Rāmopākhyāna a Web-based reader and index

Peter M. Scharf
President, The Sanskrit Library
Adjunct Professor, IIIT Hyderabad
scharf@sanskritlibrary.org
and
Dhruv Chauhan
International Institute of Information Technology, Hyderabad

Abstract

Web-based interactive readers integrated with indices provide scholars of Sanskrit a learning environment more responsive to their needs. Advances in Web technology and more stringent security requirements rendered the *Kramapāṭha* reader and indices developed for the *Rāmopākhyāna* two decades ago obsolete and inoperative. A current implementation using state-of-the art encoding and Web-development software revives the reader and its indices.

Keywords: Sanskrit, Mahābhārata, Rāmāyaṇa, text-encoding, XML, TEI

1 Introduction

The story of Rāma is one of the most popular stories in India. Its most ancient extant version is told in Vālmīki's *Rāmāyaṇa*. More than 35 other major Sanskrit works retell it, as do numerous versions in every modern Indian language and those of Southeast Asia. It spread to Tibet, China, and Japan in ancient and medieval times and continues to be appreciated in India in modern performances and television productions. Before the final redaction of the *Rāmāyaṇa* in about 25,000 verses, however, it was summarized in the *Rāmopākhyāna* in the great epic *Mahābhārata* in 704 verses in eighteen chapters comprising chapters 258–275 of the *Āraṇyaka Parvan*. There it is introduced in chapter 257 after the Pāṇḍavas' rescue of their wife Kṛṣṇā from abduction by Jayadratha when the exiled king Yudhiṣṭhira who asks the sage Mārkaṇḍeya who could be more unfortunate than he. Mārkaṇḍeya draws implications for the Pāṇḍavas from Rāma's successful defeat of Rāvaṇa and return to Ayodhyā with the help of the alliances he formed during his exile.

Scharf (2002) produced a Sanskrit reader of the *Rāmopākhyāna* that provides everything a student of Sanskrit would need for the precise understanding of each verse. The text of each verse occurs as in the critical edition (Sukthankar and others, 1933 1959) with minor revisions as described by Scharf (Forthcoming, §IIA) in Devanāgarī and is supplied with a close prose English translation. The explanation of each verse includes Roman transliteration; analysis of sandhi; identification of inflection; a glossary that includes the analysis of compounds, and nominal and verbal derivates; prose Sanskrit sentences paraphrasing the verse, and syntactic and cultural notes. Scharf (2005) and the introduction to the book describe the features of the book in detail. A second edition (Scharf, Forthcoming) will appear imminently.

Simultaneous with the preparation of the first edition of the *Rāmopākhyāna*, Scharf prepared an interactive digital reader and detailed digital indices called *Kramapāṭha*. *Kramapāṭha* consisted of two integrated apps that ran in the browser. The Web-based reader included sound in addition to the features of the printed book. The Web-based indices allowed one to filter a list of lemmata by lexical, inflectional, and text extent parameters, and to see related inflected forms, definitions, and the passages in which they occurred. An index of Pāṇinian sūtras was also included. Created before the availability of Indic Unicode Fonts, *Kramapāṭha* displayed Devanāgarī directly on the users screen using a dynamic font.

Proceedings of the Computational Sanskrit and Digital Humanities, 18^{th} *World Sanskrit Conference 2023*, pages 146–154

Great progress has been made over the past twenty years since the publication of *Rāmopākhyāna* and the development of *Kramapāṭha* in making resources available in digital form on the Web for scholars and students of Sanskrit. High quality Indic Unicode fonts are now widely available. The extensible markup language (XML) has become ubiquitous and integrated with hypertext markup language (HTML). JavaScript has been enriched and made easy to use by the development of numerous programming packages. JavaScript Object Notation (JSON), a lightweight data-interchange format is isomorphic with XML and integrated with powerful databases. However, along with the progress of the last two decades came restrictions. Enhancement of Web security rendered our Web-based version of the *Rāmopākhyāna* and its fabulous indices deployed in the Kramapāṭha reader and index applets obsolete several years ago. The present paper describes the redesign of the *Rāmopākhyāna* source data and reimplementation of the Web-based *Kramapāṭha* that utilizes recent advances in text encoding and Web technology.

