A Distributed Platform for Sanskrit Processing

Pawan Goyal[1] Gérard Huet[1]
Amba Kulkarni[2] Peter Scharf[3] Ralph Bunker[4]
(1) Inria Paris-Rocquencourt
(2) Department of Sanskrit Studies, University of Hyderabad
(3) Blaise Pascal Chair, Université Paris 7, and The Sanskrit Library
(4) Maharishi University of Management, Fairfield, Iowa

Abstract

Sanskrit, the classical language of India, presents specific challenges for computational linguistics: exact phonetic transcription in writing that obscures word boundaries, rich morphology and an enormous corpus, among others. Recent international cooperation has developed innovative solutions to these problems and significant resources for linguistic research. Solutions include efficient segmenting and tagging algorithms and dependency parsers based on constraint programming. The integration of lexical resources, text archives and linguistic software is achieved by distributed interoperable Web services. Resources include a morphological tagger and tagged corpus.

Keywords: Indian language technology, Resources and annotation, Morphology and POS tagging, Parsing.

1 Introduction

Formal and computational linguistics was dominated by English at its inception and developed in subsequent decades primarily in the environment of European languages. More recently there has been a concerted effort to undertake formal linguistic analysis of a wide variety of languages, with particular interest in those with dramatically different features, and to enrich linguistic theory to account for linguistic variety. In spite of this effort, analytic structures and procedures utilized in formal linguistics remain dominated by those invented for, and most suitable for, English and other European languages. Linguistic theory remains unduly weighted in favor of European languages even as their extension to the variety of the world's languages involves undue complication thereby revealing their inadequacy in representing language universally. The inadequacy of contemporary computational methods is vividly apparent in the analysis of Sanskrit. Recent worldwide collaboration to overcome the challenges to conducting computational linguistic research on Sanskrit offers insights into methods and procedures that may be useful generally for languages that differ markedly from western European languages.

1.1 Sanskrit

Sanskrit is the primary culture-bearing language of India, with a continuous production of literature in all fields of human endeavor over the course of four millennia. Preceded by a strong oral tradition of knowledge transmission, records of written Sanskrit remain in the form of inscriptions dating back to the first century B.C.E. Extant manuscripts in Sanskrit number over 30 million - one hundred times those in Greek and Latin combined - constituting the largest cultural heritage that any civilization has produced prior to the invention of the printing press. Sanskrit works include extensive epics, subtle and intricate philosophical, mathematical, and scientific treatises, and imaginative and rich literary, poetic, and dramatic texts. The primary language of the Vedic civilization, Sanskrit developed constrained by a strong grammatical tradition stemming from the fairly complete grammar composed by Pāṇini by the fourth century B.C.E. In addition to serving as an object of study in academic institutions, the Sanskrit language persists in the recitation of hymns in daily worship and ceremonies, as the medium of instruction in centers of traditional learning, as the medium of communication in selected academic and literary journals, academic fora, and broadcasts, and as the primary language of a revivalist community near Bangalore. The language is one of the twenty-two official languages of India in which nearly fifty thousand speakers claimed fluency in the 1991 Indian census.

India developed an extraordinarily rich linguistic tradition over more than three millennia that remains under-appreciated and under-investigated. A cursory glance at the long tradition of discussion and argumentation within and between Indian sciences of phonetics (śikṣā), grammar (vyākaraṇa), logic (nyāya), ritual exegesis (karmamīmāṃsā), and literary theory (alaṃkāraśāstra) reveals that Indian linguistic traditions have much to offer contemporary linguistic theory in the areas of phonetics, morphology, syntax, and semantics.

1.2 Computational linguistic processing challenges

Since computational linguistics developed primarily in the environment of western European languages, its methods were structured and remain suited to those languages. Prior to undertaking any kind of computational analysis of Sanskrit text, one must deal with several challenges presented by features of the language that differ markedly from those of modern western European languages. In addition to the complexity introduced by lexical complements, which is relevant in these languages

as well, Sanskrit has orthographic, prosodical, and inflectional complexities not encountered in western European languages. The rich inflectional and derivational morphology of the language permits relationships that are shown positionally in western European languages to be made known by the morphology instead. As a result, the word order is much less constrained by governance structure and is free to intimate discourse structure and performative aspects of language. For the analysis of Sanskrit syntax, therefore, positional grammars, and constituency parsers which are based on them, are not very relevant, and dependency parsers are more suitable.

1.2.1 Prosody and orthography

In English, where the heritage is received in writing, and standardized spelling and printing were introduced several hundred years ago, a given morpheme is represented with a single orthography despite the fact that it has different surface phonetic representations in different contexts. For example, the past tense suffix -⟨ed⟩ is so written despite three distinct phonetic realizations in three clearly defined contexts:

/t/ e.g. dip /dɪp/, dipped /dɪpt/
/d/ e.g. boom /buːm/, boomed /buːmd/
/ɨd/ e.g. loot /luːt/, looted /luːtɨd/

In contrast, in Sanskrit, where oral tradition dominated the sphere of learning and an advanced discipline of phonetics explicitly described prosodic changes, these prosodic changes, well known by the term *sandhi*, are represented in writing. Hence the past passive participle suffix *-ta* variously realized as *ta* or *dha* depending solely upon the phonetic context, is written as follows:

/ta/ e.g. from *su* 'press', *suta* 'pressed'
/d^ha/ e.g. from *budh* 'awake', *buddha* 'awakened'

Moreover, the prosodic changes obscure word boundaries in speech, and these word boundaries are correspondingly eliminated in writing as well. For example, *vasati* 'dwells' followed by *atra* 'here' becomes वसत्यत्र (*vasatyatra*) in continuous speech. The semisyllabic Indic scripts such as Devanāgarī forestall word separation here obliterating the word boundary. Some prosodic changes, like this one, can be separated in alphabetic Roman transcription despite the sound alteration, viz. *vasaty atra*. Other prosodic changes, however, preclude word separation even in alphabetic transcription because the final sound of the preceding word and the initial sound of the following word merge in a single sound. Thus *vidyā* 'knowledge' *āpyate* 'is attained' becomes *vidyāpyate*; the single sound *ā* belongs to both words. The most difficult task in parsing a Sanskrit sentence is determining the word boundaries. Solutions to the problem have valuable ramifications for speech analysis where a similar problem is encountered in virtually all languages.

