Vedavaapi: A Platform for Community-sourced Indic Knowledge Processing at Scale

Sai Susarla
School of Vedic Sciences
MIT-ADT University, Pune
sai.susarla@gmail.com

Damodar Reddy Challa
Vedavaapi Foundation
Bangalore

Abstract

Indic heritage knowledge is embedded in millions of manuscripts at various stages of digitization and analysis. Numerous powerful tools and techniques have been developed for linguistic analysis of Samskrit and Indic language texts. However, the key challenge today is employing them together on large document collections and building higher level end-user applications to make Indic knowledge texts intelligible. We believe the chief hurdle is the lack of an end-to-end, secure, decentralized system platform for (i) composing independently developed tools for higher-level tasks, and (ii) employing human experts in the loop to work around the limitations of automated tools to ensure curated content always. Such a platform must define protocols and standards for interoperability and reusability of tools while enabling their autonomous evolution to spur innovation.

This paper describes the architecture of an Internet platform for end-to-end Indic knowledge processing called Vedavaapi that addresses these challenges effectively. At its core, Vedavaapi is a community-sourced, scalable, multi-layered annotated object network. It serves as an overlay on Indic documents stored anywhere online by providing textification, language analysis and discourse analysis as value-added services in a crowd-sourced manner. It offers federated deployment of tools as microservices, powerful decentralized user / team management with access control across multiple organizational boundaries. social-media login and an open architecture with extensible and evolving object schemas. As its first application, we have developed human-assisted text conversion of hand-written manuscripts such as palm leaf etc leveraging several standards-based open-source tools including ones by IIIT Hyderabad, IIT Kanpur and University of Hyderabad.

We demonstrate how our design choices enabled us to rapidly develop useful applications via extensive reuse of state-of-the-art analysis tools. This paper offers an approach to standardization of linguistic analysis output, and lays out guidelines for Indic document metadata design and storage.

1 Introduction

There is growing interest and activity in applying computing technology to unearth the knowledge content of India's heritage literature embedded in Indic languages due to its perceived value to modern society. This has led to several research efforts to produce analysis tools for Indic language content at various levels – text, syntax, semantics and meaning Goyal et al. (2012; Kumar (2012; Huet (2002; Kulkarni (2016; Hellwig (2009). Many of these efforts have so far been addressing algorithmic issues in specific linguistic analysis problems. However, as the tools mature and proliferate, it becomes imperative to make them interoperable for higher order document analytics involving larger document sets with high performance. We categorize existing tools for Indic knowledge processing into three buckets - media-to-text (e.g., OCR (image to text), speech recognition (audio to text)), text-to-concept (e.g., syntax-, semantics- and discourse analysis), and concept-to-insight (e.g., knowledge search, mining, inference and decision-making). For instance, though several alternative linguistic tools exist for Samskrit text

analysis (morphological analysis, grammatical checking), they use custom formats to represent input text and analysis outcome, mainly designed for direct human consumption, and not for further machine-processing. This inhibits the use of those tools to build end-user applications for cross-correlating texts, glossary indices, concept search etc.

On the other hand, the number of Heritage Indic documents yet to be explored is staggering. Data from National Mission for Manuscripts NAMAMI (2012) indicate that there are more than 5 million palm leaf manuscripts that are scanned but not catalogued for content, let alone converted into Unicode text to facilitate search. In addition, The Internet Archive project MongoDB (2019) has a huge collection of scanned printed Indic books. Very few of them have been converted to text. There are also hundreds of thousands of online Unicode Samskrit documents yet to be analyzed linguistically for knowledge mining. Use of technology is a must to address this scale.

We believe that to take Indic knowledge exploration to the next level, there needs to be a systematic, end-to-end, interoperability-driven architectural effort to store, exchange, parse, analyze and mine Indic documents at large scale. Due to lack of standardized data representation and machine interfaces for tools, Indic document analysis is unable to leverage numerous advances in data analytics that are already available for English and other languages.

Moreover, Indic documents pose unique challenges for processing. First, a vast majority of them are handwritten or in often poorly scanned archaic printed modes in numerous languages, scripts and media. Existing linguistic tools are inadequate to handle their complexity and diversity. Second, human feedback and correction in a community-sourced mode is essential to curate Indic document content at scale for further machine processing. But the architecture of many existing tools is not amenable to incorporating human input and adapting to it. Finally, Indic knowledge collections and processing tools are fragmented across multiple organizations and administrative boundaries. Hence a centralized approach to user authentication, access control and accounting will not be acceptable.

To overcome these challenges, this paper presents Vedavaapi, a novel platform architecture for community-sourced Indic document processing to transform digitized raw Indic content into machine-interpretable knowledge base. Through Vedavaapi this paper makes the following contributions to facilitate large-scale Indic knowledge processing:

1. A federated RESTful service architecture to support dynamic Indic knowledge processing workflows by leveraging independently evolving services, where each service can be deployed and scaled independently to handle load.

2. A canonical object model to represent document analytics output that enables interoperability between multiple tools in the document processing pipeline and also transparent integration of human feedback at each stage without modifying the tools themselves.

3. A NoSQL-based object store that supports self-describing, versioned schemas to help tool and data evolution over time.

4. A uniform, hierarchical security and access control model for users and object collections that supports decentralization of policies for flexible management across organizational boundaries. This model also allows individual tool providers to meter usage for chargeback to end-users.

The rest of the paper is organized as follows. In Section 2, we define the problem of Indic Knowledge processing, its requirements and the scope of our work. In Section 3, we illustrate the challenges in the use of existing tools for Indic knowledge processing to motivate our work. In Section 4, we present the principles that guide the design of our solution Vedavaapi. In Section 5, we describe the key architectural aspects of Vedavaapi including its object model, security model and deployment. In Section 6, we present an overview of our current implementation and

a qualitative evaluation against our objectives. In Section 7, we outline ideas for future work and conclude.

2 Indic Knowledge Processing: Overview and Status

By Indic knowledge, we refer to the practices, techniques and principles that evolved in Ancient India over centuries across all disciplines. Some of that knowledge has been documented in written form via manuscripts, while some got transmitted down to the present via oral, craft and cultural traditions. The objective of Indic Knowledge Processing (IKP) is to recover, preserve, paraphrase and leverage Indic knowledge sources for contemporary applications.

