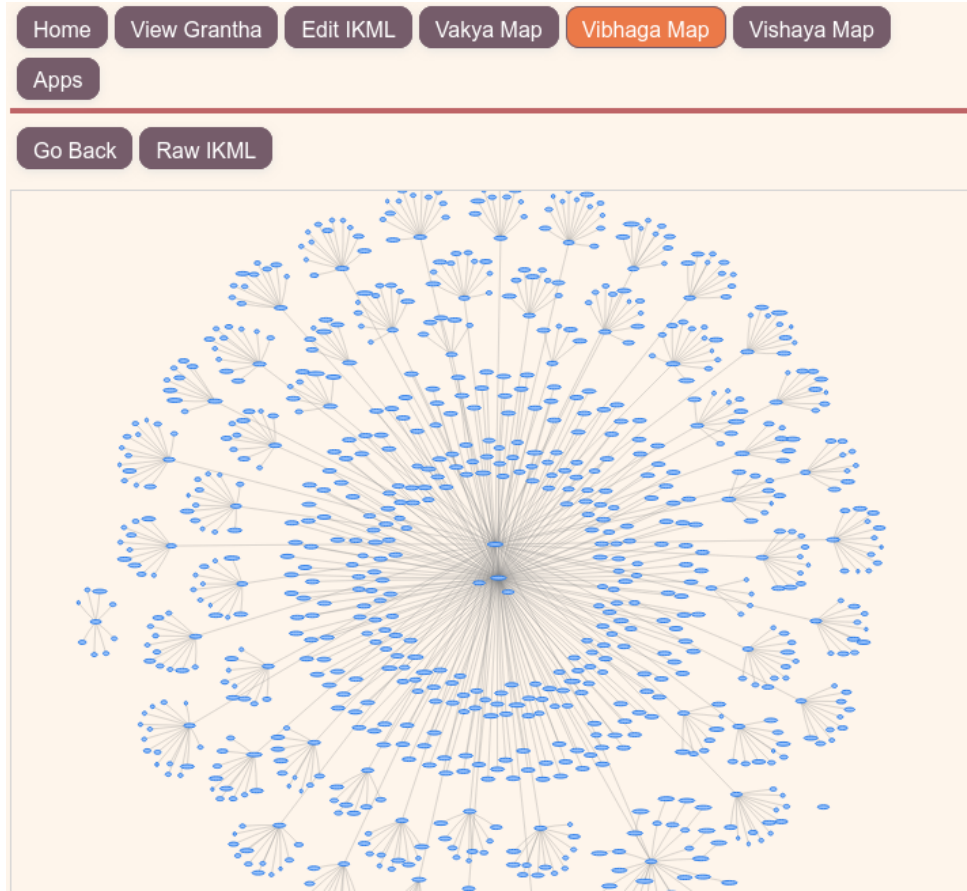


Figure 3: Vibhāga Map of Rasārṇava grantha.

6.5 Agents for Access Control

An **agent** is one who can be given access to an object in a specified role that indicates the operations allowed. Our architecture recognizes two types of agents: user and team. A **user** denotes an individual with his/her email id as a unique identifier. A **team** is a collection of users. A user can be part of multiple teams. Team membership is like a badge. A user is allowed access to an object if one of the teams he/she belongs to has that access.

All of the entities above are described using IKML tags. Hence IKML is the uniform persistent data format for everything in **e-Bhashya**.

6.6 Service Architecture

An annotated Indic document is represented as a plain text document in IKML format split into one or more files with “.ikml” extension. We use **GIT** version-controlled repository to store IKML files and track changes. The schema of IKML files is also described as an IKML file under a special tag called *ikml_schema*. We import the tag tree of an IKML file into **Neo4j** graph database to leverage the full power of its graph query language on IKML content.

Fig. 5 shows the architecture of e-Bhashya platform built to provide web-based graphical visualization, navigation and editing of Indic documents stored in an IKML file hierarchy on a cloud server.