1.2.2 Inflectional morphology

In English, inflectional morphology is minimal. A present active verbal paradigm contains six slots: three for first, second and third person times two for singular and plural number. Yet the forms that fill these slots number just two, for example, *go* and *goes* for the verb 'to go'. Description of the abstract grammatical structure requires mentioning five items while description of the forms directly requires mentioning only two. Grammatical description is therefore more prolix than listing. It is nearly as efficient to describe English morphology with reference to individual forms as it is to describe it in abstract grammatical structures. (Karp et al., 1992) create a hash table of just 317,477 forms from 90,196 lexical entries, a ratio of 3.5:1. The reverse is true for Sanskrit. In Sanskrit full

verb paradigms number hundreds of unique forms in as many as sixteen hundred slots. The modest full-form lexicon created by (Scharf and Hyman, 2009a) from a lexicon of 170,000 entries numbers more than eleven million forms, a ratio of 64.7:1. It is by far more economical to describe such forms in abstract grammatical categories than it is to list them. The implication of the brevity of grammatical description of Sanskrit in comparison to listing forms is that it is misguided to attempt to describe Sanskrit grammar with reference to individual word forms. This fact was explicitly recognized by Patañjali in his massive commentary *Mahābhāśya* on Pāṇini's concise grammatical description of Sanskrit in the *Aṣṭādhyāyī*.

1.2.3 Lexical complements

In statistical analysis of a corpus of sentences, slots for lexical complements are combined. In computational linguistic processing of English it is recognized that the forms *is* and *are* belong to the same lexical unit as the forms *was* and *were*. Likewise in Sanskrit grammar, forms derived from the root *bhū* are recognized as complements of forms derived from the root *as*.[1] Because it is concerned only with individual word forms, computational linguistic processing of English treats *is* and *are* as lexical complements of each other and *was* and *were* as lexical complements of each other in the same manner as it considers the first pair as lexical complements of the second pair. Such processing treats *a* and *an* as lexical complements in the same manner. Historical linguists differ from computational linguists in their treatment of these forms. They view *is* and *are* as inflectional varieties derived from one common root, and *was* and *were* as inflectional varieties derived from another root. They consider the two roots to be lexical complements. They recognize that *a* is a phonetic variant of *an*, both derived from the word *one*. Sanskrit grammarians agree with historical linguistics here. Only semantically related roots are treated as lexical complements. Variants such as *a/an* are treated as phonetic variants, and variants such as *is/are* are treated as inflectional variants.

1.2.4 Syntax

In languages such as English where word order is strongly associated with roles, it may be reasonable to define positions in relation to roles as is done in positional grammars. Hence in an active sentence, the first position is called the subject position, the second the verb position and the third the object position. In free-word order languages such as Sanskrit, however, position does not determine role. Although certain patterns are common — such as subject, object, verb — even unmarked word order leaves some roles in indeterminate position. Alteration of the word order does not change the roles, and the position is highly influenced by discourse structure and emphasis. Phrase-structure grammars are unsuitable to describe governance structure which is more accurately described by dependency grammars.

2 Birth of a discipline

A number of projects began accumulating digitized texts in the late 1980s. The largest collection was made by the TITUS[2], which accumulated more than eighty digital Sanskrit texts within a decade. The next decades witnessed the growth of other large collections including GRETIL, which serves as a central registry of digitized Indic texts.

In the meantime a couple of projects developed digital dictionaries of Sanskrit. The Cologne Digital Sanskrit Lexicon project began by digitizing Monier Williams' A Sanskrit-English Dictionary (MW)

[1] *Aṣṭādhyāyī* 2.4.52 *aster bhū*

[2] `http://titus.uni-frankfurt.de/`

between 1994 and 1996, and followed by digitizing several other major bilingual Sanskrit dictionaries. The Digital Dictionaries of South Asia project at the University of Chicago included Apte's and MacDonell's Sanskrit-English dictionaries among its digitized Indian language dictionaries.

There were a few early isolated attempts to process Sanskrit text mechanically, such as Pushpak Bhattacharya's Sanskrit parser included as part of his M.Tech. thesis at IIT Kanpur in 1987 and Pr. Lakshmitatachar's verbal cognition generator for Bhandarkar's Sanskrit primer developed at the Academy of Sanskrit Research in Melkote in the early 1990s (Rāmapriya and Saumyanārāyaṇa, 2001).

The Indian Ministry of Communications and Information Technology provided a strong impetus for computational processing of Indian languages beginning at the turn of the century with its Technology Development for Indian Languages program (TDIL). Several periodic conferences were launched to foster research in computational linguistics, such as the International Conference on Natural Language Processing (ICON), and the Language Engineering Conference (LEC). The Akshar Bharati group developed "Anusāraka", a language accessor, for accessing texts in other languages that employed techniques inspired by Pāṇini's *Aṣṭādhyāyī* (Bharati et al., 1995). K. V. Ramakrishnacaryulu introduced natural language processing programs specifically for Sanskrit at the Rashtriya Sanskrit Vidyapeetha, Tirupati, such as "Śābdabodha Systems and Language Technology" in 2005.

