TEITagger: Raising the standard for digital texts to facilitate interchange with linguistic software

Peter M. Scharf

Abstract: For several years, members of the International Sanskrit Computational Linguistics Consortium working to facilitate interchange between digital repositories of Sanskrit texts, and digital parsers and syntactic analyzers have recognized the need to standardize reference to particular passages in digital texts. XML has emerged as the most important standard format for document structure and data interchange, and TEI as the most important standard for the XML markup of textual documents. TEI provides methods to precisely describe divisions in texts from major sections to individual morphemes, and to associate various versions with each other. Responsible text archives, such as TITUS and SARIT, have adopted the TEI standard for their texts. After a workshop to train doctoral candidates at the Rashtriya Sanskrit Sansthan to mark-up texts in accordance with TEI in May 2017, the Sanskrit Library developed software to semi-automate the process with extensive use of regular expressions and meter-identification software, and is currently marking-up all of its texts using the TEITagger. The result will be a large repository of digital Sanskrit texts that can furnish text to the Sanskrit Heritage parser and the University of Hyderabad's parser and syntax analyzer to allow passages parsed and analyzed for dependency structure to be interlinked with their originals.

1 XML and TEI

In the age in which oral productions and hand-written documents were the predominant mode of expressing knowledge and exchanging information, each individual articulation or manuscript had its own format determined by the author and heard or read by other individuals. In the age of the print medium, presses produced multiple copies of individual productions which could be widely distributed to numerous other individuals. At the outset of the digital age, as Scharf and Hyman (2011: 2) and Scharf (2014: 16) noted, presentation of individual productions imitated the print medium. Document creators and software engineers created works to present knowledge to human readers. As Goldfarb (1990) noted, unfortunately the tendency persists as "their worst habits" as if their production were meant only for human eyes, and had no need to coordinate with software developed by others. In 1969, however, Goldfarb, Mosher, and Lorie at International Business Machines Corporation (IBM) developed the Generalized Markup Language (GML), so called based on their initials (Goldfarb 1990: xiv), to mark up documents in terms of the inherent character of their constituents, such as prose, header, list, table, etc., to enable software to format the documents variously for various devices, such as printers and display screens, by specifying a display profile without changing the document itself (Wikipedia contributors 2017). Over the next decade, Goldfarb and others developed the international Standard Generalized Markup Language (SGML), International Standards Organization (ISO) document 8897, to describe documents according to their structural and other semantic elements without reference to how such elements should be displayed. Thus in contrast to the HyperText Markup Language (HTML) which was designed to specify the display format of a text, SGML separates the inherent structure of a document from how it is presented to human readers and "allows coded text to be reused in ways not anticipated by the coder" (Goldfarb 1990: xiii).

The eXtensible Markup Language (XML) is an open-source metalanguage consisting of a stripped-down version of SGML formally adopted as a standard by the World Wide Web Consortium (W3C) in February 1998. In the couple of decades since, XML has become the single most important standard format for document structure and data interchange. Wüstner, Buxmann, and Braun (1998) noted, "XML has quickly emerged as an essential building block for new technologies, offering a flexible way to create and share information formats and content across the Internet, the World Wide Web, and other networks." Benko (2000: 5) noted, "XML is expected to become the dominant format for electronic data interchange (EDI)." A few years ago, Zazueta (2014) noted, "XML emerged as a front runner to represent data exchanged via APIs early on;" whereas "Javascript Object Notation (JSON), emerged as a standard for easily exchanging Javascript object data between systems." He continues,

> API designers these days tend to land on one of two formats for exchanging data between their servers and client developers - XML or JSON. Though a number of different formats for data have been designed and promoted over the years, XML's built in validation properties and JSON's agility have helped both formats emerge as leaders in the API space."

Benko (2000: 2) also noted that two of the seven benefits the W3C defines for establishing XML include the following:

- Allow industries to define platform-independent protocols for the exchange of data.
- Deliver information to user agents in a form that allows automatic processing after receipt.

As a simple metalanguage consisting of just seven characters (`<`, `>`, `/`, `=`, `"`, `'`, `␣`), XML allows users to develop markup languages of an unlimited variety. In order to facilitate interchange of textual documents, the Text Encoding Initiative (TEI) developed a community-based standard for the representation and encoding of texts in digital form. The TEI Guidelines for Electronic Text Encoding and Interchange define and document a markup language for representing the structural, renditional, and conceptual features of texts. They focus (though not exclusively) on the encoding of documents in the humanities and social sciences, and in particular on the representation of primary source materials for research and analysis. The Text Encoding Initiative also makes the Guidelines and XML schema that validate them available under an open-source license. TEI has become the most important standard for the XML markup of textual documents. Hence to facilitate the interchange, cross-reference, and unanticipated use of digital Sanskrit text, it is imperative that digital archives of Sanskrit texts make their texts available encoded in XML in accordance with the TEI Guidelines.

2 Sanskrit digital archives and the use of TEI

A number of organizations and individuals, such as GoogleBooks, The Million Books Project, Archive.org, the Digital Library of India, and the Vedic Reserve at Maharishi International University, have made images and PDF documents of Sanskrit printed texts available, and a number of libraries, such as the University of Pennsylvania in Philadelphia and the Raghunath Temple Sanskrit Manuscript Library in Jammu, have made images of their Sanskrit manuscripts available. Such productions have greatly facilitated access to primary source materials; yet that access is limited exclusively to being read by a human being. Although Jim Funderburk developed software to search headwords in a list and highlight that headword in digital images of dictionary pages, and Scharf and Bunker developed software to approximate the location of passages in digital images of Sanskrit manuscripts, the results of such software are also merely displays for a human reader. PDFs do not facilitate automatic processing after receipt.

Numerous groups and individuals of various backgrounds have created digital editions of Sanskrit texts and made them available on portable digital storage media and the Web. As opposed to image data, these documents consist of machine-readable character data. Most of these are structured in simple data structures, such as lines of text numbered with a composite chapter-section-line number, in text files or directly in HTML files. These documents are intended to permit access by a human to passages by searching as well as for sequential reading. While the various providers of digital text are too numerous to mention, one site has emerged as a central registry. The Göttingen Register of Electronic Texts in Indian Languages (GRETIL) lists about eight hundred such Sanskrit texts. These texts are openly available for download so that others may subject them to various sorts of linguistic processing such as metrical, morphological, and syntactic analysis. As great a service as making these texts available in digital form is, GRETIL exerted minimal discipline on its early contributors so that there is great variability in the specification of metadata. In many cases, the source edition of the text is unknown. In addition, each contributor was free to structure the document as he wished, so there is great variability in the manner of formatting verse and enumerating lines.

Although GRETIL offers the texts in a few common standard encodings including UTF8 Unicode Romanization, there is variability in how the contributors employed capitalization, encoded diphthongs versus contiguous vowel sequences, punctuation, etc. Texts available from other sources use Devanāgarī Unicode, different ASCII meta-encodings, or legacy pre-Unicode fonts. Scharf and Hyman (2011) and Scharf (2014) have already dealt with the issues regarding character encoding. Here I address higher-lever text and document structure encoding.

Even by 2006, at the start of the International digital Sanskrit library integration project, the Thesaurus Indogermanischer Text- und Sprachmaterialien (TITUS), which contributed its texts for integration with dictionaries produced by the Cologne Digital Sanskrit Dictionaries project via morphological analysis software produced by Scharf and Hyman at Brown, had begun partially using TEI tags to mark up the structure of its texts and metadata. Over the past four years, the Search and Retrieval of Indic Texts project (SARIT) marked up all of the texts which had previously been made available in various ad hoc formats at the Indology website, and some twenty additional texts, in a consistent encoding in accordance with the TEI standard. The site (`http://sarit.indology.info`) currently houses fifty-nine Sanskrit TEI documents made available under a Creative Commons license and provides clear instructions for how to mark up Sanskrit texts in accordance with TEI.

3 TEI training

At the bequest of the SARIT project, in an initial attempt to spur large-scale encoding of Sanskrit texts in accordance with the TEI standard, I conducted a one-week e-text tutorial at the Rashtriya Sanskrit Sansthan's Jaipur campus in February 2010. While several participants produced TEI versions of small portions of texts, the workshop failed to instigate the collaboration of technical expertise and abundant Sanskrit-knowing labor that SARIT had hoped. In May 2017, however, I was invited by the Rashtriya Sanskrit Samsthan to conduct a two-week TEI workshop at its Ganga Nath Jha campus in Allahabad. There I trained twenty Sanskrit doctoral candidates in how to encode texts and catalogue manuscripts in accordance with TEI Guidelines. In an additional week I worked with these students to encode twenty Sanskrit works in accordance with TEI, ten of which were delivered complete in the next month.