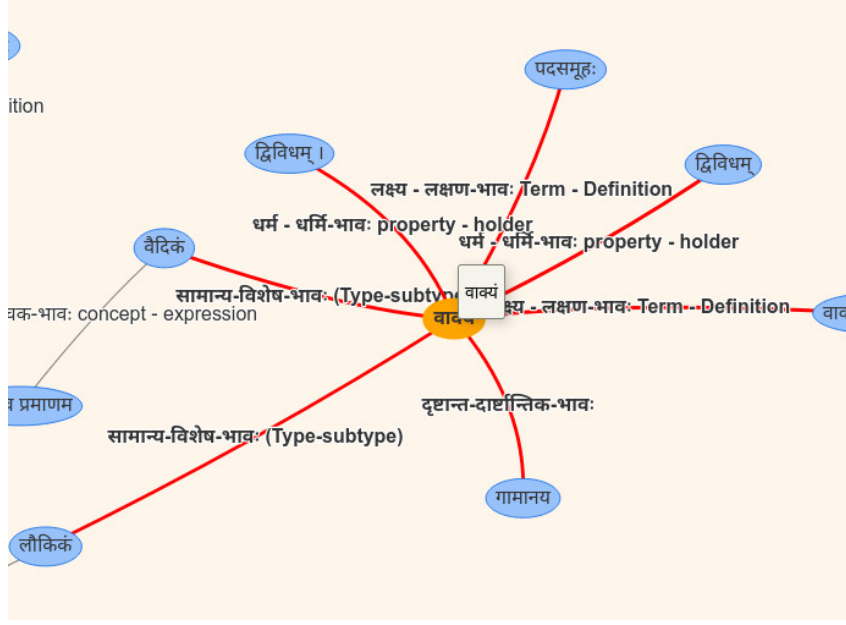


Figure 4: Viṣaya Map of Tarkasaṅgraha for the phrase वाक्यम्

We have a frontend Wordpress website running e-Bhashya as an embedded application. The application consists of a Javascript frontend interacting with a Python Flask backend server providing RESTful API access to IKML doc repository. The backend API server interacts with a Neo4j server, which caches IKML data as a graph for powerful visualization as concept maps. The Wordpress server provides user login and authentication facilities for e-Bhashya service using OpenAuth protocol via social-media accounts.

All these servers run in dedicated Docker containers. They can be scaled independently to handle load via deployment in Kubernetes clusters, where each service has multiple docker instances.

6.7 Web Search of e-Bhashya Content with Access Control

The Wordpress server also provides web access for search engine crawlers to index the IKML doc repository, so the entire e-Bhashya content including docs and their annotations is searchable with standard search engines like Google. We enforce access control on the searched content by serving it via API-based URLs. We distinguish between search engine crawl requests and other user requests using the *user-agent* field of the incoming HTTP request message.

6.8 Third-party Applications

E-Bhashya also offers an open platform for third-party applications to be developed and integrated into its service. These applications can have a Python Flask backend and a Javascript frontend that interact via a conduit wrapper API of e-Bhashya. E-Bhashya provides authentication and access control services transparently to applications so they can focus on their core functionality. We have developed several applications using this facility: tag statistics display, automatic word splitter, scanned PDF importer and an OCR proofreading editor.

6.9 Access Control

User access to objects is regulated via **Access control lists** (ACLs) assigned to objects. An **ACL** is an IKML tag specifying an agent (user or team) and the allowed role, which indicates the operations allowed. Multiple ACLs can be assigned to an object, and they apply to all children. An ACL overrides those assigned to ancestor objects, enabling powerful expression of rules and exceptions. When a user tries to access an object, the API server allows the operation

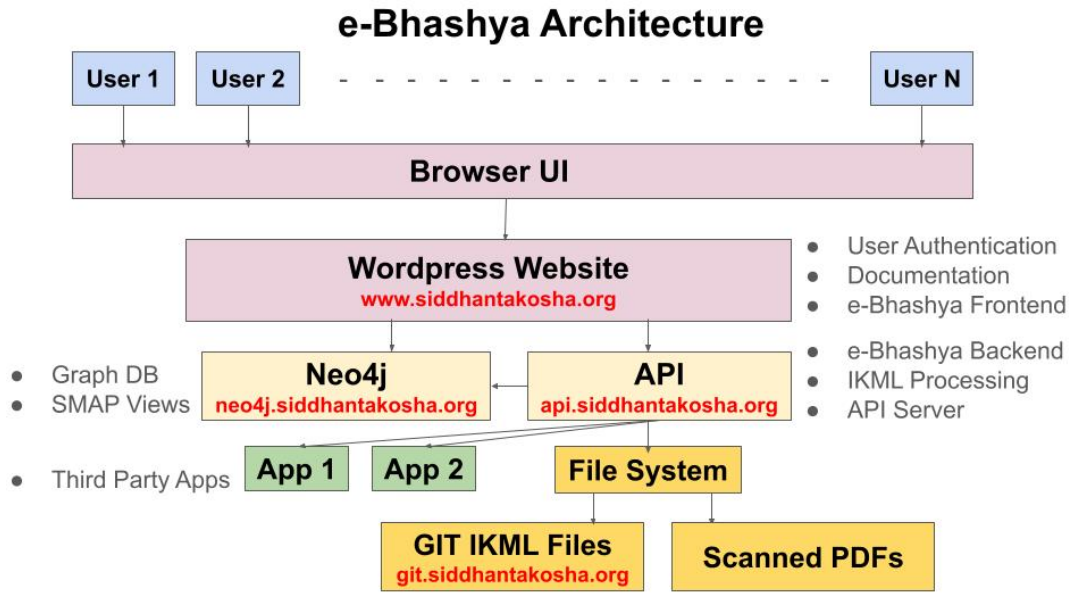


Figure 5: E-Bhashya Architecture

if it matches the first ACL in the object’s ancestor chain. Roles are explained in section 8.

7 Overview of IKML

In this section, we outline the IKML syntax and its salient features. Each line of an IKML document looks as follows:

```
[tag attr1=val1 ..] text
```

IKML defines **tags** within square brackets [] to describe a document **content** semantically. E.g., [va] denotes a vākya. Each tag and its content is given in its own line. A tag can have as its children (denoted by indentation with 2 spaces),

- **attributes** that give further information about tag content. They are specified within [] prefixed by a dot “.”. E.g. [.vibhakti]
- Other tags which denote sub-annotations of the content. E.g., [pa] enumerates each pada of the vākya separately. Repetition of same subtag multiple times denotes a collection.