Heritage Indic documents come in all media formats and sizes. They include palm leaf and other manuscripts containing hand-written text preserved over millennia, books printed over the last 2 centuries, audio/video recordings of discourses/renderings by traditional scholars, and thousands of Unicode texts available over the web. Some of these have been digitized, but not yet converted to machine-processable text. They come in dozens of Indic scripts, languages and fonts in multiple combinations NAMAMI (2016), making their organization and processing an engineering challenge. In addition, much of Indic tribal knowledge is still locked up as regional traditions yet to be recorded and captured from their practitioners. Many Indic documents use languages with similar grammatical structure to Samskrit. Samskrit literature is well known to have a rigorous linguistic discipline that makes it more amenable to machine-processing and automated knowledge extraction than other natural languages Goyal et al. (2012). IKP involves creating services to explore Indic knowledge content at various levels – text extraction, syntactic and semantic analysis, knowledge search, mining, representation and inference.

The potential for automated mining of Indic knowledge due to its linguistic base of Samskrit, coupled with the sheer size of Indic document corpus yet to be examined, opens the opportunity to pursue Scalable Indic Knowledge Processing as an impactful research area in computing. This area is inherently multi-disciplinary, and involves rich media analytics (of audio, video, images), machine-learning, computational linguistics, graph databases, knowledge modeling and scale-out cloud architecture.

Figure 1 illustrates the various stages of a typical IKP pipeline covering three distinct transformations: media to text, text to concept, and concept to insight. Each of these stages produces a high volume of metadata in the form of analysis output, content indexes and user feedback that need to be persisted. Currently, there is a huge corpus of digitized content to feed the pipeline and numerous tools for various stages of the pipeline, but disjointed and not usable in tandem.

This paper presents the architecture of a novel software platform that bridges the gaps in the IKP pipeline to help rapidly transform digitized Indic knowledge content into useful applications. Some of our target applications include an E-reader for Indic texts that provides search within scanned or audio/video documents, glossary of technical terms used in a book, concept map and knowledge map views and semantic queries. The scope of this paper is restricted to architectural issues and not the algorithmic details of specific stages of the pipeline or the end-user applications.

2.1 Requirements of an IKP Platform

In addition to scalable performance to handle millions of documents by thousands of simultaneous users, an IKP platform must have the following properties:

Durability: It must provide both data and metadata persistence, so users or services can build on prior analysis by others.

Extensibility: The platform must support functional extensions to its services via APIs. It should also provide well-documented data formats and interfaces to incorporate available knowledge sources and analytics tools into its fold. This allows existing analysis tools to be reused in larger contexts than anticipated originally.

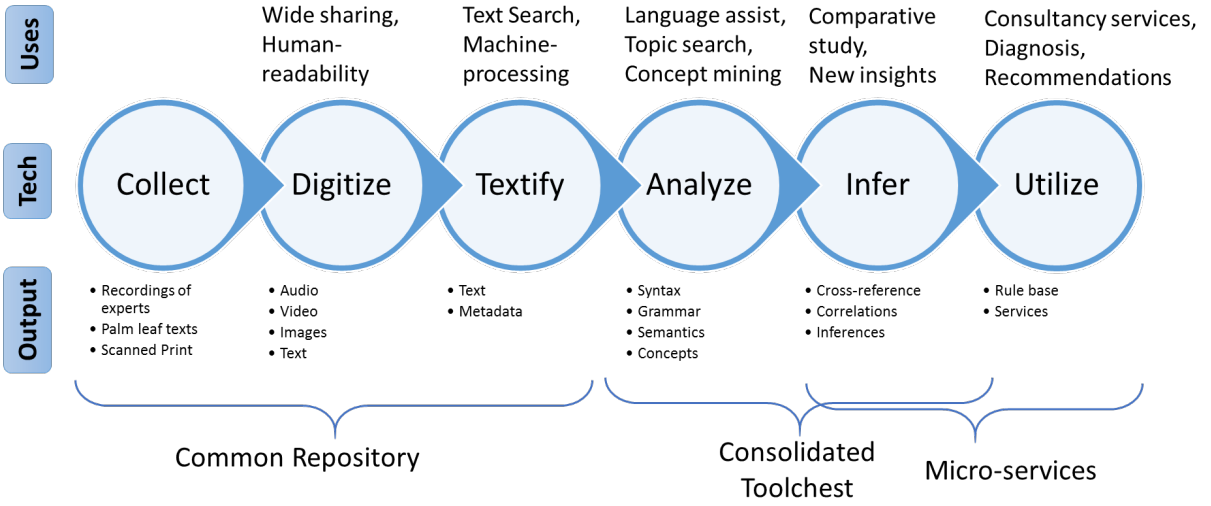

Figure 1: Indic Knowledge Processing Pipeline.

Crowd-sourcing: Ambiguity is inherent in natural language understanding. To help resolve ambiguity in analysis and enable users to enrich each other's knowledge through the platform, it must accept human feedback (analogous to Wikipedia) and and adapt to it. However to reduce user burden of repetitive corrections, the IKP system must have built-in intelligence to auto-apply suggested corrections to similar contexts.

3 Architectural Considerations for IKP

We now discuss several architectural implications of the above requirements and how existing solutions handle them.

3.1 Handling OCR Errors

First, consider the conversion of digitized content into text, referred to as the "textify" stage in the IKP pipeline of Figure 1. Optical Character Recognition (OCR) technology has matured to extract printed text in many Indic languages from high quality scanned images. Google offers a paid Vision API service Google (2019) that is more than 95% accurate on scanned images of resolution higher than 100 DPI. Open source alternatives also exist (Tesseract Tesseract (2019), Sanskrit OCR by Hellwig Hellwig (2019)), but are not found to be as effective on low-resolution or skewed scans of printed text. The accuracy levels of these services is adequate for direct human consumption for text search purposes, but not for further machine processing. Proof-reading of even a 95% accurate OCR output is a tedious manual effort. Existing OCR services do not have feedback-driven correction in their workflow. Such an adaptation facility would greatly enhance the utility of OCR by reducing repetitive manual work over time.

An IKP system must leverage these OCR tools but also facilitate building human feedback collection and tool re-training workflows around them. Another problem is that the bulk of Indic texts are in handwritten manuscripts with irregular layouts, (see Figure 2 for examples) and existing text segmentation and layout detection schemes are poor at handling them. A more effective alternative for designing an OCR solution would be to separate layout detection and text recognition into modular services and employ the best tools for each service. This enables one to handle printed as well as hand-written text recognition that improves over time, leveraging state-of-the-art tools. In Section 6, we discuss how Vedavaapi achieves that.