2 Data description

The original source data for the *Kramapāṭha* reader was stored in a set of plain ASCII text files with tab-delimited fields that used special characters to mark the position for non-ASCII characters to be displayed in HTML or print such as single and double curly quotes and the root sign, and to designate the beginning and end of Sanskrit text such as proper names in English text fields. Sanskrit text was encoded in a subset of what became the Sanskrit Library Phonetic basic encoding (SLP1) described by Scharf and Hyman (2011). Several years ago Scharf wrapped the tab-delimited fields in these text files in XML tags to check and protect data-validity. Over the past several months he transformed all the data to an XML format that conforms to the Text-Encoding Initiative guidelines by writing XSLT routines and using TEITagger (described by Scharf (2018) and wrote Document Type Definitions (DTD) and Schematron schemas to validate the data. The TEI source now consists of the following parallel files:

1. text.xml. The text of the *Rāmopākhyāna* in SLP1 encoding.
2. ansan.xml. The text with sandhi analyzed.
3. morph.xml. Morphological analysis of each word.
4. morphCheck.xml. The above with the lemma of verbs with preverbs in a single element and the Unicode character entity for the check mark between the preverbs and root.
5. morphEng.xml. The English translation of each word in morph.xml or morphCheck.xml.
6. prose.xml. Sanskrit prose paraphrases of each verse.
7. proseSandhi.xml. The above with sandhi applied.
8. notes.xml. Syntactic and cultural notes.
9. english.xml. The English translation of verses.

Scharf had written a Pāṇinian sandhi program in Pascal in 1991 and revised it in 2000. Jim Funderburk rewrote the program in Perl, Java, and Python since then, maintains these versions in a Git archive at `https://github.com/funderburkjim/ScharfSandhi`, and described them in the `README` file there. In the past couple of months, Scharf revised the Java version to default to certain combinations more commonly expected and to eliminate spaces between preverbs and finite verbs. He used an XSLT routine that called his latest revision of the Java sandhi program to reproduce the sandhied versions of the Sanskrit verses (text.xml) and Sanskrit prose sentences (proseSandhi.xml) from the original sandhi-analyzed source files (ansan.xml and prose.xml). Another XSLT routine produces the file morphCheck.xml from morph.xml.

The data in all of these files is encoded in XML in accordance with the Text-Encoding Initiative Guidelines (Consortium, 2007). The elements and attributes used for basic text encoding are described in instructions on the Search and Retrieval of Indic Texts project (SARIT) website (`http://sarit.indology.info`) as well as by (Scharf, 2018), (Scharf, 2019 2020), (Scharf, 2015), (Scharf, 2017). We review them briefly here. Every file encodes its data in the `body` element which is a child of the `text` element within the `TEI` root element. A single `div` element occurs as a child of the `body` element with a texttttype attribute value set to `parvan`,

and an `n` attribute set to `3`. The data of each chapter is encoded in a `div` element with the textttttype attribute value set to `aDyAya`, and an `n` attribute numbering the chapter according to its numbering in the *Āraṇyaka Parvan*. Each chapter groups the verses of each speaker in an `sp` element, the first child of which is a `speaker` element containing phrases such as `जनमेजयः उवाच।`, and whose subsequent children are the verses attributed to that speaker. Each verse is encoded in an `lg` element and each line of verse in an `l` element. Each `speaker` element and `lg` element has an `n` attribute that numbers the verse within the adhyāya and an `xml:id` that uniquely identifies the passage within the text by a combination of the parvan number, the adhyāya number within the parvan, and the verse number. Since an `xml:id` is required to begin with a letter, we use `m` for *Mahābhārata*. The `speaker` element of each speech includes an `s` to distinguish its `xml:id` from that of the first verse. For example, `m3.257.1s` is the value of the `xml:id` of the `speaker` element at the beginning of adhyāya 257.

The two files text.xml and ansan.xml containing the original text in SLP1, and its sandhi analysis respectively, are encoded with these elements with exactly parallel numbering and `xml:id`s. The file english.xml is nearly parallel with the exception that it uses the `p` element for the prose translation of each verse and child `s` elements for sentences within it. The `xml:id` of each `p` element is identical to that of the corresponding verse in the other two files. The files prose.xml, proseSandhi.xml and notes.xml do likewise. In the English and notes files, Sanskrit text is encoded in a `foreign` element and single and double curly quotes, ellipsis, and the root sign are rendered with the appropriate Unicode entity values.