An attribute can also be specified inline with the tag, e.g., [va label=”vākya”]. There are some predefined attributes

- **label**: descriptive phrase for a tag
- **rel_id**: auto-filled serial number for multiple repetitions of same tag within its parent tag.
- **rel_prefix**: string to be prefixed with rel_id.
- **id**: auto-generated globally unique identifying string for the tag instance.
 - id=<parent_tag.id>.<rel_prefix>-<rel_id>
 - E.g. tarka.v-10 denotes 10th vākya of tarka book
- **option**: possible preset values of an attribute can be defined using its “option” sub-attribute.

New tags can be defined along with their allowed sub tag hierarchy in a special tag called **ikml_schema**. A special tag called [**include**] <file path or URL> can be used to include IKML files in each other. This enables large documents to be modularly split into multiple

IKML files. Another special tag called [**inline**] is similar to include, but transparently makes the inlined file's content part of its parent file.

Content for a tag can be given after the tag in Unicode text without quotes. Leading and trailing white space is ignored.

7.1 IKML Schema

The IKML schema currently in use is attached in Appendix A.

7.2 Sample IKML Snippet

A sample snippet of Tarkasaṅgraha text in IKML format is listed in Appendix B.

8 E-Bhashya Access Roles

E-Bhashya service offers a multi-tier subscription model via **roles**. The following agent roles are supported:

- **Public** (no login needed): All content will be indexable by search engines, and show up in standard web search results. Access to matched content will be login-protected with custom views based on user privileges.
- **Guest** (free access with social media OAuth2 login): By default, anybody can login with their social media id. They will be auto-registered with GUEST privileges. In this role, users can view some books marked for free consumption and leave comments, but not modify existing content except deleting their own comments.
- **Subscriber** (paid annual individual subscription): a subscriber has access to specific libraries. Can add private annotations to books in those libraries and share them with public. They won't be visible to other users directly.
- **Contributor** (upon invitation only): a contributor can edit all content of libraries for which his team has access.
- **Team Admin**: The portal supports the concept of a **team** that can represent an institution or a group of individuals. A team can have one or more subscribers with **team admin** roles. This role allows them to control team membership, and access privileges to books owned by the team.
- **Super Admin**: has full read/write access to all content and can change roles of users.

The table below lists the operations allowed for various roles. In the table below, CRUD denotes Create, Read, Update and Delete operations. Each cell indicates which operations are allowed for users in a given role on content of given type.

Role	Own content	Other content	Own Annotations	Other Annotations
Public		R		R
Guest		R	CRUD	R
Subscriber	CRUD	R	CRUD	R
Contributor	CRUD	RU	CRUD	RUD
Team admin	CRUD	CRUD	CRUD	CRUD
Super admin	CRUD	CRUD	CRUD	CRUD

A library can be given access to other teams or public at large for specified roles.

9 Śāstra-compliant Notes

E-Bhashya platform offers a structured approach to annotating texts based on well-formed śāstric conventions. These annotations can be machine-generated, manually-supplied, or manual corrections of auto-generated content - appropriately marked as such for clarity. Annotations can also be nested arbitrarily, enabling powerful expression. E-Bhashya allows API-based crawling and annotation of its content. This allows machine-learning tools to mine and augment E-Bhashya content.

- **Vyākaraṇa Annotations:** this involves splitting compound words into their components by both *sandhi* and *samāsa*. Users can also add vyākaraṇa properties to words, such as liṅga, vacana, vibhakti etc. They can also add higher order śābda bodha commentary.
- **Sāhitya Annotations:** this involves annotating a text with Sāhitya śāstra-based properties such as chandas, alaṅkāra, bhāva, rasa, rīti, vṛtti, pāka etc.
- **Nāṭya śāstra Annotations:** This involves annotating a text or media clip with Nāṭyaśāstra concepts found in the clip such as mudrās, abhinaya, bhāva or rasa depicted.
- **Nyāya Annotations:** this involves tagging elements mentioned in a text and their inter-relations based on Nyāya classification of *padārthas* and *sambandhas*.
- **Tantrayukti Annotations:** this involves tagging sentences of a text's discourse by the tantrayukti used to enable accurate interpretation. Relations among multiple vākyas can also be tagged with sambandha tantrayuktis. This allows the semantic structure of a discourse to be visualized and texts navigated as a graph / network by concepts. This enables summarizing a book conceptually.

E-Bhashya platform provides a powerful way to network sentences, books and concepts as well as to visualize and navigate books as networks.

10 Unstructured Notes

E-Bhashya platform allows generic user notes on any part of its books down to a single word.

- Translations
- Student notes and comments
- Expert opinions
- Question banks

11 Implementation and Status

We have implemented E-Bhashya service as part of an online library of Indic books and śāstra treatises, called Siddhantakosha.org. It is hosted on a cloud server as a collection of Docker containers backed by a GIT-controlled IKML file hierarchy.