Figure 2: The irregularity of text layouts in Palm leaf manuscripts.

3.2 Human-assisted Language Analytics

Machine processing of Indic documents is inherently prone to errors due to ambiguity, For instance, morphological analysis Kulkarni (2016; Huet (2002) of a Samskrit sentence produces alternative semantic trees sometimes running into hundreds. Text segmentation to detect words from a punctuation-free Indic character sequence can also generate multiple alternative segmentations. Such tools still need human intervention both to supply the context to prune the choices during analysis, and to select a meaningful option from analysis output. Further, system adaptation needs to be built in to create self-improving analyzers. All this requires a mechanism to capture human feedback persistently and incorporate it into future analysis tasks. The IKP architecture should provide user-feedback-driven adaptation as a value-addition on top of individual analysis tools, and define standard interfaces to exchange that information with the tools.

3.3 Handling Data Diversity

The input data for an IKP workflow are source documents, which are mostly read-only content. The document analysis tools augment original content with one or more alternate views (e.g., morphological analysis of a sentence, a concept map, an OCR output). When a user annotates those views, some of them become irreplaceable and hence must be stored durably. From a mutability standpoint, an IKP system must deal with three types of content with different rates of churn:

Read-only Source Content that is never updated after creation,

Mutable System-inferred Content that can be reproduced by re-running analytics, and

Mutable Human-supplied Content including user annotations and corrections to system-inferred content.

IKP's data store should clearly demarcate these three types and treat them differently to avoid imbalance in storage performance. Also, for the same source content, there could be multiple alternate views at multiple levels of semantics and granularity that need to be tracked as such. For instance, there could be a sentence-level analysis, paragraph-level analysis and global analysis that coexist for a document.

3.4 Implications of Crowd-sourcing

When human input is solicited for correction, there needs to be a facility to track multiple alternate suggestions, rank them by user reputation and provide a consolidated view that represents the most acceptable suggestion. Similarly, the user feedback can be used as training data for

machine-learning tools to minimize the need for subsequent corrections. Hence an IKP system must maintain version histories for content updates.

Resolving competing suggestions in a crowd-sourcing situation is a well-understood phenomenon with numerous solutions. The IKP platform must enable the use of such solutions in IKP use cases by facilitating persistent capture of the appropriate data.

4 IKP Architecture: Guiding Principles

Based on the considerations discussed in the previous section, we outline a set of guiding principles for the design of an IKP architecture as follows:

- Federation: The architecture must adopt an open platform approach that enables services to be independently developed, deployed and maintained by multiple organizations.
- Interoperability: The architecture must allow existing tools to be leveraged in larger Indic document analytics workflows which the tool developers might not have anticipated.
- Community-sourcing: The architecture must support overlaying of human input and correction to the output of any of the services transparently.
- Decentralized security and Accounting: The architecture must allow single-sign-on across multiple services while allowing them to independently meter resource consumption by end-users for chargeback. For Indic knowledge processing to be accelerated, participation of thousands of scholars and enthusiasts across multiple organizational boundaries is essential. Decentralized authentication and authorization ensures that. Decentralized accounting allows the development of value-add services to enrich the platform in an economically viable manner.

5 Architecture of Vedavaapi

In this section, we describe the architecture of Vedavaapi, a platform we are building to facilitate large-scale IKP workflows. Vedavaapi is a web-based platform that offers rich, multi-layered annotated views of document collections stored natively or elsewhere (such as at archive.org). Figure 3 illustrates the architecture of Vedavaapi. It is organized as a set of loosely coupled web services and web applications interacting via RESTful APIs. Each such service is packaged as a cluster of Docker containers Docker (2019) for ease of deployment and scaling. A web service only responds to API requests, whereas a web application offers end-user interaction as well, via a GUI.

5.1 The Vedavaapi Ecosystem

One of the core web services is a Vedavaapi site that provides secure controlled access to annotated Indic document collections of an organization. There could be many Vedavaapi sites, and each of them offers an administrative boundary with its own user and document collection management. A Vedavaapi dashboard web application orchestrates end-user interaction with one or more Vedavaapi sites. This application handles single-sign-on user login via social media, user and team management, document collection management and launching IKP workflows via invoking other Vedavaapi web services.

To facilitate third-party IKP tools (e.g., OCR and linguistics tools) to operate on document collections of Vedavaapi sites securely, Vedavaapi provides an adapter library to be bundled with those tools. This adapter provides user authentication and secure access to any Vedavaapi site. A third-party IKP tool can be converted into a Vedavaapi IKP service by wrapping it with a RESTful API frontend along with the adapter library. Using the adapter library, IKP services interact with Vedavaapi sites to retrieve their data and store IKP output on behalf of logged in users.