During the workshop, I trained students to analyze the structure of a plain text data-file with Sanskrit text in numbered lines or verses and to construct regular expressions to recognize strings of text with fixed numbers of syllables. We constructed regular expressions to recognize a few common verse patterns and had the students submit the verses found to the Sanskrit Library's meter analyzer produced and described by Melnad, Goyal, and Scharf (2015a,b). Once we knew that verses with a certain number of syllables were typically in a certain metrical pattern, we constructed replacement expressions to transform the recognized pattern to well-formed TEI line group elements (`lg`) with subordinate line (`l`) and segment elements (`seg`) for each verse quarter (*pāda*) and to insert type, analysis, and metrical pattern attributes (`type, ana, met`) in the (`lg`) tag. The replacement expressions inserted the enumeration provided by the source document in (`n`) and (`xml:id`) attributes in the (`lg`) tag, and typed and lettered the verse quarters as well. Where complex numbers compiled the numbers of text divisions, subdivisions, and passages within subdivisions, the regular expression placed just the last in a separate group, and the replacement expression inserted that number in the value of the `n` attribute while putting the whole number in the value of the `xml:id` attribute. For example, the regular expression and replacement expression shown in Figure 1 was primarily responsible for transforming the following verse of the *Bhagavadgītā* (in Sanskrit Library ASCII encoding) to the well-structured TEI (`lg`) element with its subsidiaries shown in Figure 2:

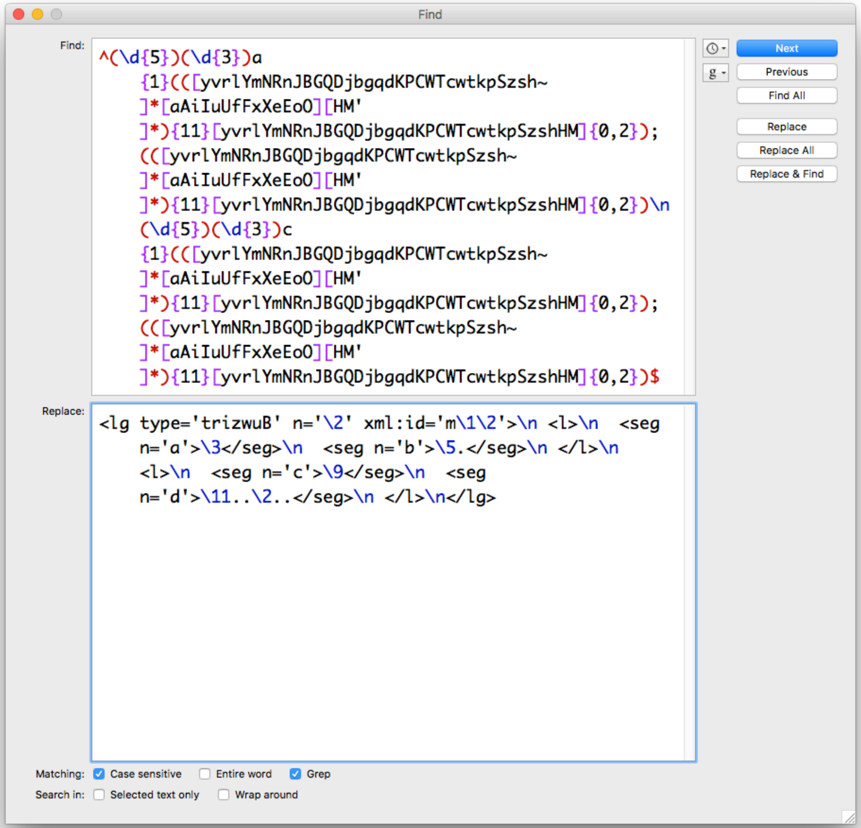
> 06024070a ApUryamARam acalapratizWaM; samudram ApaH praviSanti yadvat
> 06024070c tadvat kAmA yaM praviSanti sarve; sa SAntim Apnoti na kAmakAmI

I say, “primarily responsible,” because in fact the leading zeroes on the number of the verse were captured by this regular expression so that ‘070’ was inserted in the value of the `n` attribute; an additional regular expression removed them.

Now one will notice that the original text document conveniently indicated the break between the two verse quarters in each line of a Triṣṭubh verse by a semicolon and space. This indication allowed the regular expression to group just the text of each verse quarter without leading or trailing spaces. However, no such indication was given for the break between verse quarters in an Anuṣṭubh verse because there is frequently no word-break at the pāda boundary of the ubiquitous śloka. One would want to preserve the information whether or not there is a word break there, yet would not want a pāda to begin with a space. Hence after a regular expression inserted each verse quarter in a `seg` element, subsequent regular expressions moved leading spaces, where found, from the beginning of the second `seg` to the end of the first and set the second verse quarter on a separate line. Thus the first verse of the *Bhagavadgītā*,

Figure 1

Regular expression and replacement expression to transform a plain text verse in Triṣṭubh meter to TEI



Find:

```
^(\d{5})(\d{3})a
    {1}(([yvrlYmNRnJBGQDjbgqdKPCWTcwtkpSzsh~
    ]*[aAiIuUfFxXeEoO][HM'
    ]*){11}[yvrlYmNRnJBGQDjbgqdKPCWTcwtkpSzshHM]{0,2});
    (([yvrlYmNRnJBGQDjbgqdKPCWTcwtkpSzsh~
    ]*[aAiIuUfFxXeEoO][HM'
    ]*){11}[yvrlYmNRnJBGQDjbgqdKPCWTcwtkpSzshHM]{0,2})\n
    (\d{5})(\d{3})c
    {1}(([yvrlYmNRnJBGQDjbgqdKPCWTcwtkpSzsh~
    ]*[aAiIuUfFxXeEoO][HM'
    ]*){11}[yvrlYmNRnJBGQDjbgqdKPCWTcwtkpSzshHM]{0,2});
    (([yvrlYmNRnJBGQDjbgqdKPCWTcwtkpSzsh~
    ]*[aAiIuUfFxXeEoO][HM'
    ]*){11}[yvrlYmNRnJBGQDjbgqdKPCWTcwtkpSzshHM]{0,2})$
```

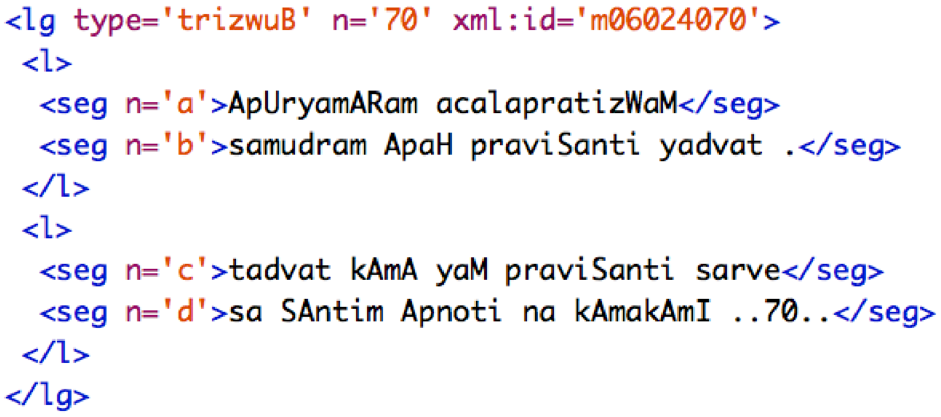
Replace:

```
<lg type='trizwuB' n='\2' xml:id='m\1\2'>\n <l>\n  <seg
    n='a'>\3</seg>\n  <seg n='b'>\5.</seg>\n </l>\n
    <l>\n  <seg n='c'>\9</seg>\n  <seg
    n='d'>\11..\2..</seg>\n </l>\n</lg>
```

Matching: [✓ Case sensitive] Entire word [✓ Grep]

Search in: Selected text only Wrap around

Figure 2
Bhagavadgītā 2.70 in Triṣṭubh meter


```
<lg type='trizwuB' n='70' xml:id='m06024070'>
 <l>
  <seg n='a'>ApUryamARam acalapratizWaM</seg>
  <seg n='b'>samudram ApaH praviSanti yadvat .</seg>
 </l>
 <l>
  <seg n='c'>tadvat kAmA yaM praviSanti sarve</seg>
  <seg n='d'>sa SAntim Apnoti na kAmakAmI ..70..</seg>
 </l>
</lg>
```

06023001a Darmakzetre kurukzetre samavetA yuyutsavaH
06023001c mAmakAH pARqavAS cEva kim akurvata saMjaya

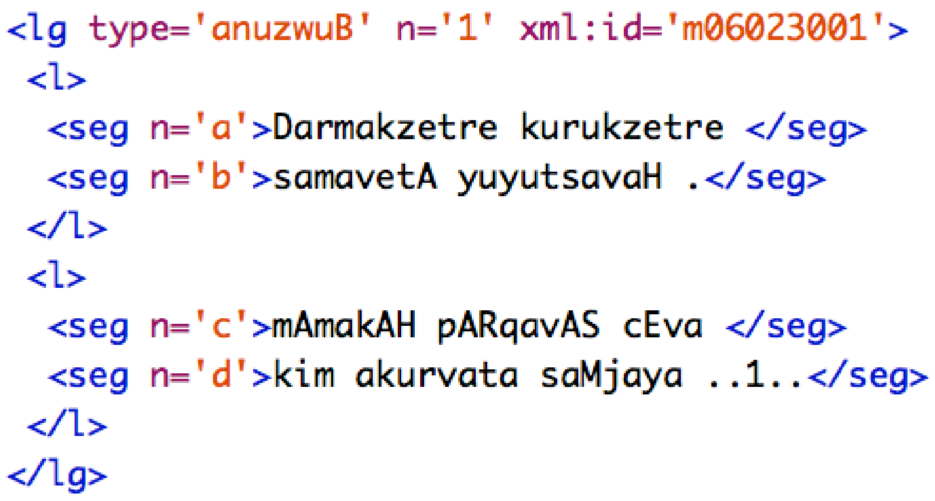
was marked up in TEI and reformatted as shown in Figure 3 with each verse quarter in a separate `seg` element.