We have created three libraries:

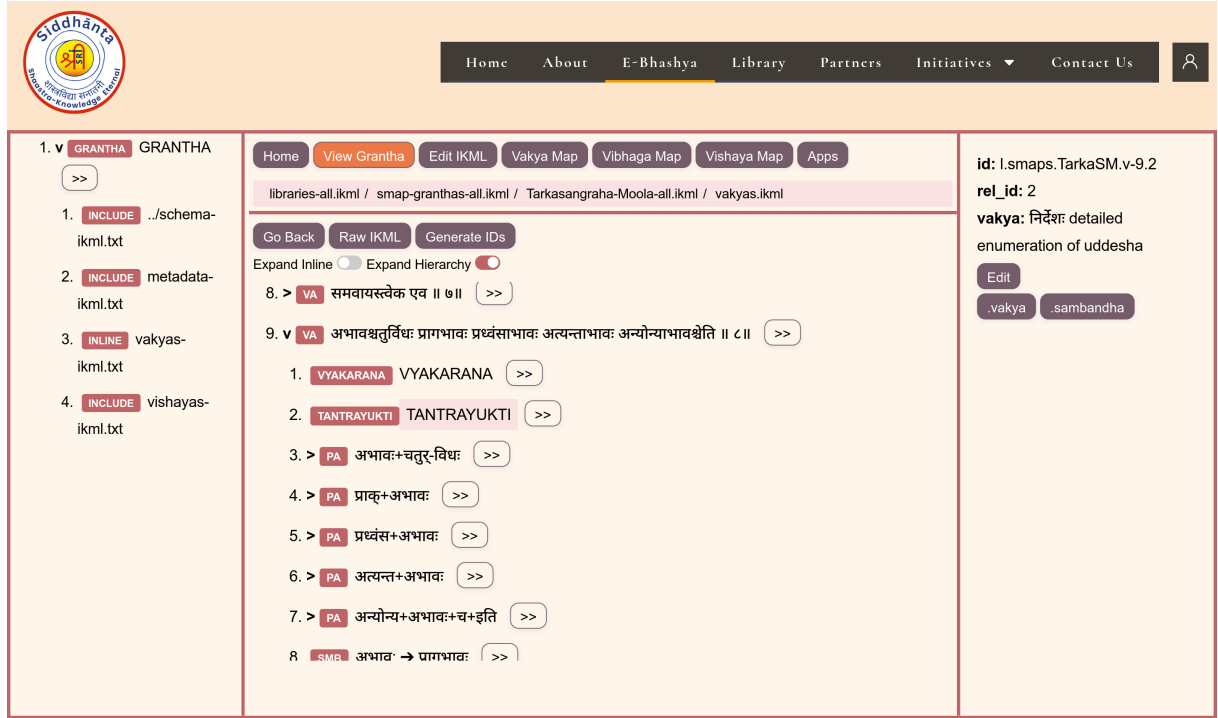
- **e-Kosha** consists of 1000 old scanned printed books whose pdfs have been converted to IKML text and made searchable by content on Google.
- **smap-granthas** consists 23 seminal Sanskrit śāstra granthas in full Devanagari text, augmented with vākya and viṣaya maps.
- **imported books** is a library of scanned printed pdf books stored elsewhere and imported into our platform as IKML text.

We have released an open-source Python package called **ikml_doc** for ikml document processing at pypi.org/project/ikml-doc with instructions on example usage. E-Bhashya uses this package internally.

11.1 E-Bhashya User Interface

The Web UI as shown in Fig. 6 consists of a tag navigation pane on the left, a tag attribute viewer and editor pane on the right, an IKML document viewer in the middle. The UI offers multiple views of an IKML document or section: as a clickable graph, as navigable tags or as raw IKML source. All edits to IKML, when saved, result in GIT commits on behalf of currently logged in user, allowing easy change tracking and undoing of erroneous updates.

So far, we have engaged dozens of remotely distributed Sanskrit students and scholars to do collaborative vākya splitting of two major śāstra texts: Rasārṇava (108 chapters and 6000 shlokas), and Bṛhat saṃhitā (1000+ chapters). The UI allows easy editing and navigation from a Smartphone as well as a computer, making it convenient for anytime, anywhere access.



The screenshot displays the E-Bhashya User Interface. At the top left is the Siddhanta logo. The navigation bar includes Home, About, E-Bhashya, Library, Partners, Initiatives, and Contact Us. The main content area is divided into three panes. The left pane shows a list of tags: 1. v GRANTHA GRANTHA, 1. INCLUDE ../schema-ikml.txt, 2. INCLUDE metadata-ikml.txt, 3. INLINE vakyas-ikml.txt, and 4. INCLUDE vishayas-ikml.txt. The central pane shows a list of tags with their attributes: 8. > VA समवायस्त्वेक एव ॥ ७ ॥, 9. v VA अभावश्चतुर्विधः प्रागभावः प्रध्वंसाभावः अत्यन्ताभावः अन्योन्याभावश्चेति ॥ ८ ॥, 1. VYAKARANA VYAKARANA, 2. TANTRAYUKTI TANTRAYUKTI, 3. > PA अभाव+चतुर्-विधः, 4. > PA प्राक्+अभावः, 5. > PA प्रध्वंस+अभावः, 6. > PA अत्यन्त+अभावः, 7. > PA अन्योन्य+अभावः+च+इति, and 8. SMB अभाव → प्रागभावः. The right pane shows metadata: id: l.smaps.TarkaSM.v-9.2, rel_id: 2, and vākya: निर्देशः detailed enumeration of uddesha. There are buttons for Edit, .vākya, and .sambandha.