In 2002, under the guidance of Amba Kulkarni, the toy morphological analyser developed at Melkote was enriched with the MW lexicon and the Dhātu-ratnākara database resulting in a wide coverage morphological analyser. Amba Kulkarni developed prototypes of several other analytic tools for Sanksrit when she began teaching specialized courses in the subject at Tirupati. Her appointment as the head of the newly formed Department of Sanskrit Studies at the University of Hyderabad, and Girish Nath Jha's appointment to the Special Center for Sanskrit Studies, J.N.U., Delhi beginning in 2002 allowed the systematic training of students in Sanskrit computational linguistics.

In 1998-1999, Peter Scharf and Ralph Bunker developed a Web-based Sanskrit reader program at Brown University called *Kramapāṭha* equiped in 2001 by Hyman with an index program and audio feature. The index program allowed Peter Scharf's *Rāmopākhyāna* to be searched by lexical and inflectional categories as well as by verbal roots, nominal stems, inflected forms, text ranges, or combinations thereof. In 2003-2004, Hyman and Peter Scharf collaborated to produce a digital edition of Whitney's roots (Whitney, 1997), that served as the source of verbal stems for their inflectional generation software. Between 2006 and 2009 Peter Scharf led the International Digital Sanskrit Library Integration project in the Classics Department at Brown University. The project created a digital Sanskrit library by integrating the texts provided by the TITUS with the digital MW dictionary of the Cologne Digital Sanskrit Lexicon project at Universität zu Köln (CDSL). The dictionary was upgraded with the assistance of Jim Funderburk and R. Chandrashekar by converting character code markers to explicit XML tags and systematically classifying and tagging additional information.

Separating linguistic processing from issues of input and display simplifies linguistic processing and also permits precise processing and display of accented dialects. Peter Scharf and Hyman designed the Sanskrit Library Phonetic encodings (SLP), described in (Scharf and Hyman, 2009b), after a thorough investigation of ancient Indian linguistic treatises, that allows all sounds represented in Vedic texts to be represented digitally. After an investigation of Sanskrit paleography, Peter Scharf initiated worldwide collaboration to extend the Unicode Standard to include 68 additional

characters required for the proper display of the ancient Vedic heritage texts of India[3]. The Unicode Standard version 5.2 incorporated the characters in two code blocks, Devanagari Extended and Vedic Extensions under South Asian Scripts on the Unicode Character Code Charts page[4]. SLP serves as the basis of a suite of transcoders that convert between standard Sanskrit Romanization of Sanskrit in Unicode, several popular Roman meta-encodings, and the Unicode pages of the major Indic scripts. Users are permitted to select their preferences for input and display at the Sanskrit Library site.

Around 2000, Gérard Huet started to develop a Sanskrit Heritage platform (SH), centered around an electronic version of the Sanskrit-French Sanskrit Heritage dictionary[5]. The dictionary was structured from the start to serve both as a computerized lexical database for morphology generation, and as a human-readable hypertext encyclopedia on Classical India (Huet, 2001, 2004). It is internally consistent in that each lexical entry is provided with hypertext links to its generating components, and it prepares the ground for syntax analysis by systematically formalizing information about complements (ākāṅkṣā).

Various tools for morpho-phonemic computation, as well as efficient structures for lexicon representation, were adapted to Sanskrit from a general computational linguistics toolkit called the Zen library, implementing general finite state transducers in functional programming style, as instances of a new relational computing paradigm called effective Eilenberg machines (Huet, 2002; Huet and Razet, 2006, 2008; Razet, 2009). The global architecture of this platform is that of interconnected Web services allowing interaction with digital libraries and other external resources. The main tool is a Sanskrit Reader, allowing segmentation (sandhi analysis), tagging, and parsing (Huet, 2003, 2005, 2006, 2007, 2009; Goyal and Huet, 2013).

In 2008 the Indian Government funded a major consortium project to develop various tools for analysis of Sanskrit text and a Sanskrit-Hindi Machine Translation System. Sanskrit scholars and computational linguists collaborated to develop the prototype of an interactive reader consisting of the 100-verse Saṅkṣepa Rāmāyaṇa. They also developed elaborate guidelines for annotating sandhi, compounds (*samāsa*) and syntactico-semantic roles (*kāraka*), and an annotated 800K corpus. Compound analysis is essential to Sanskrit parsing because 15-20% of the words in a random text are compounds and compounding is productive. A modular compound processor (Kumar, 2012) was developed that segments a given linear string into morphologically valid components (Kumar et al., 2010), determines the underlying constituency structure (Kulkarni and Kumar, 2011), and identifies the compound type (Kulkarni and Kumar, 2013). A paraphrase of the compound is then produced from the labeled constituency tree (Kumar et al., 2009). The morphological analyser previously developed by Amba Kulkarni was enhanced further by employing the head words of MW, and supplying additional derived forms before generating a full-form lexicon of 140 million words. (Kulkarni and Shukl, 2009). The constraint-based parser developed by her employing this analyser is described in section 6 below. All these tools for analysis of Sanskrit texts were used in a Sanskrit-Hindi language accessor (*anusāraka*) and a machine translation system. Comparative study of the divergences between Sanskrit and Hindi were taken up to improve the translation quality (Shukla et al., 2010) of the translator. The morphological analyser, generator, sandhi joiner and splitter, full-fledged parser, and Sanskrit-Hindi Machine Translation system were assembled in what is called *Saṁsādhanī*[6].