I also trained students in the workshop to compose regular expressions to capture the speaker lines such as *Dhrtarāṣṭra uvāca* that introduce speeches and to compose replacement expressions to put these in `speaker` elements. Similarly, I taught them to mark up prose sentences and paragraphs in `s` and `p` elements, to put speeches in `sp` elements, to insert `head` and `trailer` elements, to locate and capture enumeration of divisions, to insert `div` elements, to insert the whole in `body` and `text` elements, to insert page and line break elements, and to mark up bibliography. I then had them insert these elements in a teiHeader template in the `TEI` element, and to validate the complete TEI document. Figure 4 shows the first short speech of the *Bhagāvadgītā* with the `speaker` element in the context of parent `sp`, `div`, `body`, and `text` opening tags. Let me remark that guidelines for how to mark up Sanskrit text in accordance with TEI are conveniently available on the SARIT website.[1]

[1] `http://sarit.indology.info/exist/apps/sarit-pm/docs/encoding-guidelines-simple.html`

Figure 3

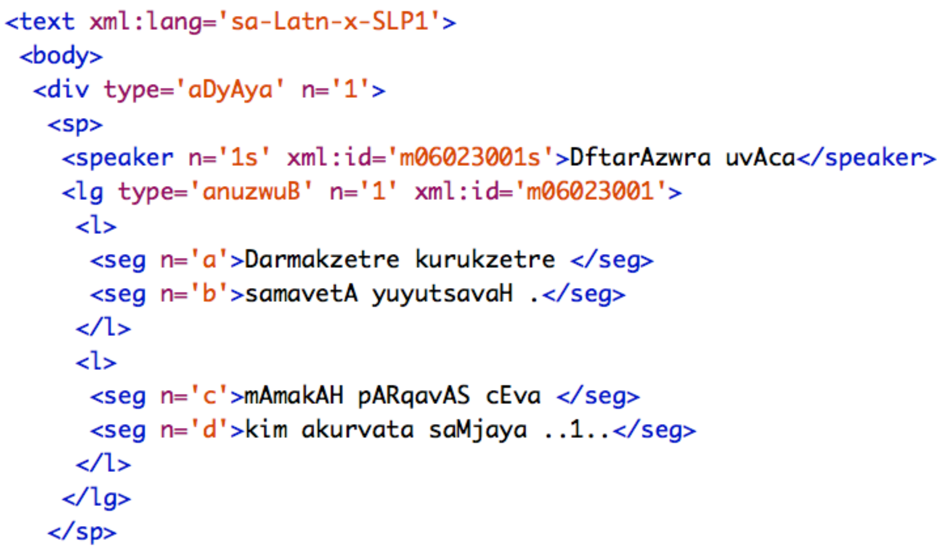
Bhagavadgītā 1.1 in Anuṣṭubh meter



```
<lg type='anuzwuB' n='1' xml:id='m06023001'>
 <l>
  <seg n='a'>Darmakzetre kurukzetre </seg>
  <seg n='b'>samavetA yuyutsavaH .</seg>
 </l>
 <l>
  <seg n='c'>mAmakAH pARqavAS cEva </seg>
  <seg n='d'>kim akurvata saMjaya ..1..</seg>
 </l>
</lg>
```

Figure 4

TEI markup of a speech in the context of division, body, and text elements



```
<text xml:lang='sa-Latn-x-SLP1'>
 <body>
  <div type='aDyAya' n='1'>
   <sp>
    <speaker n='1s' xml:id='m06023001s'>DftarAzwra uvAca</speaker>
    <lg type='anuzwuB' n='1' xml:id='m06023001'>
     <l>
      <seg n='a'>Darmakzetre kurukzetre </seg>
      <seg n='b'>samavetA yuyutsavaH .</seg>
     </l>
     <l>
      <seg n='c'>mAmakAH pARqavAS cEva </seg>
      <seg n='d'>kim akurvata saMjaya ..1..</seg>
     </l>
    </lg>
   </sp>
```

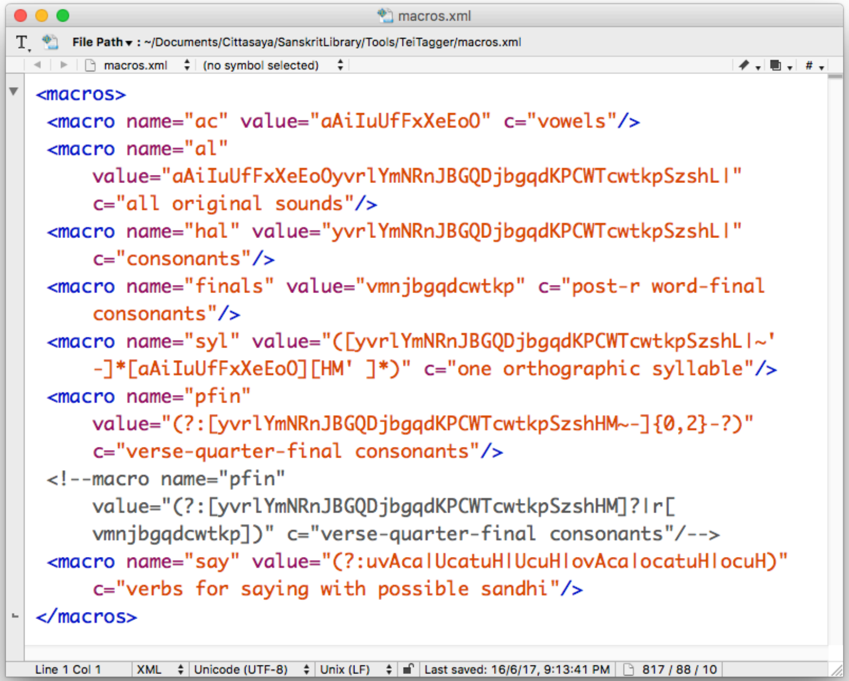
4 TEITagger software

After the experience of teaching Sanskrit students with minimal technical literacy to transform a plain text document to well-structured XML in accordance with TEI in a series of well-ordered steps, it occurred to me that I could also teach a machine to do the same. Ralph Bunker, the technical director of the Sanskrit Library, had previously developed software called Linguistic Mapper at my request so that I could compile a driver file that contained a sequence of regular and replacement expressions that implemented historical sound change rules between a proto-language and a descendant language. We created TEITagger by modifying Linguistic Mapper to process a series of such sets of regular and replacement expressions that matched specified numbers of syllables in certain arrangements that approximated metrical patterns. By creating a regular expression that counted the correct number of syllables per pāda we could convert every such verse to proper TEI markup in `lg` elements, with each line in an `l` element, and each pāda in a `seg` element. At the same time we could number the verse in an `n` attribute, insert an `xml:id`, and insert the presumed meter name and metrical pattern in a `type` attribute. The meter name and metrical pattern in the first version of TEITagger was presumed on the basis of the syllable count, not automatically checked against a pattern of light and heavy syllables.

We then revised TEITagger to include the feature of submitting a segment of text that matched a certain regular expression to our meter identification software that would identify the meter of a whole verse by checking the passage against specified patterns of light and heavy syllables as defined by classical metrical texts. If a match is found TEITagger version 2 automatically inserts the meter name, general type, and metrical pattern in `type`, `ana`, and `met` attributes of the `lg` element. To simplify the regular expression formulation in the command driver file for this program, we composed macros to represent vowels, consonants, syllables, syllable codas, and the typical terms used in the lines that introduce speeches. These macros are shown in Figure 5.

To further simplify testing segments of text for any meter type with any number of syllables, we introduced an iterative loop command and iteration variable in version 3. Thus, for example, with a command that consists of the single regular expression and replacement expression shown in Figure 6, TEITagger can evaluate every segment of text in a file with four verse quarters each consisting of n syllables per verse quarter, where the variable n is tested in order from 28–1 thereby testing for all of the verses with the same number of syllables per verse quarter. Metrical patterns with the same number of syllables per verse quarter include all 468 of the samavrtta and upajāti types as well as some of the ardhasamavrtta and viṣamavrtta type. Similar expressions can be composed to match verses with unequal numbers of syllables per verse quarter. Such metrical patterns include those of the ardhasamavrtta type and mātrāvrtta type as well as irregular variations of more regular patterns. The current version (17) also passes verse lines and individual pādas to the meter analyzer to detect their patterns in irregular verses.