Figure 6: Grantha View showing Tarkasaṅgraha text.

11.2 E-Bhashya API

The API server runs at api.siddhantakosha.org and exports the following endpoints.

GET /ikmldata	Get IKML Data in various formats (JSON, IKML, XML)
POST /ikmldata	Save IKML Data
GET /schema	Get Schema of a particular Tag name
GET /tpath/{tpath}	Get Child Tags of a particular Tag path
GET /apps	List Specs of registered third-party Apps
GET /app/{app_name}/{app_cmd}	Execute an App's backend API with appropriate parameters

11.3 Third-party Applications

Here are a few applications built on e-Bhashya.

11.4 Statistics

When this app is invoked on an IKML tag, it gives a count of all tags in its hierarchy. Below is the output for a library of Śāstra granthas currently available on E-Bhashya platform. As it states, we have 23 śāstra granthas totaling 37932 vākya of which 4249 have been split by Sandhi and samāsa. Tantrayuktis have been identified for 10700 vākya. There are a total of 238124 padas, and 13570 vākya and Nyāya Sambandhas have been tagged.

Tag Name	Count
grantha	23
va	37932
pa	238124
pa_top_level	225452
tantrayukti.vākya	10700
mimamsa	272
vyakarana.split	4249
smb	13570
su	3985
sh	5774

Table 1: Statistics of SMAP-Granthas Library

11.5 Pada splitter

Given a vākya that has been marked with delimiters indicating sandhi and samāsa components (either manually or via a vākya splitter tool), this app auto-generates the padas as child tags of the vākya, replacing existing padas if any.

For instance,

```
[va ...] abcdef ghj
  [vyakarana]
    [.src] manual
    [.split] a-b-c+d+e-f g-h+j
  [pa] a-b-c+d+e-f
    [.src] auto
    [pa] a-b-c
      [pa] a
      [pa] b
      [pa] c
    [pa] d
    [pa] e-f
      [pa] e
      [pa] f
    [pa] g-h+j
      [pa] g-h
        [pa] g
        [pa] h
      [pa] j
```

The [pa] tags are added by this app.

11.6 Scanned PDF Book Importer

Given the URL of a scanned PDF of a book and an e-Bhashya library, this app imports its contents into the library as an IKML grantha (PDF to JSON to IKML text). The app invokes Google vision API to run OCR and extract its text and layout in JSON format, and then converts it into IKML at paragraph granularity. Subsequently, it splits the book's entire text into sentences and adds them as subtags of the grantha.

```
[grantha]
  [.url] ...
  [inline] ocr-layout-ikml.txt
  [page]
    [block]
      [paragraph] ...
      [paragraph] ...
    [block]
      ...
  [page]
  [inline] vakyas-ikml.txt
  [va] ...
  [smb]
  [.target_id] <src_paragraph_id>
  [va] ...
```

We have used this app to import a popular dharma śāstra reference text called *Dharma Sindhu*, which is now available only in scanned PDF format. We invoked Google OCR to extract text into JSON format and converted it to IKML. Below is a screenshot of the result, which shows the source URL.

The screenshot displays the Siddhanta e-Bhashya library interface. The top navigation bar includes links for Home, About, E-Bhashya, Library, Partners, Initiatives, and Contact Us. The main content area is divided into three columns. The left column shows a library view with 'Imported Books' and a list of files: 'metadata-ikml.txt' (inline) and 'DHARMA SINDHU/all-ikml.txt' (include). The middle column shows the grantha view for 'DHARMA SINDHU', with a list of files: 'layout-ikml.txt' (include) and 'vakyas-ikml.txt' (include). The right column displays the grantha's metadata, including content, id, rel_id, source_url, and url, along with buttons for Edit, .title, .stitle, .author, and .publisher.

Figure 7: Displays the "Dharma Sindhu" Grantha, which was directly imported into e-Bhashya from archive.org.

Here is a size comparison of the book in various formats.

Book format	Size
Scanned PDF	81 MB
OCR output in JSON	340 MB

File Format	Layout File (MB)	Vakyas File (MB)	Total (MB)
IKML	3.9	5.9	9.8
JSON	5.5	10.5	16.0
XML	4.1	6.7	10.8
YAML	4.6	7.6	12.2

Note: Layout file contains the extracted text along with physical layout of every page of the book. Vakyas file contains all the extracted vākyas from the book.

This table shows that IKML representation has the smallest file size. It is ~1.6 times more compact than JSON for the same text content - 9.8MB vs. 16MB. The OCRred version in IKML format of an 81MB scanned PDF image file is 3.9+5.9 = 9.8MB, roughly an 8x reduction in size.

12 End-user Services

We envision the following consumer services to use the E-Bhashya portal.