[3] http://www.sanskritlibrary.org
[4] http://www.unicode.org/charts
[5] http://sanskrit.inria.fr
[6] http://tdil-dc.in/san

These various efforts started coordinating themselves around 2006, with the creation of a joint team in Sanskrit computational linguistics between INRIA and Department of Sanskrit Studies, University of Hyderabad. In October 2007 the First International Sanskrit Computational Linguistics Symposium (Huet et al., 2009), organized by Gérard Huet at INRIA, allowed the presentation of the various teams and tools, and the development of cooperative software and resources. It was soon followed by the Second Symposium, organized at Brown University by Peter Scharf in May 2008, the Third one organized at University of Hyderabad by Amba Kulkarni in January 2009 (Kulkarni and Huet, 2009), the Fourth one organized at J.N.U. Delhi by Girish Nath Jha in December 2010. The Fifth one is scheduled for January 2013, organized at I.I.T. Bombay by Malhar Kulkarni.

In 2010–2011 the Sanskrit Library linked its texts to the Sanskrit Heritage reader. Each sentence in which sandhi has not been analysed is dynamically linked to the SH parser. The parser analyses the sentence using various syntactic criteria and a full-form lexicon of 700,000 forms derived from the SH lexicon of about 25,000 words. Unpenalized solutions are selected and displayed. The site allows one to examine penalized solutions and to reedit the sentence and resubmit it for further analysis. The Sanskrit Heritage site additionally allows one to submit analysed sentences for syntactic analysis by Amba Kulkarni's dependency tree parser.

The year 2012 saw a significant progress in the integration of the various tools. Pawan Goyal integrated an HTML version of the Monier-Williams dictionary as an alternative plug-in component to the Sanskrit Reader. The simultaneous invitation of Peter Scharf and Ralph Bunker to Paris eased the synchronization of tagging schemes and development of more robust protocols for interoperable Web services. In close collaboration with the other authors, Ralph Bunker developed a software-assisted human interface for morphological tagging currently being used by human annotators to prepare a tagged corpus.

3 Basic Architecture

The basic architecture of the collaborative platform is based on interactions between various Web-services. The main idea is not to have a monolithic system but various platforms, where selected components can inter-operate. These components can be software or linguistic resources. The glue between various platforms is interoperable Web-services via user interfaces and remote procedure calls.

This way of doing distributed computing has several advantages from a software engineering point of view. Firstly, Web technology gives a universal standard user interface with XHTML. Conformant HTML pages permit a uniform viewing by the various browsers offered by the various operating systems of personal computers and workstations. The technology offers automatic adaptation to the display medium, thus accommodating tablets, personal assistants, and smartphones. Furthermore, Unicode allows display in all the scripts of all the human languages. For Sanskrit, this means it is easy to display in Devanāgari script, as well as in the standard Indological romanised script with diacritics, as well as in the various transliteration schemes in use. Actually our joint distributed platform recognizes four such transliteration schemes, permitting equally easy access to scholars trained in using one scheme or another.

Secondly, developing separate components at the various sites does not commit us to any specific programming environment. The various teams at the various sites use different programming languages. There is no need for linking the executables of these various services. Finally, versioning is distributed, there is no need of synchronisation of new versions of the various services, once clear interfaces for data interchange are agreed upon (that is, we only have to agree on the XML abstract

structures defining the marshalling of interchanged data at the interfaces).

We have not felt the need to have a sophisticated orchestration of these various services. The main ingredient is remote procedure call (so-called CGI in the Web jargon). We remedy the poverty of the memory-less protocol (HTTP) by transmitting parameters summarizing the interaction history in the current session.

One common feature of our various developments is the use of UNIX platforms (Linux or MacOS) for development and server deployment. Users of the software, for instance annotators, may of course use any client operating system.

The configuration of the various platforms allows a choice between using a service as a remote process using network communication with the proper server, or alternatively to the same service run locally on the client station. This is easily achieved in Unix clients, where Web servers such as Apache or Tomcat are easy to install. Sometimes, one service is available locally as a plug-in to the server site. Thus, the Sanskrit Heritage segmenter is available as a plug-in to the University of Hyderabad Sanskrit computational platform, in order to undo sandhi in sandhied corpus. Conversely, the University of Hyderabad Sanskrit parser, using sophisticated constraint technology, may be used in the Sanskrit Heritage platform as a filter to its segmenter-tagger, superior in precision to its original crude dependency analyser.

4 Lexicon

Independent projects had previously used different lexicons as the basis for generating inflected forms used in linguistic software. The task of coordinating those lexicons with each other and with other available lexical resources presents a challenge. A current project jointly funded by the NEH and the Deutsche Forschungsgemeinschaft extends the Sanskrit Library's multidictionary interface by integrating supplements to the major bilingual dictionaries already included, and by adding specialized dictionaries, indigenous Indian monolingual dictionaries, traditional thesauri, and traditional linguistic analyses. Even after data-entry of the various lexical sources, the task of integrating them is complicated by the different conventions used by their original compilers. Compilers differ in the scripts they use, conventions of sandhi, selection of stem versus an inflected form, determination of the base form, etc. The project examines the conventions used in each lexical source, and determines ways of mapping the differences to each other.

The Sanskrit Heritage platform uses a Sanskrit-French dictionary. The desire arose to express the output of the various tools of the platform multilingually. The first step was to project the headwords of the SH dictionary onto those of the digitalised Monier-Williams dictionary. To this end, Pawan Goyal engaged in using data-mining techniques and automated translation tools to develop a protocol for the non-trivial task of mutually linking lexical resources, as described in the remainder of this section.