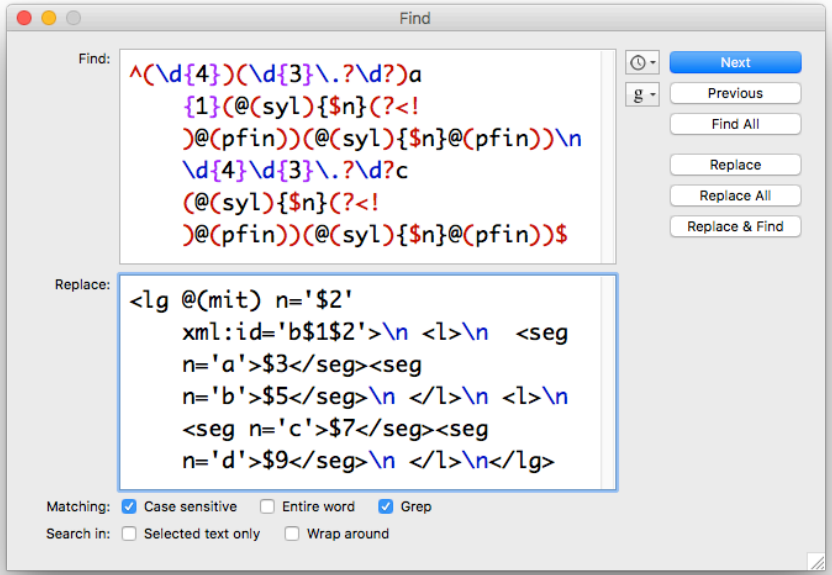
Figure 5
TEITagger macros


macros.xml
File Path ▾ : ~/Documents/Cittasaya/SanskritLibrary/Tools/TeiTagger/macros.xml
macros.xml
(no symbol selected)

```
<macros>
 <macro name="ac" value="aAiIuUfFxXeEoO" c="vowels"/>
 <macro name="al"
     value="aAiIuUfFxXeEoOyvrlYmNRnJBGQDjbgqdKPCWTcwtkpSzshL|"
     c="all original sounds"/>
 <macro name="hal" value="yvrlYmNRnJBGQDjbgqdKPCWTcwtkpSzshL|"
     c="consonants"/>
 <macro name="finals" value="vmnjbgqdcwtkp" c="post-r word-final
     consonants"/>
 <macro name="syl" value="([yvrlYmNRnJBGQDjbgqdKPCWTcwtkpSzshL|~'
     -]*[aAiIuUfFxXeEoO][HM' ]*)" c="one orthographic syllable"/>
 <macro name="pfin"
     value="(?:[yvrlYmNRnJBGQDjbgqdKPCWTcwtkpSzshHM~-]{0,2}-?)"
     c="verse-quarter-final consonants"/>
 <!--macro name="pfin"
     value="(?:[yvrlYmNRnJBGQDjbgqdKPCWTcwtkpSzshHM]?|r[
     vmnjbgqdcwtkp])" c="verse-quarter-final consonants"/-->
 <macro name="say" value="(?:uvAca|UcatuH|UcuH|ovAca|ocatuH|ocuH)"
     c="verbs for saying with possible sandhi"/>
</macros>
```

Figure 6

TEITagger iterative command to match verses with four pādas with n syllables per pāda, where an arbitrary range can be specified for n.



Find:
```
^(\d{4})(\d{3}\.?\d?)a
    {1}(@(syl){$n}(?<!
    )@(pfin))(@(syl){$n}@(pfin))\n
    \d{4}\d{3}\.?\d?c
    (@(syl){$n}(?<!
    )@(pfin))(@(syl){$n}@(pfin))$
```

Replace:
```
<lg @(mit) n='$2'
    xml:id='b$1$2'>\n <l>\n  <seg
    n='a'>$3</seg><seg
    n='b'>$5</seg>\n </l>\n <l>\n
    <seg n='c'>$7</seg><seg
    n='d'>$9</seg>\n </l>\n</lg>
```

Matching: [✓ Case sensitive] Entire word [✓ Grep]
Search in: Selected text only Wrap around

The TEITagger driver file also accepts commands to insert header and footer files so that one can add the opening XML file tags, open and close `body` and `text` tags, open and close `TEI` tags, and a `teiHeader`. Finally, TEITagger will pretty print the file if it is a valid XML file.

5 Philological use of the TEITagger software

Metrical analysis of Vedic, epic, and classical Sanskrit texts is not new. For instance, metrical analysis of the *Mahābhārata* has produced interesting results that bear on the critical composition of the text and its history. Edgerton (1939) distinguished regular versus irregular varieties of Triṣṭubh and Jagatī meters that were significantly divided between the *Virāṭaparvan* and *Sabhāparvan* respectively and thereby demonstrated separate composition and probably subsequent insertion of the *Virāṭaparvan* in the text of the *Mahābhārata.* He also described several regular patterns in the hypermetric and hypometric irregular varieties based upon the location of the caesura.

Fitzgerald (2006) reported the results of analyzing a database of the Triṣṭubh and Jagatī verses he assembled over the past couple of decades. He analyzed these metrical patterns into five segments: initial and final syllables, and three sets of three syllables each: the opening, break, and cadence. He identified three standard varieties of Triṣṭubh: (1) a regular Upajāti consisting of the alternating pādas of Indravajrā and Upendravajrā, (2) Śālinī, and (3) Vātormī; and a standard variety of Jagatī: an Upajāti consisting of alternating pādas of Vaṁśasthā and Indravaṁśā. Fitzgerald (2009) isolated two measurable variables: (1) the degree of uniformity among the pādas of the Triṣṭubh stanzas, and (2) the set of major Triṣṭubh features that were eliminated in the creation of the classical standard triṣṭubh. He isolated passages on the basis of runs of Triṣṭubh and Jagatī verses and measured the uniformity within verses in these passages to attempt to locate discontinuities that might signal different periods of composition of the passages. Fitzgerald (2004) argued, "if we are able to make reasonable arguments about historical fissures in the text, we thereby enrich our understanding of the text's possible meanings ...by distinguishing multiple voices, dialogical tension, and innovation within the otherwise synchronic, unitary, received text." In his careful unpublished study of the episode of the dice match, he was able to counter the conclusions of Söhnen-Thieme (1999), and to conclude that “this whole episode, the Upajāti passage of chapter 60 in which Duḥśāsana drags Draupadī into the sabhā by the hair, is likely later than most or all of the rest of this episode.”

Work of the sort that Edgerton and Fitzgerald have done with careful evaluation of statistics gathered with great effort over a long time could be vastly simplified and assisted by the automation provided by TEITagger. After testing TEITagger version 2 on the *Bhagavadgītā*, within a week, I tagged the entire critical edition of the *Mahābhārata*, including those with irregular patterns such as those with hypermetric or hypometric pādas. A driver file of nearly a thousand lines individually matched every possible combination of the syllable counts per pāda, triple-line and single line verses as well as the normal double-line verses. For example, a separate set of a regular expression and its replacement expression targets triple-line Triṣṭubh verses with a hypermetric first pāda, another targets such verses with a hypermetric second pāda, etc. The driver file assumed that such deviant metrical patterns ought to be classified under a certain type despite the failure of the meter analyzer to find a regular type. The task preceded and inspired the development of our iteration command and commands to send verse lines and pādas to the meter analyzer described in the previous section. The driver file I developed to tag the *Bhāgavatapurāṇa* with these features added consists of only 318 lines.

TEITagger version 2 tagged 73,436 verses and 1,057 prose sentences in 386 paragraphs. The verses include 68,860 Anuṣṭubhs, 2,970 Triṣṭubhs, 431 Jagatī, 322 Indravajrā, 0 Upendravajrā, 496 of the standard Upajāti variety alternating the two preceding, 88 Śālā, 78 Vāṇī (other Upajātis), 31 Aparavaktra (an ardhasamavrtta meter), 22 Praharṣiṇī, 16 Rucirā, 9 Mālinī, 4 Vasantatilakā, 4 Puṣpitāgrā, 1 Śārdūlavikrīḍita, 1 Halamukhī, 1 Āryāgīti (a type of Āryā), 1 mixture of half Kāmakrīḍā and half Kāmukī, and a hundred unidentified. The unidentified metrical patterns include for instance, 1 mixture of half Kāmukī and half unidentified, 1 mixture of a deviant pāda with subsequent Anuṣṭubh, jagatī, and Triṣṭubh pādas, as well as 98 other uninvestigated unidentified patterns.

The results of TEITagger version 2 are presented in Table 1 in comparison with some of the results Fitzgerald (2009) reported. One can see that there is a minor discrepancy of one passage in the enumeration of the prose passages. The cause of this discrepancy needs to be investigated. Yet otherwise there is astonishing consistency in the enumeration of the prose and verse passages. There is a discrepancy of just two verses of the Anuṣṭubh meter. The discrepancy of 41 Triṣṭubh/Jagatī verses and 52 fancy meters is probably largely due to TEITagger's incorrect assumption that a number of irregular meters with 11–12 syllables per pāda were of this type rather than fancy metrical patterns. For if the meter analyzer failed to identify a verse, TEITagger relied on syllable count alone to classify it.