- Śāstric Analysis of Web Content
- Training Datasets for Indic Knowledge processing Tool Development
- Online Courses
- Online Proficiency Tests
- Research Aids

12.1 Śāstric Analysis of Web Content

Bhāratīya Śāstras provide an alternative scientific framework and perspective to understand the contemporary world and current events. By analysis, we mean describing/labelling those events in śāstra’s technical terminology and conceptual categories, and predicting their evolution based on śāstric models. This will eventually devise śāstric solutions to current problems.

For example, current geopolitics can be analyzed and predicted from nīti, artha and dharma śāstras. Music and dance can be tagged with Indic Aesthetics concepts. Social media conversations can be tagged with Nyāya-based logic.

We call this *Śāstra Prayoga*, and use it to facilitate training and evaluation of śāstra students in śāstric thinking. To facilitate such śāstric annotation, we envision developing browser plugins to help users tag any web-based content with śāstra tags provided by E-Bhashya. Such tags will be stored in E-Bhashya platform and overlaid on public websites for viewing by other users.

12.2 Training Datasets for Indic Knowledge Processing

E-Bhashya API allows its entire collection to be programmatically accessed and augmented. For instance, its sandhi and samāsa-split sentences can be used to train AI models, which can be redeployed for further automated linguistic tagging. Its manually tagged sentences with tantrayuktis can be used to develop automatic tantrayukti identification tools.

E-Bhashya architecture provides a way to seamlessly integrate human and machine intelligence for reliable knowledge processing at scale, and bring śāstra application to the mainstream.

12.3 Online Courses

E-Bhashya allows a rich and hands-on training mechanism for students of Bhāratīya knowledge. It exposes students to a larger portion of text via self-study and guided assignments than traditional teaching methods.

The Vākya map enables concept-based exploration of śāstra texts at progressively increasing levels of detail comfortable to the reader. This is in contrast to a flat discourse, which can be overwhelming.

Teaching a śāstra text using E-Bhashya platform involves giving reading assignments to students, training and assessing the student’s ability to apply śāstric concepts practically. Teachers can pose questions on a larger portion of a text, and students can submit answers directly on the platform as their annotations. This allows peer learning, collaborative team projects and multi-textual comparative studies along with project reports that can be built on by future students over time.

This level of rich engagement with Bhāratīya knowledge can spur innovation, but is not possible with traditional pedagogy.

12.4 Online Proficiency Tests

E-Bhashya allows thorough examination of a candidate’s prowess in a śāstra, not only in theory but also in application.

We envision Śāstra Proficiency Tests to be conducted on E-Bhashya platform. Students can be asked to submit their commentary as annotations on any śāstra text including its adhikaraṇas, vākyas or concepts. They can be asked to apply śāstra paribhāṣā to characterize contemporary content.

12.5 Research Aids

E-Bhashya converts a text into a network of individually referenceable semantic constructs (e.g., sentences, concepts, sections etc) via its tag data model. This enables seamless networking and comparative study of concepts across multiple treatises. For example, one can quickly reference explanation of a technical term used in a main text in its bhashya.

We believe this use of technology is crucial to navigate the vast volume of Indic knowledge corpus, and dramatically reduces the time to develop new insights.

13 Evaluation

In this section, we examine how well e-Bhashya design meets the requirements we have stipulated for a transformative Indic knowledge engineering solution.

13.1 Versatility

The unified data representation of IKML coupled with its amenability to machine processing enables us to cover a variety of use cases unlike related solutions.

13.2 Portability

Our design choice of text-based representation of primary application data and metadata enables us to leverage a wide variety of existing technologies for search, visualization and linguistic processing without getting locked in.

13.3 Extensibility

Our solution is designed from the ground-up for seamless integration of human and machine input. Our API enables third-party extension of our functionality as demonstrated above.

13.4 Security

Our API-based access control with ACLs allows a variety of institutional deployments of Indic libraries to coexist and share each other’s resources while protecting their assets.

13.5 Scalability

The independently scalable server components enable elastic scaling of resources with load. However, we are yet to evaluate the scalability of our platform in practice.

14 Scope for Collaboration

Our solution is designed from the ground up with an open architecture to facilitate collaboration between diverse players in the Indic Knowledge ecosystem. Instead of reinventing the wheel, it is essential to harness existing content, tools, technologies and talent to augment them with new value.

Śāstra scholars can use our platform to come up with new intuitive presentation of Indic knowledge for modern consumption. Tool developers can apply their tools on large datasets by porting them to our platform. Content repository owners and publication houses can add new search, visualization and mining capabilities to their content via our platform while retaining its ownership. AI and ML experts can use our platform as a source of training data for new Indic knowledge mining tools.

15 Conclusions and Future Work

In this paper, we have described the architecture and implementation of a web platform built for scale called e-Bhashya for creating searchable, annotatable digital libraries of Indic texts. At its core lies a novel markup language called IKML.

We have already gained experience in using the platform for unique visualization of śāstra treatises as knowledge maps, and are planning to augment several existing Indic document repositories with śāstra-compliant notes and training in śāstra prayoga.