In the files morph.xml and morphCheck.xml, each element containing Sanskrit text, i.e. the `l` and `speaker` elements, encodes each Sanskrit word in a `w` element. The root of each inflected verb and the stem of each nominal is encoded as the value of the `lemma` attribute of the `w` element. Its lexical identifier is encoded as the value of the `w` element's `type` attribute, and its inflectional identifier is encoded as the value of its `subtype` attribute. Each word `w` element is numbered within the verse `lg` element or the `speaker` element in the value of the `n` attribute and provided with an `xml:id` that uniquely identifies it. The `xml:id` is formed by adding `.w` and the number to the end of the `xml:id` is ancestor `lg` element or parent `speaker` element. For example, the value of the word, janamejayaḥ, which is the first word of the `speaker` element of the first speech of adhyāya 257 is `m3.257.1s.w1`. Morphemes of words are encoded in an `m` element as children of the `w` element that contains the word in its `text` node, and each is supplied with a `type` attribute whose value is its lexical identifier. If a morpheme is an inflected word whose nominal termination has not been deleted in an *aluk-samāsa*, its `m` element is supplied with a `baseForm` attribute whose value is its stem, and with a `subtype` attribute whose value is its inflectional identifier. The analytic paraphrase (*vigraha-vākya*) of a compound is encoded in a `seg` element, supplied with a `type` attribute with the value `vigraha`, as the first child of the `w` or `m` element that contains the compound. These `m` and `seg` elements are all supplied with `n` attributes that number them and `xml:id` attributes with unique ids. In the case of submorphemes of morphemes, these numbers and ids show the hierarchy of the morpheme relative to its ancestor `w` element. For example, in the first verse in the morphCheck.xml file, the preverb *prāpya* has the morpheme *pra√āp* with the `xml:id` with the value `m3.257.1.w4.1`. This morpheme has two children: the preverb *pra* and the root *√āp* which have the `xml:id`s with the values `m3.257.1.w4.1.1` and `m3.257.1.w4.1.2` respectively. The `m` element may contain child `seg` elements that separate the markers from roots and augments and similar technical grammatical units classified by appropriate values of the `type` attribute. These are not considered separate morphemes and are not enumerated or supplied with `xml:id`s. Additional elements such as `bibl` and its child `biblScope` identify references to Pāṇini's *Aṣṭādhyāyī*, Amarasiṁha's *Nāmaliṅgānuśāsana*, and other works. The English translation of each word or morpheme, or each analytic paraphrase of a compound, is encoded in the morphEng.xml file in exactly the same structure.

3 Indices and their production

Given the high degree of analysis of the words in the *Rāmopākhyāna* undertaken in the preparation of the *Kramapāṭha* reader, and the intention to make it useful for learning Sanskrit, it was deemed useful and practical to make the text accessible from various angles by the construction of detailed indices. The original index included an index of lemmata consisting of a list of all the lexical bases of nominal and verbal forms as well as their morphemes, and an index of Pāṇinian sūtras. The new indices include a word index in addition. The lemma index can be limited by a number of parameters. First of all, it can be limited to the lemmata of whole words that occur in the text, excluding their morphemes. The list can also be limited by selecting any combination of lexical categories, inflectional categories, and extents of text. For example, one can search for all masculine nominative singular perfect active participles. One finds eight instances, five of which are the form *jaghnivān* from the root *han* 'slay', one of which is the same with the preverb *ā*, and the remaining two of which are the form *sameyivān* from the root *i* with the preverbs *sam* and *ā* 'come together'. If one limited the search to chapter 263 in addition, one would find just one instance: *ājaghnivān*. It is also possible to show the list of all occurring inflected forms of the lemma of any inflected word or the immediate derivatives of any lemma including morphemes.

In order to make the data for these indices accessible for a Web-based index, Scharf wrote XSLT routines to reassemble the data from the files morphCheck.xml and morphEng.xml in three index files: stems.xml, words.xml, and sutras.xml. The first and principal index file is like a dictionary: it gives the lemma, its lexical identifier, and its English translation. Yet it also includes a list of the derivates of that lemma, a list of its inflected forms, and a list of verses in which the lemma occurs including the `xml:id` of the word or morpheme of each occurrence. Each of the inflected forms also includes a list of the verses in which that form occurs with their `xml:id`s of each occurrence. By looking up a derivate in the same index file, one can find the same information for that lemma and so recursively access all of the derivates of any morpheme.