Using TEITagger version 17 with the more refined feature of sending verse lines and quarters to the meter analyzer, and with some revision of the meter analyzer itself, I reevaluated the metrical patterns of the *Mahābhārata*. In this version, I made no assumptions about the conformity of deviant patterns to regular types; instead, where the meter analyzer failed to find a match for a verse, I permitted it to seek a match of each line of the meter, and failing to find a match for a line, to seek a match for each pāda in the line. Where lines or pādas within a verse were identified as the same, the metrical information was combined so that along with a single type classification for the verse only the deviant lines or pādas are classified separately. Labels consisting of the meter names in SLP1 for each different meter found within a verse are separated by a forward slash in the value of the `type`-attribute of the `lg`-element that contains the verse in the TEI file. These labels are preceded by letters indicating the pādas so labeled.

Table 2 shows the numbers of verses with one to six metrical identifications for the verse as a whole or parts of the verse individually. Table 3 shows the meters recognized. Column three of Table 3 shows the number of the meter indicated in column one that was recognized as a verse. Column four shows the number of additional sets of double lines recognized within triple-line meters. Column five shows the number of lines recognized in verses not recognized as verses or sets of double lines. Column six shows the number of pādas recognized in lines not recognized as lines. The first line of each section divided by double horizontal lines tallies the numbers of that general metrical type. Rows beginning with *Upajāti* in bold in the Triṣṭubh and Jagatī sections tally the numbers for the Upajāti type patterns listed in subsequent rows within the same section. The Upajāti numbers are included in the tally for the section as a whole as well. At the bottom of the table, the row labeled *Identified* in bold summarizes the total number of verses, additional pairs of lines, additional lines, and additional verse quarters recognized. The row labeled *No type* shows the number of verses not recognized before querying the meter analyzer regarding lines and pādas, and the total number of pādas that remain unidentified. The pādas that remain unidentified are provided with the label `no_type` within the value

Table 1
Metrical and non-metrical passages in the Mahābhārata identified by TEITagger v. 2 compared with those identified by Fitzgerald

passage type	**syllables/pāda**	**TEITagger**	**Fitzgerald 2009**
passages		73,822	73,821
prose			
paragraphs		386	385
sentences		1,057	
verse		73,436	73,436
Anuṣṭubh	8	68,860	68,858
Triṣṭubh/Jagatī	11–12	4,385	4,426
Triṣṭubh	11	2,970	
Indravajrā	11	322	
Upendravajrā	11	0	
Upajāti	11	662	
Indravajrā/Upendravajrā	11	496	
Śālā	11	88	
Vāṇī	11	78	
Jagatī	12	431	
Fancy meters		100	152
Halamukhī	9	1	
Aparavaktra	13/12	31	
Puṣpitāgrā	12/13	4	
Praharṣiṇī	13	22	
Rucirā	13	16	
Vasantatilakā	14	4	
Mālinī	15	9	
Kāmakrīḍā/Kāmukī	15/16	1	
Śārdūlavikrīḍita	19	1	
Āryāgīti	7 caturmātrās + 2	1	
unidentified		100	

of the `type`-attribute in the TEI file. No lines or line pairs are so labeled because if they are unidentified their pādas are sent to the meter analyzer individually for analysis. The row labeled *Total* in bold shows the total number of verses in the *Mahābhārata* in column three but in column six just the total number of pādas analyzed individually.

Table 2
Mixed metrical patterns in the Mahābhārata identified by TEITagger v. 17

type	**identified**	**not fully**	**total**
single	70,242	3,194	73,436
mixed	689	2,505	3,194
double	85	4	89
triple	468	994	1,462
quadruple	129	1,451	1,580
quintuple	5	23	28
sextuple	2	33	35

TEITagger version 17 found matches for each of the fourteen varieties of Triṣṭubh Upajāti patterns and the several Jagatī Upajāti patterns named separately. It also found several additional samavrtta metrical patterns for lines and verse quarters not found by analyzing whole verses. Rows headed by these meter names show blanks in the columns for verses and lines where no verses or lines of that type were found. These initial results of applying TEITagger to analyze the metrical patterns in the *Mahābhārata* demonstrate its capacity to reveal detailed information about a massive work and to mark up the results in a way that permits computational compilation so that these results may be presented to scholars in ways that may inspire further insight.

Table 3
Metrical patterns in the Mahābhārata identified by TEITagger v. 17

meter type	**syllables/ pāda**	**verse**	**2/3 lines**	**line**	**quarter**
Anuṣṭubh	8	68,360	10	521	633
Anuṣṭubh3	8	68,322	10	518	610
Pramāṇikā	8	38	0	1	22
Vidyunmālā	8			2	1

meter type	**syllables pāda**	**verse**	**2/3 lines**	**line**	**quarter**
Vibhā	8				6
Haṁsaruta	8				1
Triṣṭubh	11	1,355	62	970	3,252
Indravajrā	11	171	3	271	941
Upendravajrā	11	94	0	174	805
Vātormī	11	1	30	0	597
Rathoddhatā	11	5	0	0	0
Śālinī	11	38	0	0	909
Upajāti	11	1,046	29	525	0
Bhadrā	11	68	2	167	0
Haṁsī	11	90	0	188	0
Kīrti	11	114	3	0	0
Vāṇī	11	98	4	0	0
Mālā	11	73	1	0	0
Śālā	11	82	0	170	0
Māyā	11	50	3	0	0
Jāyā	11	50	1	0	0
Bālā	11	82	5	0	0
Ārdrā	11	68	3	0	0
Rāmā	11	62	1	0	0
Rddhi	11	85	3	0	0
Buddhi	11	67	2	0	0
Siddhi	11	57	1	0	0
Jagatī	12	411	4	94	343
Vaṁśasthā	12	359	3	73	181
Indravaṁśā	12	1	0	5	95
Bhujaṅgaprayāta	12	3	0	0	0
Kāmadattā	12				4
Vaiśvadevī	12			3	55
Śruti	12			2	8
Upajāti	12	48	0	16	0
Śaṅkhanidhi	12	1	0	2	0
Padmanidhi	12	2	0	14	0
Vaṁśamālā	12	45	1	0	0
Fancy		116	0	37	32

meter type	**syllables pāda**	**verse**	**2/3 lines**	**line**	**quarter**
Halamukhī	9	1	0	0	0
Śuddhavirāj	10				1
Aparavaktra	13/12	27	0	3	0
Puṣpitāgrā	12/13	33	0	3	0
Praharṣiṇī	13	8	0	1	1
Rucirā	13	28	0	11	28
Prabhavatī	13				1
Vasantatilakā	14	3	0	0	1
Praharaṇakalikā	14			1	0
Mālinī	15	9	0	0	0
Śārdūlavikrīḍita	19	1	0	0	0
Upagīti	5cm+l+1cm+g	6	0	29	0
Āryāgīti	7cm+gg	0	0	1	0
Identified		70,242	76	1,622	4,267
No type		3,194			4,297
Total		73,436			8,564

6 Communication between TEI files and linguistic software

As mentioned in section 1, one of the principal benefits of encoding Sanskrit texts using TEI XML is to fulfill the need to coordinate directly, without human intervention, with software developed by others, possibly in ways not anticipated. In particular, by encoding Sanskrit texts in TEI we anticipate coordinating a large repository of digital Sanskrit texts with parsers and syntax analyzers, such as the Sanskrit Heritage parser and the University of Hyderabad's संसाधनी. TEI provides robust standardized methods to coordinate various versions of texts and to refer to particular divisions and segments within a text so that parsed and syntactically analyzed passages may be interlinked with their originals. Naturally, the highest levels of coordination between versions would require standardized identification of the repository that houses the original file from which a passage was taken and submitted to a linguistic analysis tool on another site. An attribute value pair such as simply `repository='sl'`, or more officially `repository='US-RiPrSl'` using the International Standard Identifier for

Libraries and Related Organizations (ISIL), ISO 15511, might identify the Sanskrit Library as the repository. Obviously standardized identification of the file within the repository is required, either by collection and item identifiers or by filename. These identifiers should be interpretable programmatically as a URL, or be a URL directly provided with a submission. For example, if I submit the first verse of the unanalyzed text of the *Mahābhārata* to the Sanskrit Heritage parser I might provide the URL `http://sanskritlibrary.org/texts/tei/mbh1.xml` with my submission.

A second level of standardized identification is required to identify the type of analysis. When the Sanskrit Library analyzed the TITUS archive's texts for inclusion in 2006, it discovered a surprising variety in the degree and type of analysis of sandhi. Some of these encoding practices can be specified in the encoding description of a document. However, standard designation of various degrees of analysis is needed to coordinate versions. At the least, one might consider standard designation for the types of analysis of Sanskrit texts described in Table 4. For clarity, it is strongly recommended that these different degrees of analysis be located in separate files, not combined in a single file. TEI provides simple means of coordinating such versions by synchronizing element identifiers (`xml:id`).