Let us consider the stem, *aṅga* in Sanskrit. This stem appears in two SH entries:

aṅga$_1$: membre; partie du corps; le corps en entier; la personne, la forme
En: member, part of the body the whole body, the person, the form

aṅga$_2$: affirme, confirme, ou exprime le désir ou l'impatience bien, d'accord; certes, vraiment; s'il vous plaît; vite
En: affirms, confirms, or express a desire or impatience, okay, sure, really, please, quickly

MW also has this stem listed in two different entries as:

aṅga$_1$: a particle implying attention, assent or desire, and sometimes impatience, it may be rendered, by well

aṅga$_2$: a limb of the body

So, in this example, *aṅga*$_1$ in SH should link to *aṅga*$_2$ in MW. Clearly, the matching can not be done simply based upon the name of the stem, but the concepts involved in the corresponding entry also need to be used. Thus, the headword linking problem is not trivial because of 1). **Homophony indexes:** SH and MW have their own systems of giving homophony indexes to the entries. Thus *aṅga*$_1$ in SH may correspond to either *aṅga*$_1$ or *aṅga*$_2$ in MW, and 2). **Cross-lingual resources:** While SH is a dictionary from Sanskrit to French, MW is a dictionary from Sanskrit to English. Thus, it is difficult to match the direct meanings as obtained after extracting the meaning text from both the lexicons.

4.1 Labeling MW with the lexical information

Pawan Goyal converted the XML file of the MW dictionary described in section 2 to strict XHTML by XSLT (Extensible Stylesheet Language Transformations). Each entry in the XHTML dataset was labeled with its lexical category information. This avoids the homophony problem across the lexical categories. For an example, the Sanskrit word *bhū* can be used as a noun, meaning 'earth' as well as a root meaning 'to become' and is given the homophony *bhū*$_2$ and *bhū*$_1$ in the MW XML dataset for the lexical categories corresponding to noun and root respectively. In the XHTML file, the nouns were labeled with a suffix '*-pr*[7]' and the roots were labeled with a suffix '*-dh*[8]*.C*$_n$', where *C*$_n$ denotes the corresponding class number of the verb (*gaṇa*) and varies from 1 to 10. In the case where a root entry has more than one *gaṇa*, multi-labels were given to that entry. Nominal verbs and verbs with preverb sequences were labeled with the suffix '*-dh.*Nom[9]' and '*-cpvb.*(ps,dh). *cpvb.*(ps,dh) denotes the verb with a preverb sequence 'ps' and the root 'dh'. Thus a verb '*ānī*', consisting of preverb '*ā*' and root '*nī*' was labeled as *ānī-cpvb.*(*ā*,*nī*).

4.2 Matching Dictionary Headwords

Once the MW was labeled with the lexical information, the headwords from the SH dictionary were matched with the MW headwords based on their lexical categories, which is explicit in the SH dictionary. As a rough estimate, there are approximately 16000 nouns, 600 roots, 110 nominal verbs and 1200 verbs with preverb sequences in the SH dictionary.

From the MW XHTML pages, the labels corresponding to each entry were extracted (called MW_{ent} henceforth), which, as discussed above, were marked with the lexical information. For each lexical category, the entry in the SH dictionary was looked up in the MW_{ent} and the search results were categorized in one of the following categories for further treatment: 'one to one' mapping of headword, 'many to one' mapping, 'one to many' mapping, 'many to many' mapping and 'not found'. For instance, 'one to many' mapping implies that a single headword in SH maps to many different headwords in MW and requires further disambiguation to select the desired match among the many possible matches.

While one to one mapping indicates that an entry in SH matches with one and only one entry in MW and the match results were used as it is, the cases of many to one mappings were very rare and were

[7] '*pr*' stands for *prātipadika*, the substantival base.
[8] '*dh*' stands for *dhātu*, the verbal base
[9] *dh.*Nom stands for the nominal verbs

dealt with in the same way as that of one to one mapping. The reasoning behind this design decision was the fact that MW has a wider coverage of lexicon entries. The next two cases, 'one to many' and 'many to one' mappings were problematic because these require further disambiguation to select exactly which of the many headwords in MW corresponds to the SH headword. To solve this problem, an approach based on matching the word concepts in the two dictionaries was employed.

4.3 Matching dictionary headwords using concept matching

A dictionary headword can be considered to be a concept-node in the particular ontology expressed by the dictionary and thus, the problem of matching dictionary headwords can be seen as the problem of ontology mapping. In the particular approach adopted by the authors, the problem of headword matching was translated to the problem of matching the concepts expressed by the particular concept-nodes, these headwords represent in different ontologies.

Matching the headword concepts was not so trivial because of the fact that while the SH dictionary stores the word concepts in French, the MW dictionary contains word concepts in English. To overcome this problem, the concepts from the SH dictionary were extracted and translated into English using Google Translate[10]. This ensured that we had the concepts for each entry in SH in the same language as that in MW. For a given word in SH, the concepts from the corresponding MW headwords were extracted. The concepts were preprocessed to remove stopwords such as {on, for, to, and, by} etc. These concepts were then considered similar to the Wordnet notion of 'synset'. Representing these concepts as a 'bag-of-words', the matching between the concept vectors X and Y was performed using the following function:

$$match(X, Y) = \sum_i \sum_j sim(X_i, Y_j)$$

where $sim(X_i, Y_j)$ denotes the similarity function between the strings X_i and Y_j, belonging to the concept vectors X and Y respectively. The similarity function accounted for the fact that even if the two strings have different suffixes, they represent the same concept. For example, 'rained' and 'raining' represent the same concept. The similarity function was directly proportional to the intersecting letters between X_i and Y_j and was inversely proportional to the maximum number of letters in X_i or Y_j. A threshold value was also given such that the similarity function only contributes to the matching score if its value is greater than the threshold.

Once an SH word was matched to all the possible MW headwords, the matching values were sorted and the headword with the highest match was marked as the suitable match. However, if the top two headwords have the same matching score (this also includes the case, where all the headwords obtained a score of 0.0), the SH headword along with all the MW headwords and their meanings were dumped in a text file interface, where the exact match was decided manually.