The word.xml file is somewhat similar but simpler. The entry for each word includes its lemma, its lexical identifier and its inflectional identifier, its English translation, and a list of the verses in which it occurs with the `xml:id` of each occurrence. The sutra.xml file has Pāṇinian sūtra numbers as its main entries. An additional file has a list of sūtras with their numbers as in the *Kāśikā*. Each entry in the sutra.xml file includes a list of the lemmata to derive which the sūtra was cited, each of which has a list of the verses in which that lemma occurs, including the `xml:id` of each occurrence. This arrangement allows one to find the lemma in the stems.xml index file if desired. Each entry also includes a the inverse: a list of the the verses in which that sūtra is cited each of which indicates the lemma to derive which it is cited with the `xml:id` of each occurrence. This arrangement permits easy access to the instances of the citation of the sūtra.

4 Web-interface implementation

Using the Oxygen XML editors XML to JSON converter, Scharf produced JSON versions of the reader files:

1. ansan.json
2. english.json
3. morphCheck.json
4. morphEng.json
5. notes.json
6. prose.json
7. proseSandhi.json
8. text.json

and of the index files:

1. stems.json
2. sutras.json
3. sutraOnly.json
4. words.json

Chauhan used the MongoDB database to house these data, Elasticsearch as a secondary database, and the Flask and VueJS application development frameworks. The MondoDB database is a schema-less database in which one can have any type of data in separate documents and the flexibility to store data of different types. ElasticSearch permits speedy, complex searches. Chauhan migrated the data into both databases and used the RESTful APIs made available in the Flask framework to develop the interaction between Elasticsearch and the frontend framework VueJS. The latter is a progressive framework for building user interfaces, promotes modular application structure and provides practical ways to build component-based, dynamic user interfaces where one can manipulate and manage DOM elements in powerful ways. Figure 1 shows a diagram of the architecture of the system.

The *Kramapāṭha* reader display allows one to show any of the information one wishes to see regarding a verse: text, transliteration, sandhi analysis, prose, notes, and translation, by clicking buttons at the right of the window. Figure 2 shows the display of the first two and last three of these six types of information. By clicking any word in the Devanāgarī or Romanization, one can display the information relevant to that word in stages. In the first stage the word's inflectional identifier and lemma are shown. In the second stage its lexical identifier and English translation are shown, and in the third stage its morphemic analysis is shown. Figure 3 shows the display of the first speaker and verse in adhyāya 257 with its Roman transliteration in which the user has clicked the word *naravyāghrāḥ* three times to show all three stages of information relevant to the word.

The index display allows one to select the list of whole words, morphemes, or inflected forms. By selecting one word in the list at the left, its occurrences are shown in the middle. At the right is an interface in which one can select lexical and morphological categories and text extents by which to limit the list at the left. Figure 4 shows the list of inflected forms selected in the index. Here the word *adrśyam* is selected and its list of occurrences is shown in the middle with its inflectional identifier, its lemma, and its lexical identifier. Beneath this is shown the verses in which it occurs and its English translation. At the right is shown the pane to limit the list by inflectional identifier with the nominal subpane selected but no parameters within it chosen.

Figure 5 shows the list of the lemmata of whole words that occur in the text, here called *free morphemes*. The lemma *antar* has been selected and the list of verses in which it occurs, either independently, as part of a compound, or as a gati with a verbal root, is shown in the middle. At the head of that list is shown the selected lemma in bold with its lexical identifier in parenthesis. Immediately below is shown the independent word in which it occurs and its verse number. Below that is shown the three derivates in which the lemma occurs, and below them is shown the list of verses in which these forms occur. At the tail end of the list is shown its English translation and beneath that the sūtra cited in the derivation of *antarhita*. The pane at the right shows the interface to limit the list by lexical category. Here the pronominal sub-pane has been selected but no parameters within that pane have been selected. At the bottom of the right pane is shown the interface to limit the list by text extent. The very first and last verses of the text are shown by default.