References

- [Ajotikar and Scharf2020] Tanuja P Ajotikar and Peter M Scharf. 2020. Development of a TEI standard for digital Sanskrit texts containing commentaries. In Devendranath Pandeya, Dipesh Katira, and Janakisharan Acharya, editors, *Proceedings of the International Conference Bhāṣyaparamparā Jñānapravāhaśca*, pages 462–476, Veraval. Shree Somnath Sanskrit University.
- [Ajotikar and Scharf2023] Tanuja P Ajotikar and Peter M Scharf. 2023. Development of a TEI standard for digital Sanskrit texts containing commentaries: A pilot study of bhaṭṭi's rāvaṇavadha with mallinātha's commentary on the first canto. In Amba Kulkarni and Oliver Hellwig, editors, *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*, pages 128–145, Canberra, Australia (Online mode), January. Association for Computational Linguistics.
- [Ajotikar et al.2024] Tanuja P Ajotikar, Ketaki Kaduskar, and Peter M Scharf. 2024. Using TEI for digital Sanskrit editions containing commentaries: A study of kālidāsa's raghuvaṃśa with mallinātha's sañjīvanī. In Arnab Bhattacharya, editor, *Proceedings of the 7th International Sanskrit Computational Linguistics Symposium*, pages 52–66, Auroville, Puducherry, India, February. Association for Computational Linguistics.
- [Cummings2013] James Cummings, 2013. *The Text Encoding Initiative and the Study of Literature*, chapter 25, pages 451–476. John Wiley & Sons, Ltd.
- [Goyal and Huet2016] Pawan Goyal and Gérard Huet. 2016. Design and analysis of a lean interface for sanskrit corpus annotation. *Journal of Language Modelling*, 4(2):145–182.
- [Gupta et al.2020] Ashim Gupta, Amrith Krishna, Pawan Goyal, and Oliver Hellwig. 2020. Evaluating neural morphological taggers for Sanskrit. In Garrett Nicolai, Kyle Gorman, and Ryan Cotterell, editors, *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 198–203, Online, July. Association for Computational Linguistics.
- [Hellwig2010 2021] Oliver Hellwig. 2010–2021. Dcs - the digital corpus of sanskrit. <http://www.sanskrit-linguistics.org/dcs/index.php>.

- [Ide and Véronis1995] Nancy Ide and Jean Véronis. 1995. *Text encoding initiative: Background and contexts*, volume 29. Springer Science & Business Media.
- [Ide1994] Nancy Ide. 1994. Encoding standards for large text resources: The text encoding initiative. In *COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics*.
- [Krishna et al.2023] 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.
- [Kulkarni et al.2010] Malhar Kulkarni, Chaitali Dangarikar, Irawati Kulkarni, Abhishek Nanda, and Pushpak Bhattacharyya. 2010. Introducing sanskrit wordnet. In *Proceedings on the 5th global wordnet conference (GWC 2010), Narosa, Mumbai*, pages 287–294.
- [Nehrdich et al.2024] 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.
- [Nelakanti et al.2024] Anilkumar Nelakanti, Amba Kulkarni, and Nakka Shailaj. 2024. Start: Sanskrit teaching; annotation; and research tool—bridging tradition and technology in scholarly exploration. In *Proceedings of the 7th International Sanskrit Computational Linguistics Symposium*, pages 113–124.
- [Sandhan et al.2023] Jivnesh Sandhan, Anshul Agarwal, Laxmidhar Behera, Tushar Sandhan, and Pawan Goyal. 2023. Sanskritshala: A neural sanskrit nlp toolkit with web-based interface for pedagogical and annotation purposes. *arXiv preprint arXiv:2302.09527*.
- [Scharf and Chauhan2023] Peter M Scharf and Dhruv Chauhan. 2023. Rāmopākhyāna: A web-based reader and index. In *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*, pages 146–154.
- [Scharf2018] Peter Scharf. 2018. TEITagger: Raising the standard for digital texts to facilitate interchange with linguistic software. In *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 17th World Sanskrit Conference*, pages 229–257.
- [Susarla and Challa2019] Sai Susarla and Damodar Reddy Challa. 2019. A platform for community-sourced Indic knowledge processing at scale. In Pawan Goyal, editor, *Proceedings of the 6th International Sanskrit Computational Linguistics Symposium*, pages 68–82, IIT Kharagpur, India, October. Association for Computational Linguistics.
- [Terdalkar and Bhattacharya2021] Hrishikesh Terdalkar and Arnab Bhattacharya. 2021. Sangrahaka: A tool for annotating and querying knowledge graphs. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1520–1524.
- [Terdalkar and Bhattacharya2023] Hrishikesh Terdalkar and Arnab Bhattacharya. 2023. Antarlekhaka: A comprehensive tool for multi-task natural language annotation. In Liling Tan, Dmitrijs Milajevs, Geeticka Chauhan, Jeremy Gwinnup, and Elijah Rippeth, editors, *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pages 199–211, Singapore, December. Association for Computational Linguistics.
- [Terdalkar et al.2023] Hrishikesh Terdalkar, Arnab Bhattacharya, Madhulika Dubey, S Ramamurthy, and Bhavna Naneria Singh. 2023. Semantic annotation and querying framework based on semi-structured ayurvedic text. In Amba Kulkarni and Oliver Hellwig, editors, *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*, pages 155–173, Canberra, Australia (Online mode), January. Association for Computational Linguistics.