Note that the problem of solving 'one to many' mapping in the case of verbs with preverb sequences can be expressed in terms of matching the structure (ps,dh) between SH and MW and assuming that the roots have been mapped from SH to MW by following the procedure outlined above, this would not require the matching of headword meanings again. For example, consider the verb *saṃmā*$_1$ in SH, which is analysed as consisting of preverb *sam* and the verb *mā*$_1$. Once this *mā*$_1$ has been mapped to *mā*$_3$ in SH, this information can be utilized to match *saṃmā*$_1$ in SH with *saṃmā-cpvb*(*sam*,*mā*$_3$) in MW.

[10] http://translate.google.com/

5 Segmentation and Tagging

The first computational problem attacked in the framework of the Sanskrit Heritage platform was segmentation, i.e. *sandhi-viccheda*. Sandhi occurs in Sanskrit in several places. In generative morphology, it occurs internally for stem formation and affixes glueing, for instance for declension and conjugation. This so-called internal sandhi is complex, since it gives rise to long-distance retroflexion, not easily invertible by finite-state methods. On the other hand, junction of words within the sentence, as well as compound formation, uses a simpler notion of external sandhi, that may be modeled as a rational relation over words, invertible by finite-state techniques. It was thus decided to divide the task into generation of non-compound word forms in pre-processed databanks, and analysis of sentences in terms of these elementary forms.

A general toolkit for computational linguistics in functional programming style, called Zen (Huet, 2002), was first implemented. It uses in a systematic manner a notion of *decorated tries*, usable both as efficient data structures for lexicon representation, and applicative representation of automata and transducers. A new general framework for relational programming, called effective Eilenberg machines (Huet and Razet, 2006, 2008; Razet, 2009), was designed as a restriction to partial recursive relations of a mathematical model of automata theory due to Samuel Eilenberg (Eilenberg, 1974). This framework, allowing reactive programming over streams of data, proved adequate to the efficient solution of the segmentation problem in Sanskrit (Huet, 2005).

Since segmentation is directed by the inflected forms databases, presented as lemmatized segments, each segmentation solution gives rise to a canonical tagging, where each segment is tagged with the set of combinations of lexemes and morphological parameters used to generate it. The main problem to be faced was the enormous number of potential solutions of even moderately long sentences. In order to control this complexity, several devices were introduced. Firstly, the notion of binary compound was generalized into a notion of multi-segment pre-compounds, replacing a potentially exponential number of binary trees into a single linear pre-compound. Secondly, a dependency graph analysis was performed, on a restricted subset of semantic roles (mostly agent and patient). Each analysis gives rise to some penalties issued from graph-matching. Restricting the segmentation solutions to the minimal penalty ones yields a shallow parser with more manageable output. These developments have been well documented in publications (Huet, 2003, 2005, 2006, 2007, 2009), and thus will not be detailed further here.

The first effective collaboration effort between INRIA and Department of Sanskrit Studies, University of Hyderabad consisted in making the Saṁsādhanī dependency parser available as a further filtering on the Heritage engine segmenter, making it more precise, and allowing the visualization of the dependency graph, labeled with semantic roles. Conversely, the Heritage segmenter was made available in the Saṁsādhanī system as a plug-in, allowing the processing of sandhied corpus.

One important concern is that of the correctness of the computational processes used in the morpho-phonetics routines of the Heritage engine with respect to the Pāṇinian grammatical tradition, seen as a gold standard. To this effect, Pawan Goyal and Gérard Huet strived to give correspondences between the computation routines, and sequences of rewrite rules (*sūtra*s) from Pāṇini's grammar. The current state of this correspondence will appear as (Goyal and Huet, 2013).

The next concern was to start the development of a Sanskrit dependency treebank, a necessary development to benefit from statistical methods and obtain more precise analysers. To this effect, a cooperation between the Sanskrit Library effort and the Sanskrit Heritage platform was started, in view of using the platform for semi-automatic annotation of corpus by Sanskrit specialists. This more recent development is giving rise to a new interface to the segmenting tool, allowing the

synthetic visualisation of all segmentation solutions. The annotator may select any segment, and an automatic tool trims away from the forest of all solutions the ones that are inconsistent with the choice. This allows an exponential saving, and fast focusing on the intended interpretation.

6 Dependency Parsing

The parse of positional languages such as English are well expressed by constituency structure while languages like Sanskrit which are morphologically rich and to a large extent free word order are better represented by a dependency tree where the nodes represent the prātipadikas or dhātus and the edges between the nodes represent the relations between them expressed through the suffixes. Unlike other languages such as English where special efforts were put in as described in PARC (King et al., 2003), Stanford dependency manual (M. Marneffe and Manning, 2006) etc. for defining the set of relations, we are fortunate to have a well defined set of relations for Sanskrit described in traditional grammar books. All these relations have been compiled and classified under the two broad headings viz. inter sentential and intra sentential relations (Ramakrishnamacharyulu, 2009). This work provided a starting point for developing guidelines for annotation of Sanskrit texts at kāraka level and also for the development of an automatic parser for Sanskrit. These tags were further examined from the granularity point of view and a subset of 31 tags was chosen for annotation as well as for developing the parser (Kulkarni and Ramakrishnamacharyulu, 2013). The criterion used for deciding the granularity is simple. If one can tell one relation from the other purely on the basis of syntax or morphology, then the two relations were treated as distinct.