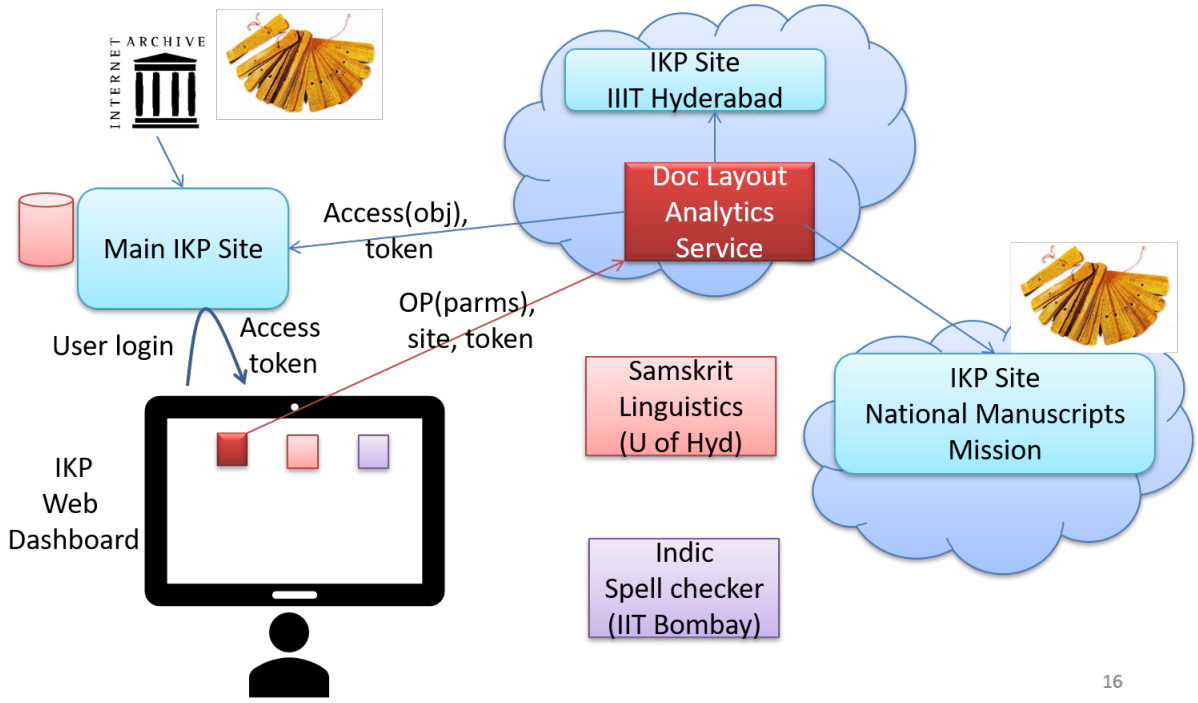

Figure 3: Vedavaapi Federated Architecture. Example IKP services that are active in this illustration are Samskrit Linguistics, Indic Spell Checker and Doc Layout Analytics services.

An IKP service can be registered with multiple Vedavaapi sites to offer its services via specific API endpoints or to manipulate specific document types. When an end-user requests an IKP operation on an Indic document at a site, he/she is presented with a list of registered IKP services available for that operation. For instance, multiple OCR tools can be made available to extract text from a scanned page.

A Vedavaapi site consists of a persistent object store, user and team management service, access control service, and an OAuth service. The object store service houses all of the site's persistent metadata in a NoSQL database as JSON objects, and provides a powerful navigational query interface. The document source images are stored in the local file system.

5.2 User Authentication

Each Vedavaapi site maintains its own user accounts, teams and access control permissions for its document collection, and exports itself as an OAuth service provider. The Vedavaapi dashboard application authenticates a user via social media login and registers a new user to a site upon first access. Soon after login, it procures an OAuth access token to represent the user for subsequent operations at the site. Unlike cookies, the access token can be passed around to other IKP services to represent the user when accessing Vedavaapi documents.

When a third-party IKP service needs to access or update a document at a Vedavaapi site, it simply passes on the access token it received from its caller (usually the Vedavaapi dashboard application). Thus IKP tool developers are relieved from performing user authentication and access control. IKP services can also invoke other IKP services recursively while representing the same user transparently throughout the delegation chain. Moreover, given the access token, any IKP service provider can retrieve the user profile for accounting / metering the user's operations against his/her quota. This enables the service to chargeback based on usage regardless of where it is the invocation chain.

5.3 Vedavaapi Object Model

A Vedavaapi site stores and manages Vedavaapi objects, which are of three types - agents, resources and annotations. An agent is either a user (either human or bot) who has an account with the site and needs to be authenticated, or a collection of users called a team. Resources are the objects whose access by users needs to be regulated, and which can be annotated by users. Annotations are pieces of information tagged to resources or other annotations, such as the output of an IKP analysis. Every object is referred to by its unique UUID generated by the underlying object store (in our case, MongoDB MongoDB (2016)).

Examples of resources include scanned books, text documents, videos, and collections of other resources such as libraries. IKP applications can define their own resource types. Vedavaapi recognizes a special type of resource called SchemaDef, which describes the schema of any Vedavaapi object using the JSONSchema description language standard Schema (2019). Resources form a strict parent-child hierarchy, whereas an annotation can refer to multiple resources and hence induces a directed acyclic graph. Examples of annotations include transcript, translation, commentary, linguistic analysis output etc.

Vedavaapi object model allows object relationships to be captured via three types of links - source / parent object, target / referred object and members list of a collection object. All objects are referred by their UUIDs issued by the underlying object store (MongoDB in our case). The source / parent link is used to link a resource to its container or parent resource such as books to their library or pages to their book. The target / referred link is used to link an annotation to its referred object such as a transcript to a paragraph. Figure 5 illustrates a network of Vedavaapi objects generated in a typical OCR workflow.