Once a file containing the version of a text with a specific degree of analysis is identified, standardized reference to particular sections and passages is required. TEI provides machine-readable methods for declaring the element used and the structure of references within two elements of the `teiHeader`:

- `tagsDecl`
- `refsDecl`

The tagging declaration may be used to document the usage of specific tags in the text and their rendition.[2] Figure 7 shows the `tagsDecl` element used for the Sanskrit Library's TEI edition of the critical edition of the *Mahābhārata*. Because the value of the `partial` attribute is specified as false, the tags listed as values of the `gi` attribute of the `tagUsage` elements are all the elements and the only elements that occur under the text element. The `lg`, `l`, and `seg` elements are used to mark up verses as shown in figures 2, 3, and 4, in the last of which are shown also the use of the `body`, `div`, `sp`, and `speaker` elements. The `p` and `s` elements are used to mark up paragraphs

[2]See the TEI P5 guidelines at `http://www.tei-c.org/release/doc/tei-p5-doc/en/html/HD.html#HD57`, and `http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-tagsDecl.html`

Table 4
Degrees of analysis of Sanskrit texts

1. continuous text (*saṁhitā-pāṭha*)
   a. with breaks only where permitted in Devanāgarī script, i.e. only after word-final vowels, visarga or anusvāra
   b. with breaks where permitted in Roman script, i.e. after consonants as well
   c. with breaks where permitted in Roman script with designation immediately following characters representing sounds that result from single replacement sandhi at word boundaries
2. sandhi-analyzed text (*pada-pāṭha*)
   a. with word final visarga throughout, without designation of compound constituents
   b. distinguishing visarga originating in final *s* from visarga from final *r*
   c. with designation (but not analysis) of compound constituents as permitted in Devanāgarī script, i.e. after constituent-final vowels, visarga or anusvāra
   d. with designation (but not analysis) of compound constituents as permitted in Roman script, i.e. after constituent-final consonants as well
   e. with designation (but not analysis) of compound constituents as permitted in Roman script, with designation immediately following characters representing sounds that result from single replacement sandhi at constituent boundaries
   f. with analysis of sandhi between compound constituents as well
3. morphologically analyzed text
4. lexically and morphologically analyzed text
5. syntactically analyzed text
   a. dependency structure
   b. phrase structure

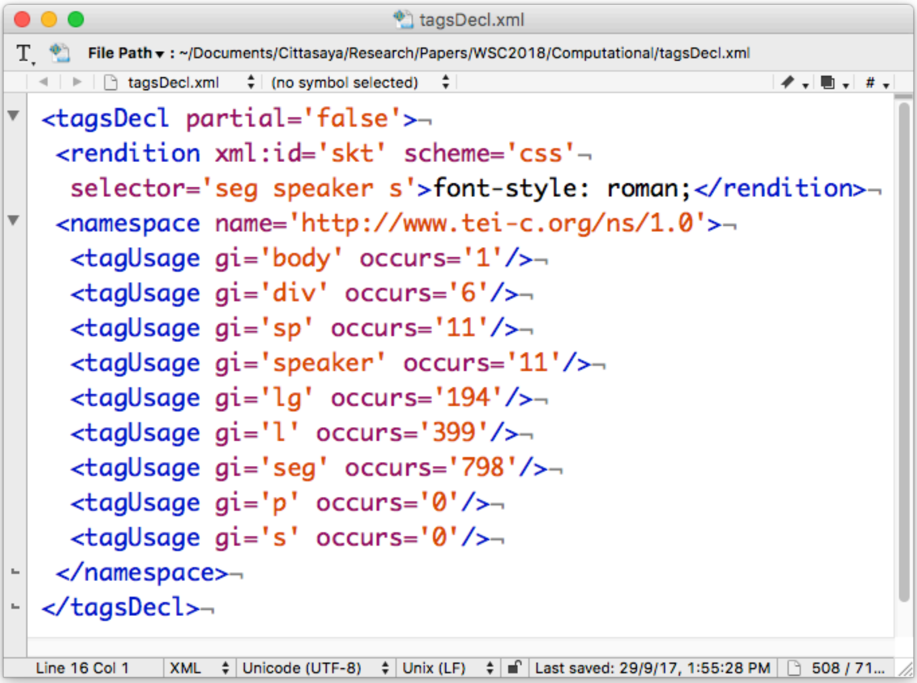
and sentences in prose. The numbers listed as values of the `occurs` attribute in the `tagUsage` elements indicate the number of occurrences of the element named in the value of the `gi` attribute. The numbers shown are those for the *Svargārohaṇaparvan*. Those mentioned as values of the `selector` attribute of the `rendition` element with `xml:id='skt'` are all the elements and the only elements that render Sanskrit text in SLP1 to be transcoded to Unicode Roman, Devanagari, or another Indic Unicode encoding for redisplay. These elements provide all that is necessary to extract Sanskrit text from the encoding for display in HTML, and for submission as a unit to metrical, morphological and syntactic analysis software. The attribute values of the elements listed in the `rendition` element with `xml:id='sktat'` lists all the attributes and the only attributes whose values are Sanskrit text in SLP1 to be transcoded. These attribute values are Sanskrit terms that might be used to display menus in an HTML display to select divisions such as parvan, and adhyāya.

The reference declaration describes the reference system used in the text.[3] TEI offers the possibility of describing the pattern of canonical references formally in a manner amenable to machine processing. A regular expression describing the pattern of the canonical reference is paired with a replacement expression that describes the path to the attributes that contain the referenced numbers (`n` attributes of `div` and `lg` elements in verse in the *Mahābhārata*, and of `p`, and `s` in prose). Figure 8 shows the `refsDecl` element of the Sanskrit Library's TEI edition of the *Svargārohaṇaparvan*. The pattern shown in the `matchPattern` attribute of the first `cRefPattern` element describes a canonical reference to any verse quarter in the *Mahābhārata*. The three sets of digits separated by periods refer to the parvan, adhyāya, and verse; the letter refers to the pāda, for example, 6.24.70a refers to the first pāda of the seventieth verse of the twenty-fourth adhyāya of the sixth parvan shown in Figure 2. (The 24th adhyāya of that parvan is the second in the *Bhagavadgītā*.) The first of the two `cRefPattern` elements gives a replacement expression that matches a path that has verses directly as children of a `div` element; the second, one that has verses as children of an intervening `sp` element within an adhyāya. Subsequent `cRefPattern` elements describe shorter references to whole verses, adhyāyas, and parvans. These elements and attributes directly provide an unambiguous method to

[3]See the TEI P5 Guidelines at `http://www.tei-c.org/release/doc/tei-p5-doc/en/html/HD.html#HD54`, and `http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-refsDecl.html`

Figure 7

The `tagsDecl` element in the Sanskrit Library's TEI edition of the Svargārohaṇaparvan of the Mahābhārata


tagsDecl.xml
File Path ▾ : ~/Documents/Cittasaya/Research/Papers/WSC2018/Computational/tagsDecl.xml
tagsDecl.xml
(no symbol selected)
Line 16 Col 1
XML
Unicode (UTF-8)
Unix (LF)
Last saved: 29/9/17, 1:55:28 PM
508 / 71...

```
<tagsDecl partial='false'>
 <rendition xml:id='skt' scheme='css'
  selector='seg speaker s'>font-style: roman;</rendition>
 <namespace name='http://www.tei-c.org/ns/1.0'>
  <tagUsage gi='body' occurs='1'/>
  <tagUsage gi='div' occurs='6'/>
  <tagUsage gi='sp' occurs='11'/>
  <tagUsage gi='speaker' occurs='11'/>
  <tagUsage gi='lg' occurs='194'/>
  <tagUsage gi='l' occurs='399'/>
  <tagUsage gi='seg' occurs='798'/>
  <tagUsage gi='p' occurs='0'/>
  <tagUsage gi='s' occurs='0'/>
 </namespace>
</tagsDecl>
```

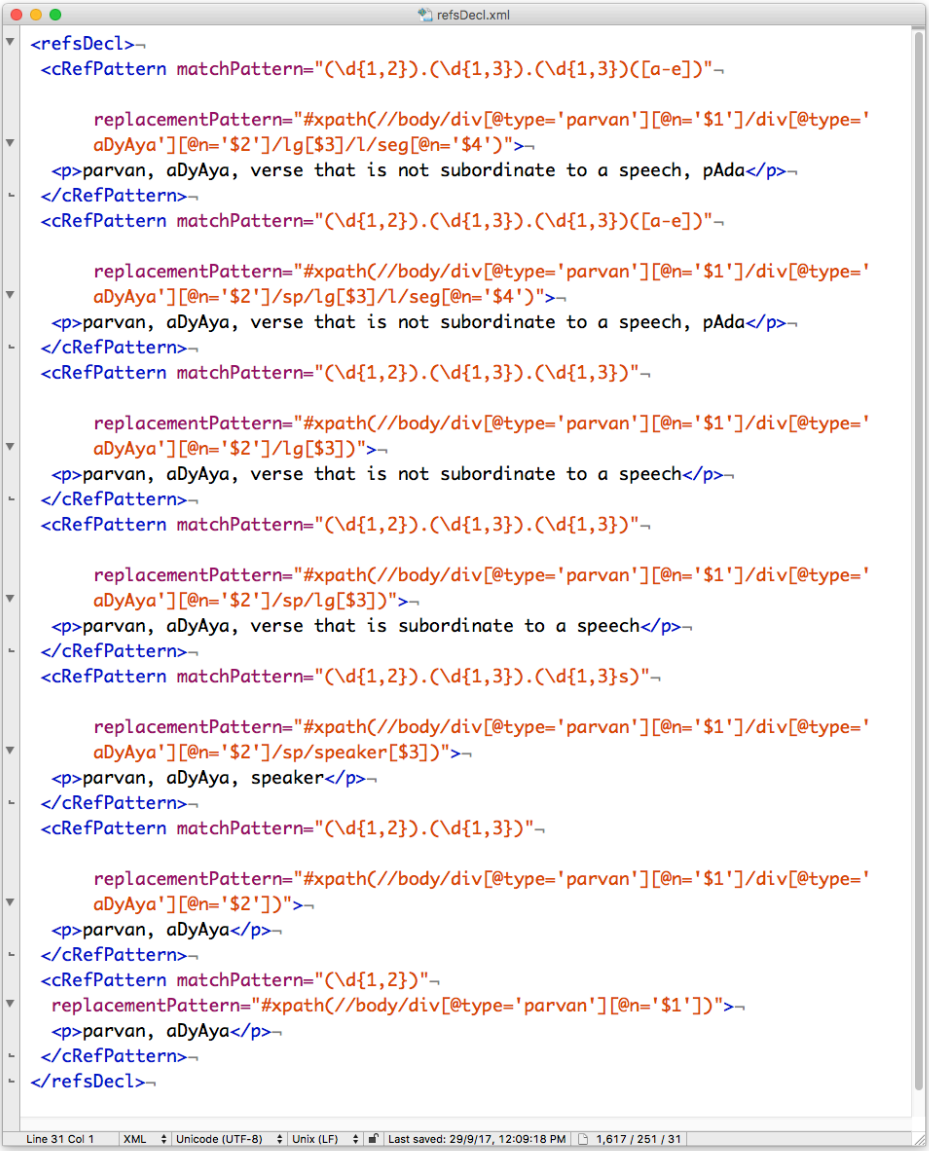
resolve canonical references to particular passages. Yet, processed in the opposite direction, from the replacement path to the match expression, the references provide a means to compose canonical references from `n` attributes.