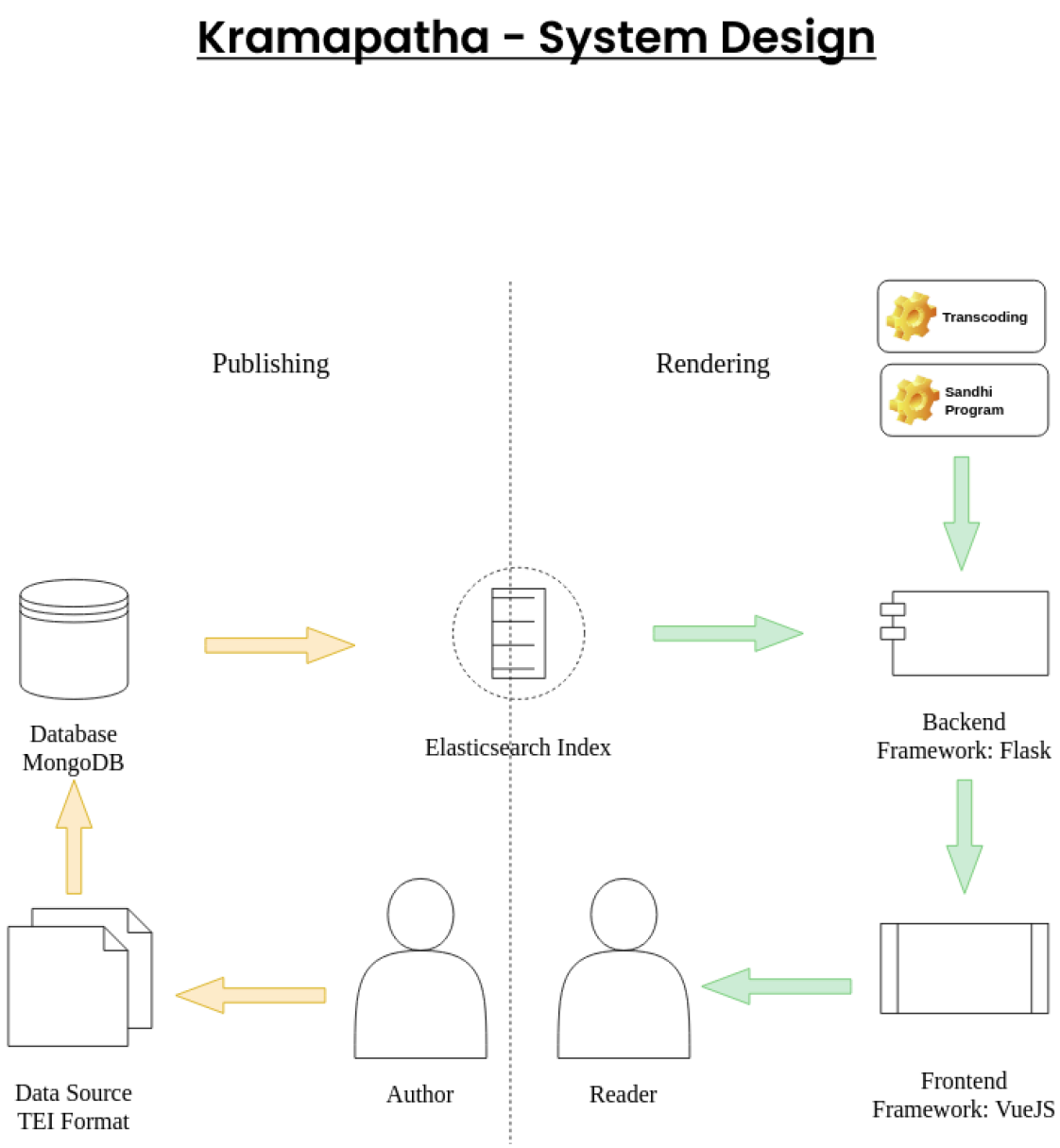
We are still refining the display of items in the interface and their arrangement. The second index figure shown at present appears crowded with too much information of different types shown at once. We expect to have the display finished with a few weeks work.

5 Conclusion

The interactive reader and index permit scholars of Sanskrit to access the information they need as they need it. A printed text can only present a single view, so the printed edition of the

Rāmopākhyāna presents all the information for each verse on a page at once. The Web-based reader, in contrast, more effectively facilitates learning by hiding information until desired by the reader. While the printed text can supply limited indices in appendices, the Web-based indices dynamically interact with the text display to permit the user to navigate the text easily.