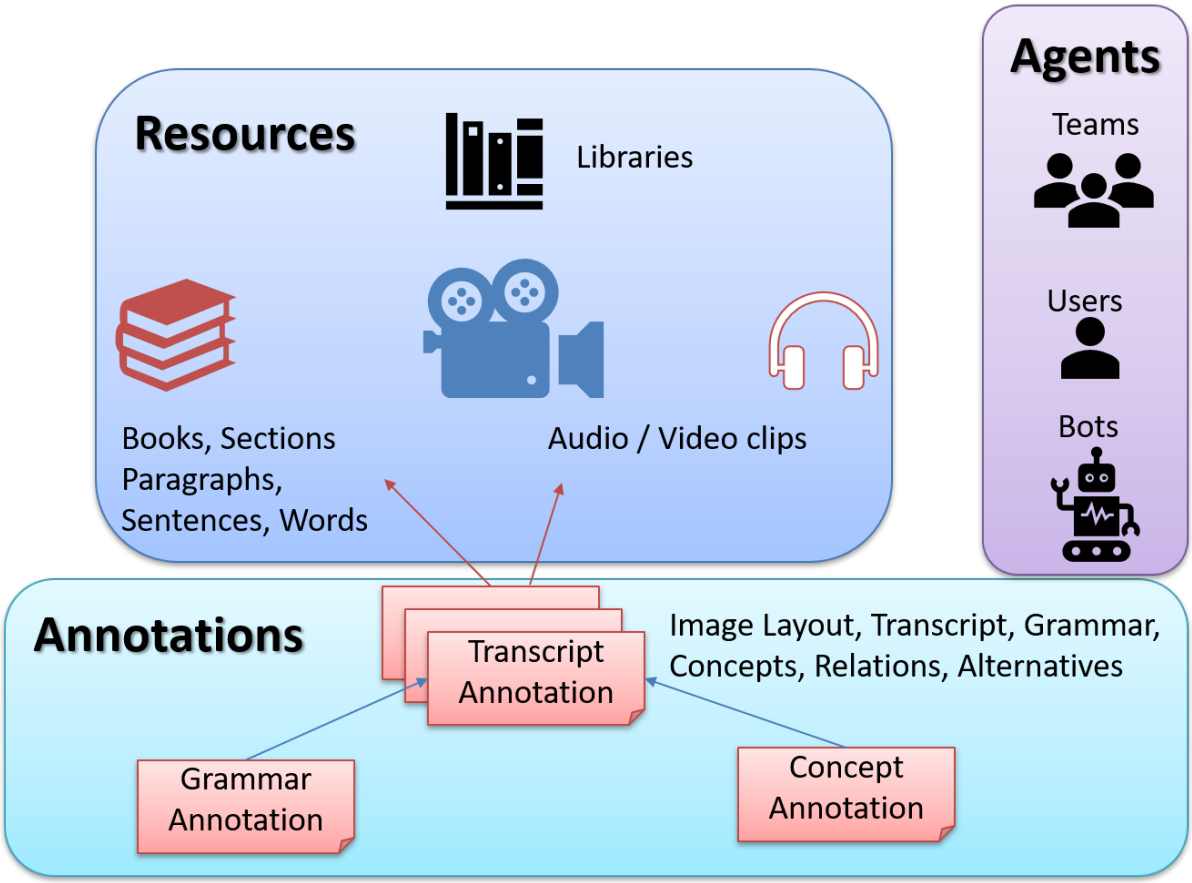

Figure 4: Vedavaapi Object Model.

Often, an IKP workflow needs to persist the ordering of objects in a collection, e.g., pages

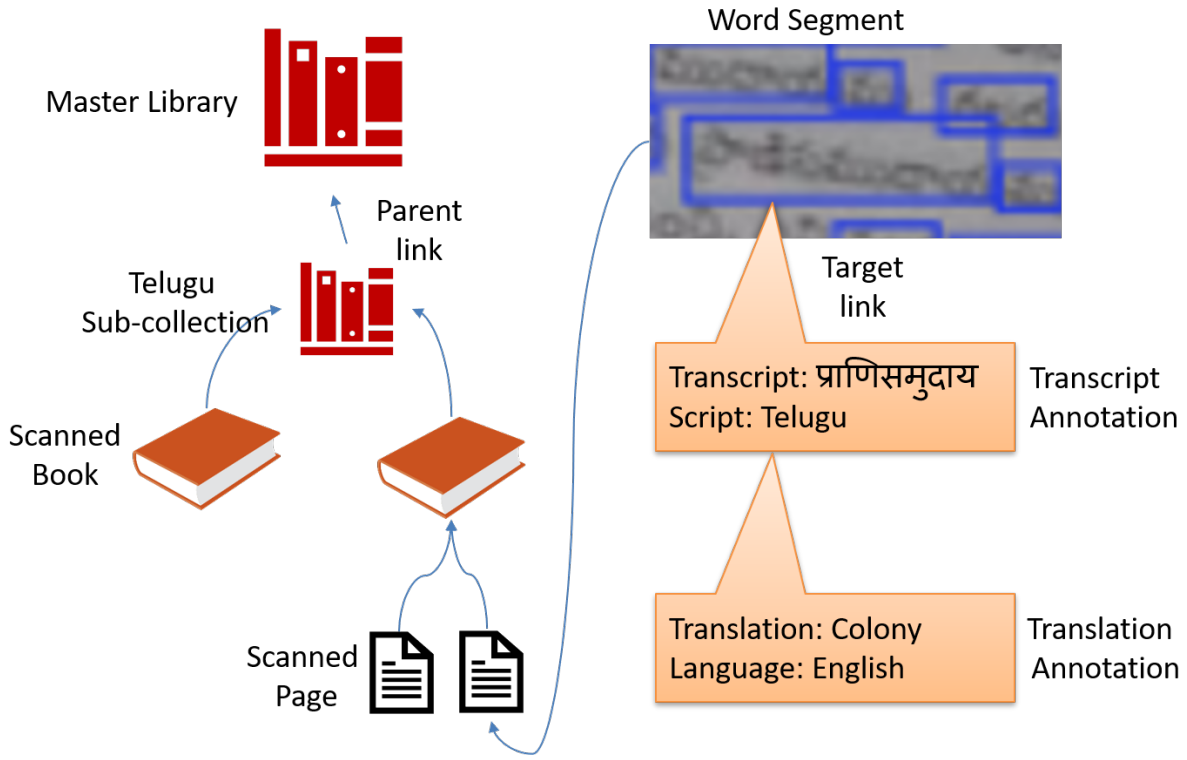

Figure 5: An Example Vedavaapi Object Network showing both resources and annotations.

in a book or words in a page. To facilitate that, we define a Sequence resource object as one that enumerates its child resources via their numeric index field. Sometimes, we also need to persist different orderings of the same set of objects, e.g., a user's bookmarked pages in a book. To support that, we define a sequence annotation object as one that explicitly enumerates a set of arbitrary object ids in a specific order via its own "members" field. To capture multiple alternatives produced by an IKP analysis output, we define a choice annotation object as one that returns one of its referring annotations according to a selection strategy such as first, random, vote, etc. For instance, when a morphological analysis produces multiple alternatives, they can be persisted under a choice annotation to be presented to users for voting. Finally, to represent arbitrary semantic linkage among concepts or among sentences in a discourse, we need a generic way to explicitly annotate the relation among groups of objects. We define a relation annotation object as one that captures the semantic relations among two or more resources or annotations.

Figure 6 illustrates the entire class hierarchy of Vedavaapi objects along with their inter-linkage conventions.

5.4 Vedavaapi Access Control

A large-scale IKP platform must allow different teams the flexibility to manage access to their own document collections independently, while providing administrative override when required. Vedavaapi provides fine-grain control over operations on object content as well as the inter-object network by users and teams. To do so, Vedavaapi recognizes the following operations on objects:

- read: allows reading this object's content, i.e, metadata attributes
- updateContent: allows updating the object's attributes
- delete: allows deleting the object and delinking it from its network.
- linkAnnos: allows creating annotations on this object.