Appendix A.

Here is the schema of tags described in IKML. Under each tag, the schema lists allowed child tags. [.option] tag is used to list allowed preset values of an attribute.

Schema-ikml.txt

```
[ikml_schema]
  [.lang] Sanskrit
  [.script] Devanagari
  [include] shaastra-schema-ikml.txt
  [include] vyakaraṇa-schema-ikml.txt
  [include] nyaya-schema-ikml.txt
  [include] mimamsa-schema-ikml.txt
  [root_tags]
    [va]
    [sh]
    [cm]
    [tr]
    [smb]
    [vibhaga]
  [.src label="annotation source"]
    [.option] auto
    [.option] manual
    [.option] endorsed
    [.option] corrected
  [.role label="Access Role"]
    [.option] public
    [.option] subscriber
    [.option] contributor
    [.option] admin
    [.option] superadmin
  [library label="Book Collection"]
    [grantha]
    [team role=".."] team_name
  [user label="User"] email
    [.name] name
    [.phone] phone
  [team label="Institution"] team_name
    [user role=".."] email
  [sahitya label="Sahitya annotation"]
    [.vritti]
  [pa label="Pada"]
    [.rel_prefix] p
    [vyakarana]
    [cm]
    [smb]
  [su label="Sutra"]
    [.rel_prefix] su
  [adhyaya label="Chapter"]
    [.rel_prefix] ady
  [vibhaga label="Section"]
    [.rel_prefix] sec
    [vibhaga]
```

```
[va]
[vishaya label="Concept"]
  [.rel_prefix] c
  [vishaya]
[va label="Vakya"]
  [.rel_prefix] v
  [pa]
  [tr]
  [cm]
  [smb]
  [vyakarana]
  [nyaya]
  [sahitya]
  [mimamsa]
  [tantrayukti]
[tr label="translation" lang="English"]
[cm label="comment" userid=""]
[sh label="shloka"]
  [.rel_prefix] sh
  [.image label="image url"]
  [va]
  [smb]
  [tr]
  [cm]
  [vyakarana]
  [nyaya]
  [sahitya]
  [mimamsa]
  [tantrayukti]
[grantha label="grantha"]
  [.title] ..
  [.stitle label="short title"] ..
  [.author label="author"] ..
  [.publisher label="publisher"] ..
  [page rel_prefix="pg"]
    [.pgnum label="page number"]
    [.image]
    [block rel_prefix="b"]
      [paragraph rel_prefix="pr"]
        [va]
        [sh]
    [va]
    [sh]
[smb label="Sambandha"]
  [.rel_prefix] r
  [.srcid]
  [.targetid]
  [.src_phrase]
  [.target_phrase]
  [nyaya]
  [tantrayukti]
```

Appendix B.

Here is the IKML source of a snippet of the Tarkasaṅgraha text. All the **id** attributes and **rel_id** attributes are auto-generated.

```
[va id=1.smaps.TarkaSM.v-10 rel_id=10] तत्र गन्धवती पृथिवी ।
[.src] manual
[vyakarana id=1.smaps.TarkaSM.v-10.1 rel_id=1]
[.src] manual
[.split] तत्र गन्धवती पृथिवी ।
[tantrayukti id=1.smaps.TarkaSM.v-10.2 rel_id=2]
[.vakya] पदार्थः Term definition
[pa id=1.smaps.TarkaSM.v-10.p-1 rel_id=1] तत्र
[.src] auto
[pa id=1.smaps.TarkaSM.v-10.p-2 rel_id=2] गन्धवती
[.src] auto
[pa id=1.smaps.TarkaSM.v-10.p-3 rel_id=3] पृथिवी
[.src] auto
[va id=1.smaps.TarkaSM.v-11 rel_id=11] सा द्विविधानित्याऽनित्या च ।
[.src] manual
[vyakarana id=1.smaps.TarkaSM.v-11.1 rel_id=1]
[.src] manual
[.split] सा द्वि-विधा नित्या+अनित्या च ।
[tantrayukti id=1.smaps.TarkaSM.v-11.2 rel_id=2]
[.vakya] निर्देशः detailed enumeration of uddesha
[pa id=1.smaps.TarkaSM.v-11.p-1 rel_id=1] सा
[.src] auto
[pa id=1.smaps.TarkaSM.v-11.p-2 rel_id=2] द्वि-विधा
[.src] auto
[pa id=1.smaps.TarkaSM.v-11.p-2.p-1 rel_id=1] द्वि
[.src] auto
[pa id=1.smaps.TarkaSM.v-11.p-2.p-2 rel_id=2] विधा
[.src] auto
[pa id=1.smaps.TarkaSM.v-11.p-3 rel_id=3] नित्या+अनित्या
[.src] auto
[pa id=1.smaps.TarkaSM.v-11.p-3.p-1 rel_id=1] नित्या
[.src] auto
[pa id=1.smaps.TarkaSM.v-11.p-3.p-2 rel_id=2] अनित्या
[.src] auto
[pa id=1.smaps.TarkaSM.v-11.p-4 rel_id=4] च
[.src] auto
```