A generative grammar of any language provides rules of generation. For analysis, we require a mechanism by which we can reverse these rules. The reversal in general may not always be deterministic. This problem of non-determinism was well recognised by the mīmāṁsakas (exegesists) who proposed three conditions viz. ākāṅkṣā (expectancy), yogyatā (mutual compatibility), and sannidhi (proximity) as necessary for proper verbal cognition. The ākāṅkṣā is the syntactic expectancy a word has in order to co-relate to the other. This expectancy may be either mutual or one-way. Yogyatā helps in ruling out solutions which satisfy syntactic expectancy but which are not meaning-compatible. Sannidhi is defined as an utterance of words without any gap. The words with mutual expectancy should not be separated by other words. The condition of not allowing separation is only a necessary condition in the process of śābdabodha 'verbal cognition'. We have implemented a parser that uses two constraints viz. ākāṅkṣā and sannidhi. Implementing yogyatā requires a semantically rich lexicon. A study was undertaken to understand the structure of a Sanskrit thesaurus "Amarakośa" and comparison of its synonyms with those of Sanskrit Wordnet (Nair and Kulkarni, 2010; Nair, 2011). The implementation of yogyatā is postponed till a reasonable size of semantically rich lexicon is available.

The problem of parsing is modelled as finding a directed Tree from a Graph where the nodes correspond to the words in a sentence and the edges correspond to the relations between them. Ākāṅkṣā postulates the possible relations. Together with sannidhi it also imposes certain constraints. These constraints are solved using a generic constraint solver Minion [11]. The parses are ranked associating costs to various relations. Detailed description of the parser is available in the earlier publication (Kulkarni et al., 2010). After getting a parse, the sharing of arguments, clausal relations and the anaphora resolution indices are marked.

Let us now describe the communication between Saṁsādhanī and Heritage engine. Both platforms provide a segmenter as well as a parser. The segmenter of Saṁsādhanī (Kumar et al., 2010) works

[11] `http://minion.sourceforge.net`

in two stages. In the first stage it generates all possible segments following the sandhi rules, and in the second stage it validates the splits by a morphological analyser, throwing out almost 90% of the splits it has generated in the first stage. This results in slowing down the process. Heritage splitter on the other hand splits only if the split is morphologically valid, and thus is an efficient implementation. The parser of Saṁsādhanī is a full fledged one which handles various kinds of relations among words, sharing of arguments, and also anaphora resolution to some extent. The shallow parser of Heritage (Huet, 2007) uses mostly minimum information of transitivity of a verb as a sub-categorisation frame and models it as a graph-matching algorithm. In order to benefit from each other's work, we worked towards plugging-in these modules in each other's engine. Though the two systems were developed using different programming environments, their communication through UNIX pipes made their composition transparent. We just had to agree on the input and output specifications for the modules. Here we faced the linguistic challenges. These linguistic challenges owe to different systems being followed for the morphological analysis. Saṁsādhanī follows the Pāṇinian system while Heritage precompiles certain derivations into paradigm tables in the Western manner. This leads to differences in stems of the words in certain cases. For example, Heritage takes *aham* as the stem for the first person pronoun, while for Saṁsādhanī the stem is *asmad*. Similarly, in case of adjectives, Saṁsādhanī treats the feminine, neuter and masculine stems as different, whereas Heritage derives all the forms from the same stem. Another problem was mapping the verbal roots, since there are several classifications of verbs (*dhātupāṭha*s), and there are various views concerning verbal forms in *-yati* (roots of class 10 vs denominative verbs vs causative conjugations). The sole dhātupāṭha available in an exploitable electronic form is the *Mādhavīya dhātuvṛtti*[12]. An effort is on to link various *dhātuvṛttis* through the canonical index of verbal roots and canonical meanings (Shailaja and Kulkarni, 2013). Meanwhile, the number of primary verbal roots being a closed set, these roots were mapped manually based on their meanings. Then there is a problem of homonymy index. Saṁsādhanī uses Apte's Practical Sanskrit-Hindi dictionary. So there is a need to match the head entries of the Heritage Sanskrit-French dictionary with those of Apte's Sanskrit Hindi dictionary. In the current parser, since it does not attempt to disambiguate the words, the homonymy index is just ignored. The effort described in section 4 above may be repeated with Apte's dictionary to map the homonymy indices.

7 Annotation Tools

Syntactic research on Sanskrit is hindered by the fact that there does not exist a morphologically and syntactically tagged corpus of Sanskrit texts. Despite the large number of digitized texts now available at various websites, and the significant number that have been partially or fully sandhi-analysed, only relatively small portions of a small number of texts have been morphologically tagged. In June, Ralph Bunker, Gérard Huet, and Peter Scharf collaborated to create an interface that allows machine-assisted human-validated tagging. Sentences in digital texts in the Sanskrit Library (SL) are fed to the SH parser. The results of possible solutions are summarized in a user-friendly single-page interface that allows a Sanskrit scholar to select among presented words, stems, and morphological tags. As elements are validated, competing solutions are deprecated in the solutions summary and unique tags are automatically copied to the candidate solution. The interface also allows the scholar to edit and resubmit the sentence for re-analysis by the SH-parser, to edit, add, or delete words, stems, and tags, or to tag the sentence manually. Inflectional morphology tagsets designed independently by Peter Scharf and by Gérard Huet in categories familiar to Europeans and by Amba Kulkarni in Pāṇinian terms were mutually mapped and rendered convertible. A convenient dialogue box for tag construction ensures ease and validity of tagging. Results are saved in XML

[12] `http://sanskrit1.ccv.brown.edu/Sanskrit/Vyakarana/Dhatupatha/index2.html`

files that can be reviewed with the same interface. The project contracted IIT Bombay to engage two post-doctoral Sanskrit researchers to utilize the interface to tag digital texts.

We built the webpage for each sentence by parsing the HTML output of the SH parser and converting it to the SL format. This procedure permitted immediate integration of the SH and SL resources while work began to develop the next version of the SH parser with summarization.

The summary mode greatly improves the robustness of the SL/SH interface, specially for long complicated sentences, where the large number of potential solutions could possibly choke the server. Both the SH and SL servers are installed locally on machines running Ubuntu Linux or Mac OSX. The SH parser uses a locally installed Apache server. The SL webpages are served using a Tomcat server installed on the assistant's machine. The SL sentence webpage performs most of its work in Javascript in order to enhance responsiveness of the page. Installing the servers locally allows the assistants to tag the sentences independent of Internet access.