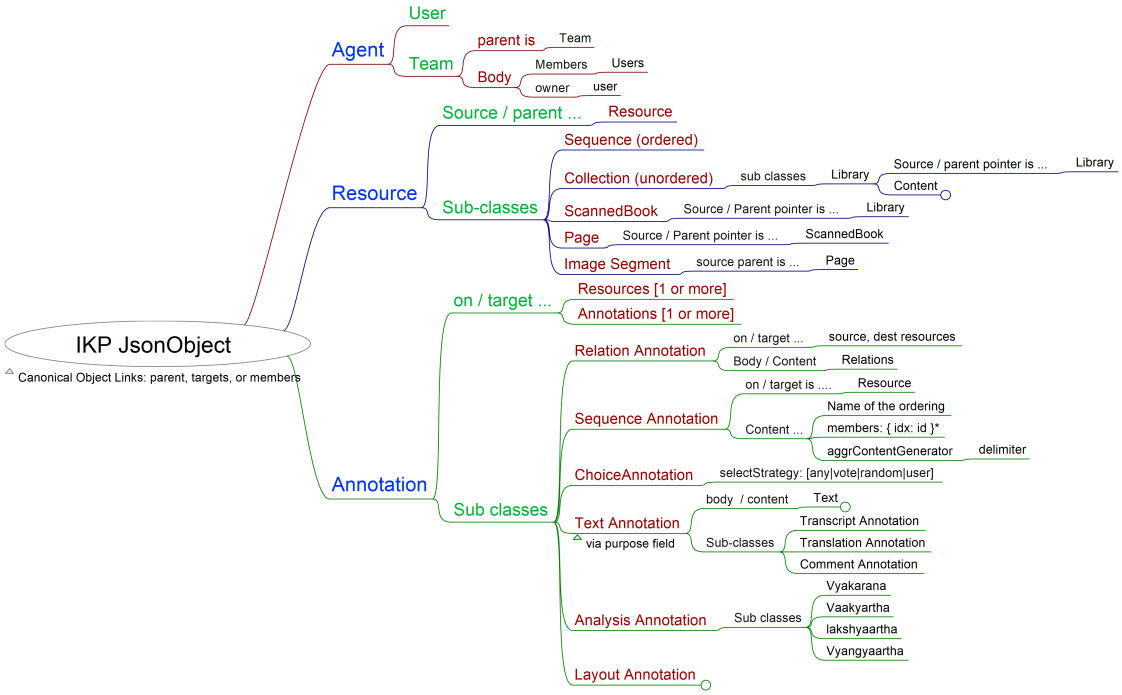

Figure 6: Class Hierarchy of Vedavaapi Objects.

- linkChildren: allows linking child resources to this resource.
- updateLinks: allows re-parenting a resource, re-targeting an annotation or changing the members of a collection
- updateAcls: allows updating the access control list of this object.

Vedavaapi uses an ID card approach to authorizing user operations on objects. A user can be part of multiple teams. Hence a user "carries" i.e., inherits the IDs of all the teams to which one belongs.

A Vedavaapi access control list (ACL) is a persistent attribute of the object and applies to all objects - user and teams objects as well as resources and annotations. Moreover, resources inherit the ACLs from their parent resources and annotations inherit ACLs from the objects they target. The ACL comprises three lists for each operation type:

- Granted IDs: the list of user and team IDs that are allowed this operation
- Revoked IDs: the list of user and team IDs that are not allowed this operation.
- Prohibited IDs: the list of user and team IDs that are prohibited this operation

A wildcard '*' in a list matches any ID. Vedavaapi authorizes user operations on objects using ACLs as follows: a user is allowed an operation on an object if at least one of the IDs one possesses is allowed that operation, and none of the IDs is prohibited that operation.

Access control based on ID cards avoids the need to check a user for team membership at access control time, which happens frequently. ACL inheritance offers a convenient and intuitive way for administrators to control access to large object networks. Prohibited IDs feature allows quarantining a user or team in an emergency security breach situation.

As a concrete example, if management of a library and its book collection needs to be delegated to a team, that team can be given update ACLs for the library's child resource hierarchy while

revoking the updateLinks operation to prevent the team from changing how the library connects to the parent document collection.

5.5 Vedavaapi API Overview

Table 1 outlines the APIs exported by Vedavaapi site to client applications. It consists of user and team management, authentication, object store access and ACL management. The APIs mainly support create, read, update and delete (CRUD) operations on various resources. In addition, the uniform object model of Vedavaapi allows a single API to manage diverse object types while also providing a powerful bulk operation interface on object graphs for efficiency. Specifically, the object store offers a versatile graph traversal API that not only is used for retrieving object networks but also upload or modify them. The query for graph traversal takes an attribute-based selection criterion to pick the initial objects and a list of hop criteria to guide the navigation from those objects to others via selected links.

API Cluster	APIs
Accounts	OAuth login and portable access tokens
~~Accounts~~	CRUD operations on users and teams
Object Store	CRUD operations on objects (resources, annotations, schemas, services)
~~Object Store~~	Object graph traversal, queries, updates and deletes
ACLs	CRUD operations on ACLs for given resource
~~ACLs~~	resolve permissions for currently logged in user

Table 1: Vedavaapi API Overview

6 Implementation and Evaluation

We have implemented most of the core Vedavaapi functionality in Python using Flask web services framework to provide RESTful API access. The object store is implemented as a python wrapper around MongoDB. The wrapper provides schema validation, user access control and multi-hop navigational queries on the raw objects stored in MongoDB database. We have implemented the Vedavaapi web dashboard as a standalone AngularJS application that can connect to multiple Vedavaapi sites via their API.

The objective of the Vedavaapi platform is to facilitate leveraging existing tools to rapidly create larger and effective IKP workflows. To evaluate how well our architecture achieves this objective, we have repackaged several existing open-source and private software modules to create an image-to-text conversion pipeline for scanned Indic documents - both printed and handwritten ones. Unlike existing OCR solutions, our solution enables human intervention to compensate for machine errors as well as OCR retraining for improved effectiveness. To do so, we have ported the following existing tools to run as IKP services in the Vedavaapi ecosystem:

- IIIF Book importer: This service imports layout and page information of scanned books uploaded to large digitized archives including https://archive.org/. We wrote a python library with a Flask API to import an entire scanned book from archive.org from its url as a Vedavaapi resource hierarchy. This way, we can offer IKP services on scanned books stored elsewhere. This took a couple of days of development effort, as Vedavaapi object schema was expressive enough to incorporate their metadata. Figure 7 shows a screenshot of a book imported via this service.