Figure 1: The architecture of the *Kramapāṭha* reader and index system design



References

TEI Consortium, editor. 2007. *TEI P5*. TEI Consortium.

Peter M. Scharf. 2022. *Rāmopākhyāna—the story of Rāma in the* Mahābhārata. The Sanskrit Library, 2 edition.

Peter M. Scharf and Malcolm D. Hyman. 2011. *Linguistic issues in encoding Sanskrit*. The Sanskrit Library.

Figure 2: The *Kramapāṭha* reader showing the text, Roman transliteration, prose Sanskrit sentences, notes, and translation

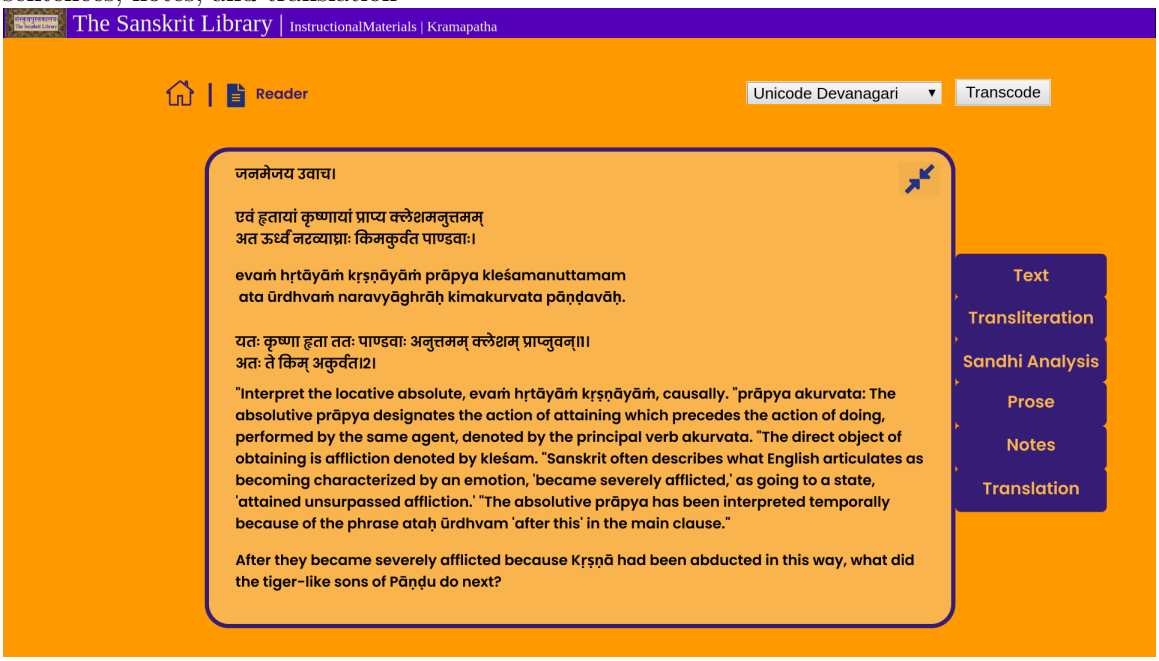


Figure 3: The *Kramapāṭha* reader showing the text, Roman transliteration, and information relating to the word *naravyāghrāḥ*