Once a standard system of exact references to specific passages in unanalyzed continuous text has been adopted, reference to various versions of analyzed passages are easily constructed by specifying in addition the degree of analysis described in Table 4. One method of doing this in a TEI document would be to specify the degree of analysis as a value of the `ana` attribute of the `text` element. Another would be for archives to add a standard addition to the filename.

Linguistic software that produces TEI output would add elements subordinate to those containing text in the TEI document that contains the continuous text. A document that contains analyzed sandhi but no further analysis would insert each word (*pada*), including compounds (*samasta-pada*), in a `w` element. A document that contains compound analysis would insert the lexical constituents of compounds in a `w` element subordinate to the compound's `w` element. Although the types of analysis described in Table 4 do not envision tagging non-lexical morphemes such as the infix *a* and suffix *ti* in the verb *gacchati*, such morphemes would be inserted in an `m` element. TEI provides attributes that may be used for lexical and morphological analysis of each word in a `w` element. The stem of the word is made the value of the `lemma` attribute. We have chosen to make the lexical identifier a value of the `type` attribute and the morphological identifier a value of the `subtype` attribute. Figure 9 shows our TEI mark up of the sandhi analysis of the first verse of the *Bhagavadgītā*, *MBh.* 6.23.1, and Figure 10 shows our TEI mark up of the lexical and morphological analysis of the same verse. Where authors deliberately compose passages that are amenable to more than one analysis (*śleṣa*), alternative analyses — whether of verses, lines, verse quarters, prose passages, or individual words — may be analyzed in separate files where, in order to permit coordination, they may be supplied with the identical division numbers and xml:ids as their unanalyzed passages and the preferred analysis.

As a result of standardized coordination of markup and reference between Sanskrit text archives and Sanskrit computational software, HTML displays showing the unanalyzed version of a verse might be able to include a set of links to various analyzed versions for the convenience of students and scholars of Sanskrit. Conversely, displays of the results of analysis of a passage might also provide links to the unanalyzed source.

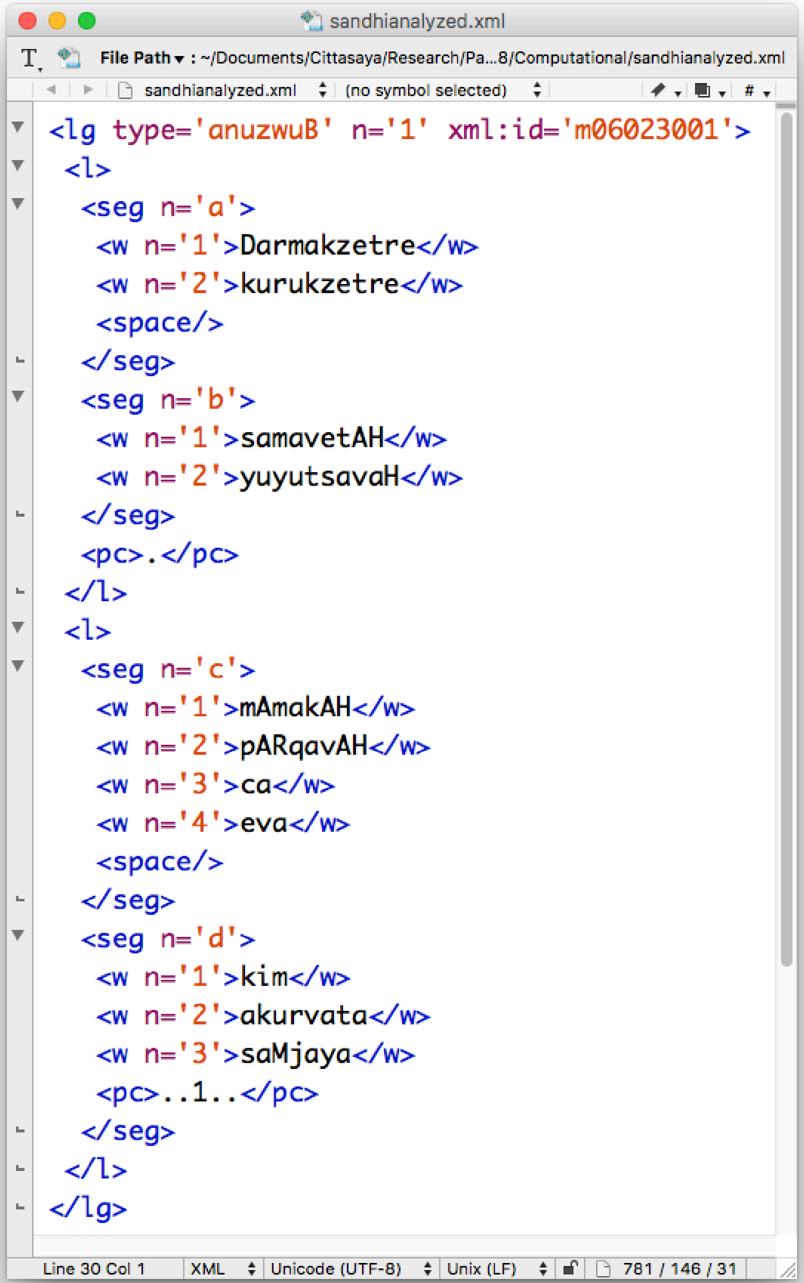
Figure 8
The `refsDecl` *element in the Sanskrit Library's TEI edition of the Svargārohaṇaparvan of the Mahābhārata*



```
<refsDecl>
 <cRefPattern matchPattern="(\d{1,2}).(\d{1,3}).(\d{1,3})([a-e])"

     replacementPattern="#xpath(//body/div[@type='parvan'][@n='$1']/div[@type='
     aDyAya'][@n='$2']/lg[$3]/l/seg[@n='$4'])">
  <p>parvan, aDyAya, verse that is not subordinate to a speech, pAda</p>
 </cRefPattern>
 <cRefPattern matchPattern="(\d{1,2}).(\d{1,3}).(\d{1,3})([a-e])"

     replacementPattern="#xpath(//body/div[@type='parvan'][@n='$1']/div[@type='
     aDyAya'][@n='$2']/sp/lg[$3]/l/seg[@n='$4'])">
  <p>parvan, aDyAya, verse that is not subordinate to a speech, pAda</p>
 </cRefPattern>
 <cRefPattern matchPattern="(\d{1,2}).(\d{1,3}).(\d{1,3})"

     replacementPattern="#xpath(//body/div[@type='parvan'][@n='$1']/div[@type='
     aDyAya'][@n='$2']/lg[$3])">
  <p>parvan, aDyAya, verse that is not subordinate to a speech</p>
 </cRefPattern>
 <cRefPattern matchPattern="(\d{1,2}).(\d{1,3}).(\d{1,3})"

     replacementPattern="#xpath(//body/div[@type='parvan'][@n='$1']/div[@type='
     aDyAya'][@n='$2']/sp/lg[$3])">
  <p>parvan, aDyAya, verse that is subordinate to a speech</p>
 </cRefPattern>
 <cRefPattern matchPattern="(\d{1,2}).(\d{1,3}).(\d{1,3}s)"

     replacementPattern="#xpath(//body/div[@type='parvan'][@n='$1']/div[@type='
     aDyAya'][@n='$2']/sp/speaker[$3])">
  <p>parvan, aDyAya, speaker</p>
 </cRefPattern>
 <cRefPattern matchPattern="(\d{1,2}).(\d{1,3})"

     replacementPattern="#xpath(//body/div[@type='parvan'][@n='$1']/div[@type='
     aDyAya'][@n='$2'])">
  <p>parvan, aDyAya</p>
 </cRefPattern>
 <cRefPattern matchPattern="(\d{1,2})"
  replacementPattern="#xpath(//body/div[@type='parvan'][@n='$1'])">
  <p>parvan, aDyAya</p>
 </cRefPattern>
</refsDecl>
```