- Mirador Book Annotator: Then we ported a sophisticated open-source book viewer and annotator web application (written in JavaScript) called Mirador to operate on Vedavaapi-hosted books. We achieved this by using our Vedavaapi client-side adapter library in JavaScript as a plugin to Mirador to source its book information and serve it from our site. Mirador has a built-in annotation facility that lets users manually identify text segments

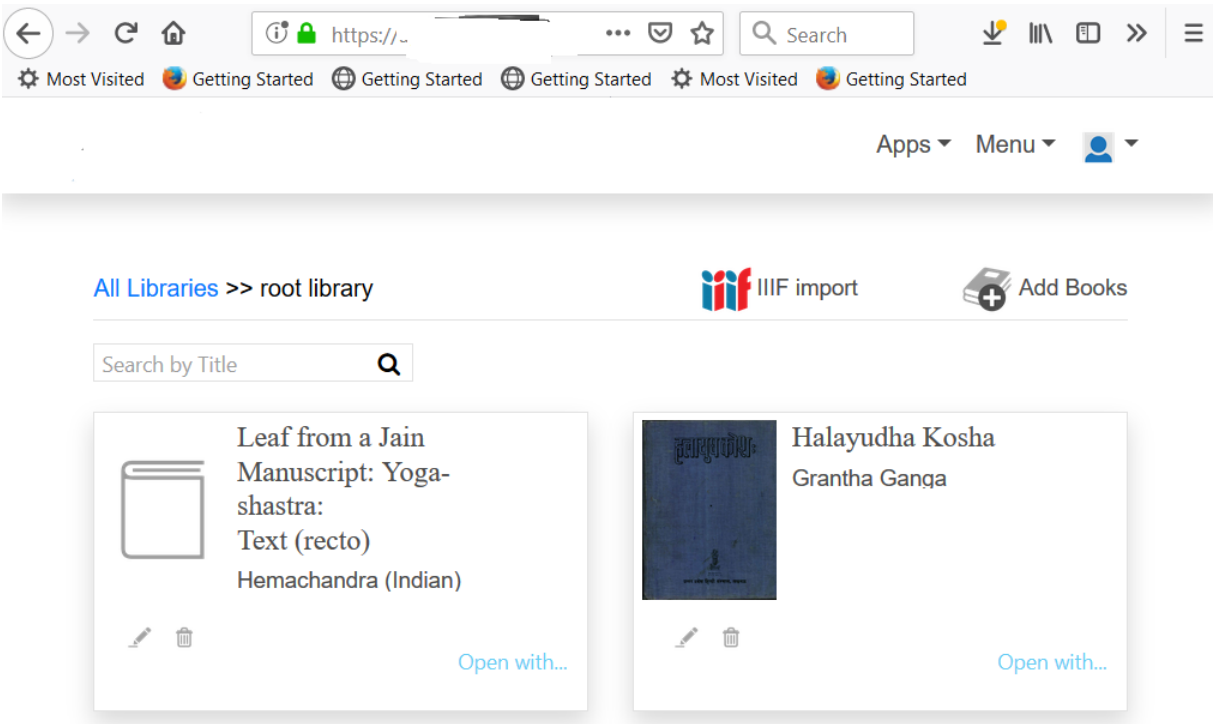

Figure 7: Screenshot of Vedavaapi library view showing books imported from archive.org via IIIF importer.

and also optionally transcript the text. We added persistence by storing those annotation on Vedavaapi backend site. This took one week of effort.

- Indic OCR Tools: OCR tools such as Tesseract and Google Vision API service provide both segmentation as well as text recognition from images in an XML-based standard format called hOCR. We created a wrapper service around them to import and export hOCR formatted data as annotations in Vedavaapi. We added a plugin to Mirador to invoke a user-selected OCR service to pre-detect words of a scanned page. This took one week of effort and greatly helped jumpstart text conversion for many printed texts available publicly.

- hOCR Editor: We ported an open-source web-based text editor for HOCR-formatted output to ease user experience in text conversion compared to Mirador. With our hOCR importer and exporter libraries already in place, this step took a day of effort, mainly to persist edits incrementally on Vedavaapi site. Figure 8 shows a screenshot of a post-OCR editing session. We imported an 800-page book called "Halayudha Kosha" from archive.org using IIIF importer application into Vedavaapi. We then invoked Tesseract OCR on the 10th page. We opened the OCR output using the hOCR editor as shown in the figure.

With these applications integrated with Vedavaapi platform, we got a complete solution for text conversion of archive.org books using OCR tools as well as crowd-sourced human correction working within 2 weeks. However, the layout detection of existing OCR tools on hand-written palm leaf manuscripts is poor due to irregular and overlapping lines in such documents. In parallel, a research group at IIIT Hyderabad developed a deep-learning-based layout detector for palm leaf manuscripts called Indiscapes Prusty et al. (2019) that automatically draws polygons around lines of text, holes, images and other artifacts by training on manual shape annotations. It requires a machine with GPU for the training step.

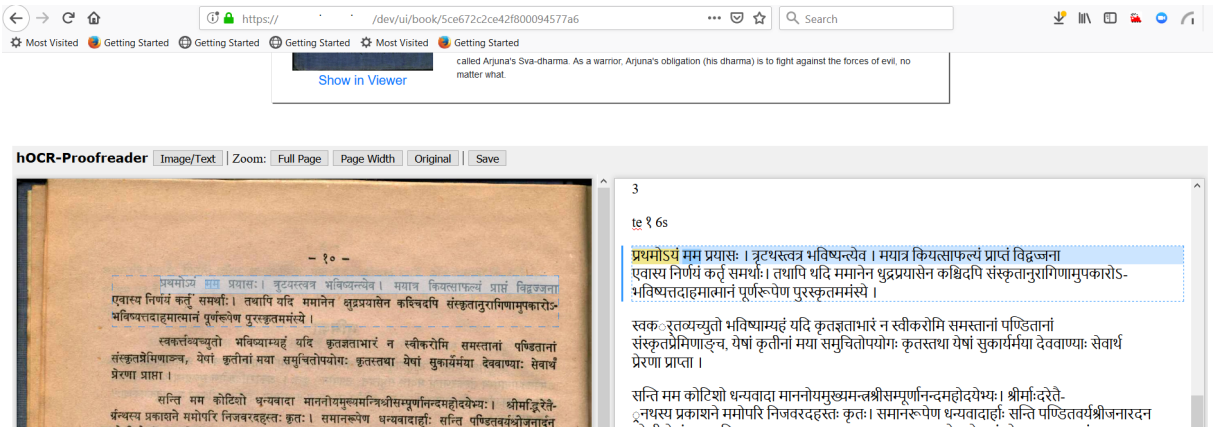

Figure 8: hOCR Editor running within Vedavaapi dashboard for proofreading Tesseract OCR output on a printed page from archive.org. The original image is shown on the left and the word editor is on the right. The yellow is corrected word.