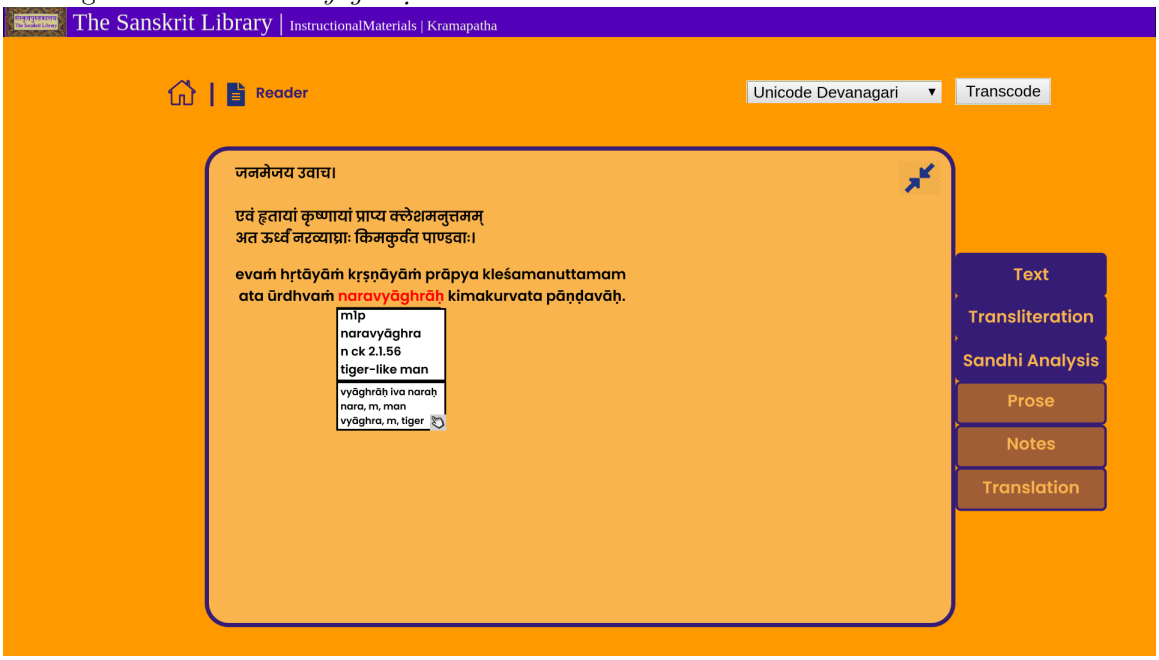


Figure 4: The *Kramapāṭha* inflected forms index

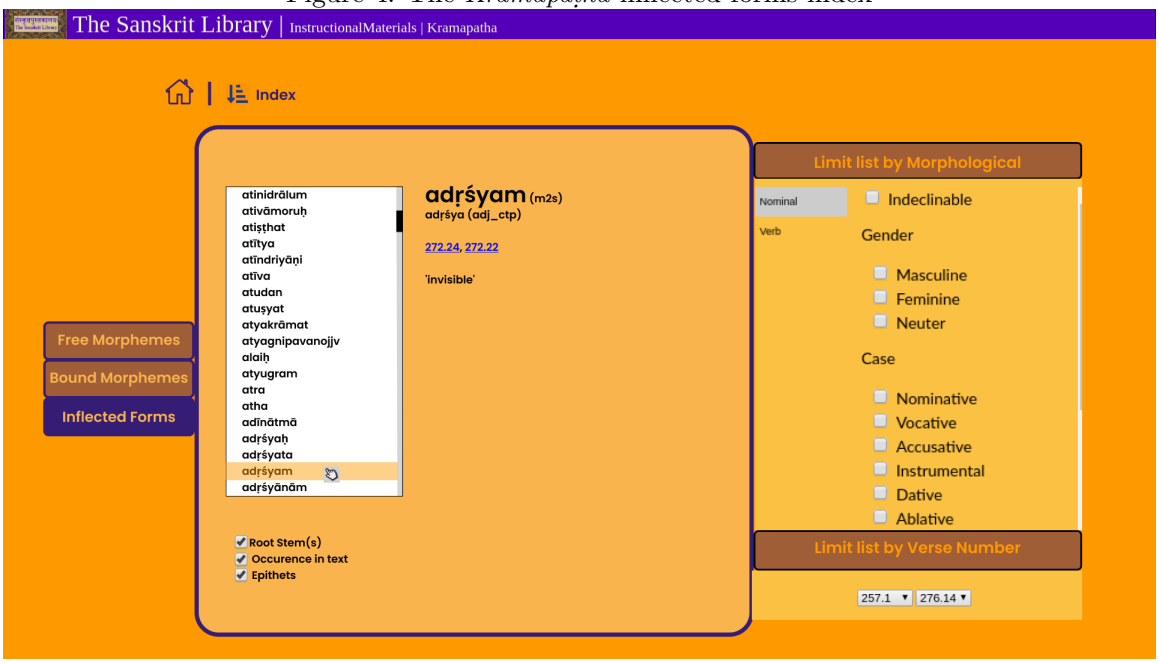
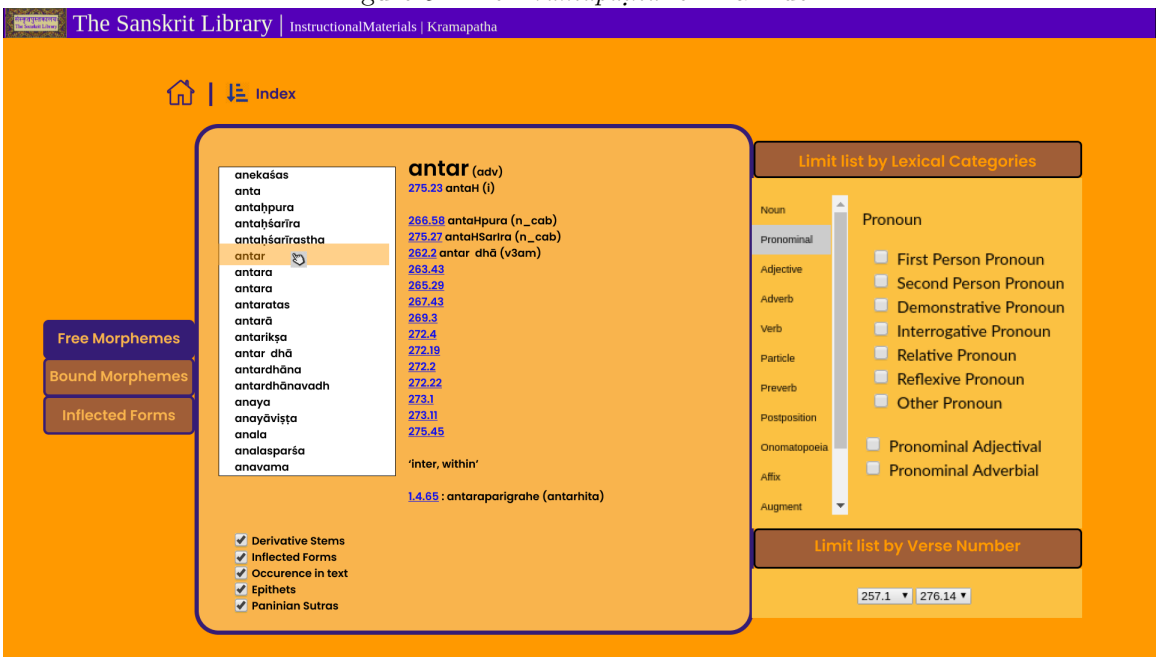


Figure 5: The *Kramapāṭha* lemma index



Peter M. Scharf. 2002. *Rāmopākhyāna—the story of Rāma in the* Mahābhārata. RoutledgeCurzon, 1 edition.

Peter M. Scharf. 2005. Rāmopākhyāna: the story of Rāma in the *Mahābhārata*. In T. S. Rukmani, editor, *The* Mahābhārata, pages 49–60. Munshiram Manoharlal. Paper presented at the International Conference on the *Mahābhārata*, 18–20 May 2001, Concordia University, Montreal, Quebec, Canada.

Peter M. Scharf. 2015. Providing access to manuscripts in the digital age. In Justin Thomas McDaniel and Lynn Ransom, editors, *From Mulberry leaves to silk scrolls*, number 1 in The Lawrence J. Schoenberg Studies in Manuscript Culture, pages 231–271. University of Pennsylvania Libraries.

Peter M. Scharf. 2017. Accessing manuscripts in the digital age. volume 6, pages 131–173. National Mission for Manuscripts.

Peter M. Scharf. 2018. Teitagger. In Gérard Huet and Amba Kulkarni, editors, *Computational Sanskrit and Digital Humanities, selected papers presented at the World Sanskrit Conference, University of British Columbia, Vancouver, 9–13 July 2018*, pages 169–191. DK Publishers Distributers.

Peter M. Scharf. 2019-2020. Issues in digital sanskrit philology and computational linguistics. 80:347–375.

Vishnu Sitaram Sukthankar et al., editors. 1933–1959. *The Mahābhārata*. Bhandarkar Oriental Research Institute. Vol. 4, *The Āraṇyakaparvan (part 2)*,1942.