Figure 9

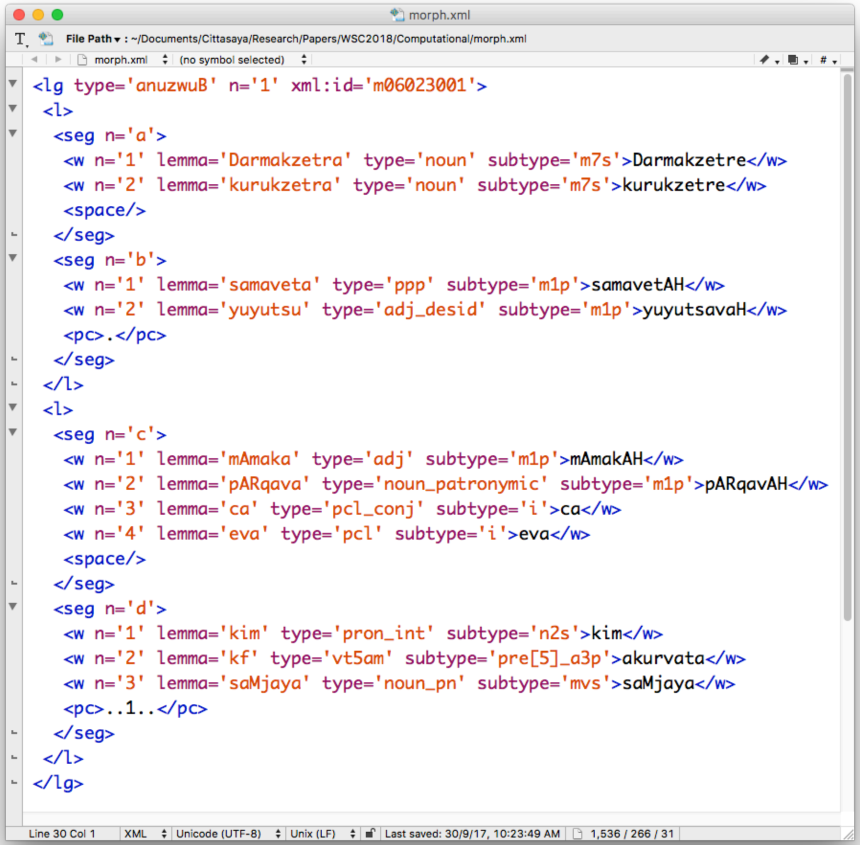
TEI mark up of the sandhi analysis of MBh. 6.23.1, the first verse of the Bhagavadgītā



```
<lg type='anuzwuB' n='1' xml:id='m06023001'>
 <l>
  <seg n='a'>
   <w n='1'>Darmakzetre</w>
   <w n='2'>kurukzetre</w>
   <space/>
  </seg>
  <seg n='b'>
   <w n='1'>samavetAH</w>
   <w n='2'>yuyutsavaH</w>
  </seg>
  <pc>.</pc>
 </l>
 <l>
  <seg n='c'>
   <w n='1'>mAmakAH</w>
   <w n='2'>pARqavAH</w>
   <w n='3'>ca</w>
   <w n='4'>eva</w>
   <space/>
  </seg>
  <seg n='d'>
   <w n='1'>kim</w>
   <w n='2'>akurvata</w>
   <w n='3'>saMjaya</w>
   <pc>..1..</pc>
  </seg>
 </l>
</lg>
```

Figure 10

TEI mark up of the lexical and morphological analysis of MBh. 6.23.1, the first verse of the Bhagavadgītā


morph.xml
File Path ▾ : ~/Documents/Cittasaya/Research/Papers/WSC2018/Computational/morph.xml
morph.xml
(no symbol selected)
Line 30 Col 1
XML
Unicode (UTF-8)
Unix (LF)
Last saved: 30/9/17, 10:23:49 AM
1,536 / 266 / 31

```
<lg type='anuzwuB' n='1' xml:id='m06023001'>
 <l>
  <seg n='a'>
   <w n='1' lemma='Darmakzetra' type='noun' subtype='m7s'>Darmakzetre</w>
   <w n='2' lemma='kurukzetra' type='noun' subtype='m7s'>kurukzetre</w>
   <space/>
  </seg>
  <seg n='b'>
   <w n='1' lemma='samaveta' type='ppp' subtype='m1p'>samavetAH</w>
   <w n='2' lemma='yuyutsu' type='adj_desid' subtype='m1p'>yuyutsavaH</w>
   <pc>.</pc>
  </seg>
 </l>
 <l>
  <seg n='c'>
   <w n='1' lemma='mAmaka' type='adj' subtype='m1p'>mAmakAH</w>
   <w n='2' lemma='pARqava' type='noun_patronymic' subtype='m1p'>pARqavAH</w>
   <w n='3' lemma='ca' type='pcl_conj' subtype='i'>ca</w>
   <w n='4' lemma='eva' type='pcl' subtype='i'>eva</w>
   <space/>
  </seg>
  <seg n='d'>
   <w n='1' lemma='kim' type='pron_int' subtype='n2s'>kim</w>
   <w n='2' lemma='kf' type='vt5am' subtype='pre[5]_a3p'>akurvata</w>
   <w n='3' lemma='saMjaya' type='noun_pn' subtype='mvs'>saMjaya</w>
   <pc>..1..</pc>
  </seg>
 </l>
</lg>
```

References

Benko, Matthew. 2000. *Understanding XML*. Tech. rep. URL: `https://faculty.darden.virginia.edu/GBUS885-00/Papers/PDFs/Benko%20-%20Understanding%20XML%20draft%20TN.pdf`.

Edgerton, Franklin. 1939. “The epic triṣṭubh and its hypermetric varieties.” *Journal of the American Oriental Society* 59.2: 159–74. DOI: `www.jstor.org/stable/594060`.

Fitzgerald, James L. 2004. “A meter-guided analysis and discussion of the dicing match of the *Sabhāparvan* of the *Mahābhārata*.”

—. 2006. “Toward a database of the non-*anuṣṭubh* verses of the *Mahābhārata*.” In: *Epics, Khilas, and Purāṇas: continuities and ruptures*. Proceedings of the Third Dubrovnik International Conference on the Sanskrit Epics and Purāṇas. Ed. by Petteri Koskikallio. Zagreb: Croatian Academy of Sciences and Arts, pp. 137–48.

—. 2009. “A preliminary study of the 681 *triṣṭubh* passages of of the *Mahābhārata*.” In: *Epic undertakings: proceedings of the 12th World Sanskrit Conference*. Ed. by Robert Goldman and Muneo Tokunaga. Delhi: Motilal Banarsidass, pp. 95–117.

Goldfarb, Charles F. 1990. *The SGML Handbook*. Oxford: Clarendon Press.

Melnad, Keshav, Pawan Goyal, and Peter M. Scharf. 2015a. “Identification of meter in Sanskrit verse.” In: *Sanskrit syntax: selected papers presented at the seminar on Sanskrit syntax and discourse structures, 13–15 June 2013, Université Paris Diderot, with a bibliography of recent research by Hans Henrich Hock*. Providence: The Sanskrit Library, pp. 325–46.

—. 2015b. “Updating Meter Identifying Tool (MIT).” In: (Bangkok, June 28–July 2, 2015). Paper presented at the 16th World Sanskrit Conference, Bankok.

Scharf, Peter M. 2014. “Linguistic issues and intelligent technological solutions in encoding Sanskrit.” *Document numérique* 16.3: 15–29.

Scharf, Peter M. and Malcolm D. Hyman. 2011. *Linguistic issues in encoding Sanskrit*. Delhi: Motilal Banarsidass.

Söhnen-Thieme, Renate. 1999. “On the composition of the Dyūtaparvan of the Mahābhārata.” In: *Composing a Tradition*. Proceedings of the First Dubrovnik International Conference on the Sanskrit Epics and Purāṇas, August 1997. Ed. by Mary Brockington and Peter Schreiner. Zagreb: Croatian Academy of Sciences and Arts, pp. 139–54.

Wikipedia contributors. 2017. *IBM Generalized Markup Language.* In: *Wikipedia: The Free Encyclopedia.* Wikipedia.

Wüstner, E., P. Buxmann, and O. Braun. 1998. “XML — The Extensible Markup Language and its Use in the Field of EDI.” In: *Handbook on architectures of information systems.* Ed. by P. Bernus, K. Mertins, and G. Schmidt. International Handbooks on Information Systems. Berlin, Heidelberg: Springer.

Zazueta, Rob. 2014. *API data exchange: XML vs. JSON.* How do you spell API? URL: `https://www.mashery.com/blog/api-data-exchange-xml-vs-json`.