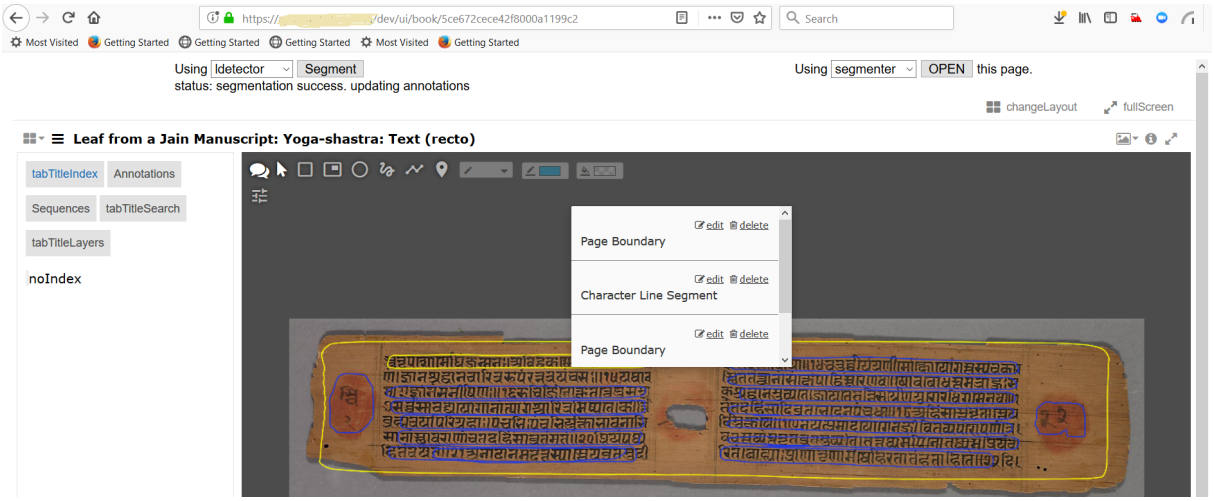

Figure 9: Palm leaf manuscript's lines detected with IIIT Hyderabad's palm leaf layout detector and edited through Mirador viewer within Vedavaapi dashboard. The blue contours around the lines were auto-detected and labeled by the tool as "Character Line Segments".

- Layout Detector for Palm leaf Manuscripts: Hence we have created a palm leaf layout detector based on IIIT Hyderabad tool. It takes a page image URL from a Vedavaapi site, detects line segments and posts them back as annotations to that page on Vedavaapi with empty text label. The training model file is maintained at IIIT Hyderabad, while the detector runs as Vedavaapi service. Subsequently, we were able to use the hOCR editor to type the text manually, thereby creating a crowd-sourced workflow for online transcription of hand-written text. Porting the tool to Vedavaapi took 2 days of effort as most of the functionality was in place. Figure 9 shows the screenshot of this service running from within Vedavaapi dashboard.

- Samsaadhanii Linguistic Toolkit: We are currently in the process of incorporating Samsaadhani toolset as an IKP service to be invoked on Vedavaapi-hosted Samskrit text data. This will test Vedavaapi's ability to leverage community-sourcing to eliminate ambiguity in linguistic analysis output. This is still a work in progress.

7 Lessons Learnt and Future Directions

Our experience with devising and leveraging the Vedavaapi platform to create IKP workflows indicates that a carefully designed object model that takes the data needs of existing tools can greatly enhance the ability to reuse these tools in providing useful end-to-end IKP solutions. While many of the design choices we had made got validated through the OCR pipeline experiment, we need to work on incorporating the higher order linguistic analysis tools to fully validate the design. During this journey of developing the IKP platform, we realize that there are a lot of popular, well-designed tools already developed and used in different contexts. To really facilitate widespread adoption of such a platform, it should be simple to adapt them to fit into its ecosystem.

Hence the next steps in this effort would be to incorporate tools for text segmentation, Samskrit linguistics and knowledge mapping to pave the way for a robust, popular platform for innovation around Indic knowledge.

8 Conclusion

In this paper, we made the case for ensuring interoperability of tools and services to accelerate the pace of Indic knowledge processing. While numerous point solutions exist, we have identified that the lack of end-to-end systems approach hinders rapid progress in this field. We present a novel platform approach to IKP architecture that combines the best practices of scale-out cloud computing, careful metadata design and flexible security protocols to significantly accelerate progress in this field.

References

Docker. 2019. Docker: Enterprise Container Platform. https://www.docker.com/.

Google. 2019. Google Cloud Vision API. https://cloud.google.com/vision/.

Pawan Goyal, Gérard Huet, Amba Kulkarni, Peter Scharf, and Ralph Bunker. 2012. A distributed platform for Sanskrit processing. In 24th International Conference on Computational Linguistics (COLING), Mumbai.

Oilver Hellwig. 2009. Extracting dependency trees from sanskrit texts. Sanskrit Computational Linguistics 3, LNAI 5406, pages 106–115.

Oliver Hellwig. 2019. Sanskrit OCR. http://www.sanskritreader.de/.

Gérard Huet. 2002. The Zen computational linguistics toolkit: Lexicon structures and morphology computations using a modular functional programming language. In Tutorial, Language Engineering Conference LEC'2002.

Amba Kulkarni. 2016. Samsaadhanii: A Sanskrit Computational Toolkit. http://sanskrit.uohyd.ac.in/.

Anil Kumar. 2012. Automatic Sanskrit Compound Processing. Ph.D. thesis, University of Hyderabad.

MongoDB. 2016. MongoDB NoSQL Database. http://www.mongodb.com/.

MongoDB. 2019. Internet Archive: Digital Library of free and borrowable books. http://www.archive.org/.

NAMAMI. 2012. Performance Summary of the National Mission for Manuscripts, New Delhi, India. http://namami.org/Performance.htm.

NAMAMI. 2016. National manuscript mission, new delhi, india. http://namami.org/.

Abhishek Prusty, Sowmya Aitha, Abhishek Trivedi, and Ravi Kiran S. 2019. Indiscapes: Instance segmentation networks for layout parsing of historical indic manuscripts. In Accepted for publication in ICDAR 2019.

JSON Schema. 2019. JSON Schema Standard. http://json-schema.org/.

Tesseract. 2019. Tesseract OCR. https://opensource.google.com/projects/tesseract.