In the meantime, Amba Kulkarni has designed XML output of her parser for integration with the next version of the SL tagging interface, and Pawan Goyal and Gérard Huet have designed a new interactive HTML interface that summarizes the union of all solutions returned by the SH parser. This new interface presents a summary of possible sentence segmentations with each possible word positioned at the point where it begins beneath the sentence. As users validate particular words, inconsistent segmentations are discarded. When the number of parsing solutions is sufficiently low, the user can switch to explicit listing of solutions, allowing the semi-automatic selection of ambiguous morphological features.

8 Conclusions and Future Work

Decades of independent digitization of Sanskrit texts and lexical resources and development of Sanskrit linguistic resources have culminated in the collaborative efforts to develop the automated processing of Sanskrit text described in this paper. The emphasis of this collaboration over the past several months has been to build an annotation toolkit to help linguists create a morphologically tagged corpus. Due to the paucity of resources for the Sanskrit language, creating a large-scale annotated corpus is a prerequisite to the use of statistical methods for developing high-performance and robust Sanskrit text analysers. Since it is expensive to produce annotated corpora by hand, our efforts are directed towards reducing the annotation labor by building tools to permit semi-automated annotation. As discussed, the platform produced by the collaboration includes a state-of-the-art user interface with interactions between digital libraries and various text analysers.

The annotated corpus will help us explore the use of statistical methods to enhance our existing models for text analysis. Oliver Hellwig (Hellwig, 2009b,a) has already demonstrated promising results by using various statistics from inflected form n-grams to build a POS tagger. The morphologically tagged corpus under construction will allow the extended use of statistics on more abstract linguistic features. Since the corpus used as the source for annotation consists of complete texts that preserve the context of sentences within their discourse structures, the tagged corpus will be potentially helpful to pursue research towards discourse-level dependency parsing, including anaphora resolution and ellipsis determination.

References

Bharati, A., Chaitanya, V., and Sangal, R. (1995). *Natural Language Processing. A Paninian Perspective*. Prentice-Hall of India, New Delhi.

Cardona, G. (1988). *Pāṇini: his work and its traditions*. Motilal Barnasidass.

Eilenberg, S. (1974). *Automata, Languages, and Machines, volume A*. Academic Press.

Gillon, B. S. (1995). Autonomy of word formation: evidence from Classical Sanskrit. *Indian Linguistics, 56 (1-4)*, pages 15–52.

Gillon, B. S. (2009). Tagging classical Sanskrit compounds. In Kulkarni, A. and Huet, G., editors, *Sanskrit Computational Linguistics 3*, pages 98–105. Springer-Verlag LNAI 5406.

Goyal, P. and Huet, G. (2013). Completeness analysis of a Sanskrit reader. In *Proceedings, 5th International Symposium on Sanskrit Computational Linguistics*. D. K. Printworld(P) Ltd.

Hellwig, O. (2009a). Extracting dependency trees from Sanskrit texts. In Kulkarni, A. and Huet, G., editors, *Sanskrit Computational Linguistics 3*, pages 106–115. Springer-Verlag LNAI 5406.

Hellwig, O. (2009b). SanskritTagger, a stochastic lexical and POS tagger for Sanskrit. In Huet, G., Kulkarni, A., and Scharf, P., editors, *Sanskrit Computational Linguistics 1 & 2*, pages 266–277. Springer-Verlag LNAI 5402.

Huet, G. (2001). From an informal textual lexicon to a well-structured lexical database: An experiment in data reverse engineering. In *Working Conference on Reverse Engineering (WCRE'2001)*, pages 127–135. IEEE.

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

Huet, G. (2003). Towards computational processing of Sanskrit. In *International Conference on Natural Language Processing (ICON)*.

Huet, G. (2004). Design of a lexical database for Sanskrit. In *Workshop on Enhancing and Using Electronic Dictionaries, COLING 2004*. International Conference on Computational Linguistics.

Huet, G. (2005). A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger. *J. Functional Programming*, 15,4:573–614.

Huet, G. (2006). *Themes and Tasks in Old and Middle Indo-Aryan Linguistics, Eds. Bertil Tikkanen and Heinrich Hettrich*, chapter Lexicon-directed Segmentation and Tagging of Sanskrit, pages 307–325. Motilal Banarsidass, Delhi.

Huet, G. (2007). Shallow syntax analysis in Sanskrit guided by semantic nets constraints. In *Proceedings of the 2006 International Workshop on Research Issues in Digital Libraries*, New York, NY, USA. ACM.

Huet, G. (2009). Formal structure of Sanskrit text: Requirements analysis for a mechanical Sanskrit processor. In Huet, G., Kulkarni, A., and Scharf, P., editors, *Sanskrit Computational Linguistics 1 & 2*. Springer-Verlag LNAI 5402.

Huet, G., Kulkarni, A., and Scharf, P., editors (2009). *Sanskrit Computational Linguistics 1 & 2*. Springer-Verlag LNAI 5402.

Huet, G. and Razet, B. (2006). The reactive engine for modular transducers. In Futatsugi, K., Jouannaud, J.-P., and Meseguer, J., editors, *Algebra, Meaning and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday*, pages 355–374. Springer-Verlag LNCS vol. 4060.

Huet, G. and Razet, B. (2008). Computing with relational machines. *ICON'2008 tutorial,* `yquem.inria.fr/~huet/PUBLIC/Pune_tutorial.pdf`.

Joshi, S., Roodbergen, J., and Akādemī, S. (2004). *The Aṣṭādhyāyī of Pāṇini with Translation and Explanatory Notes*. Number v. 11 in The Aṣṭādhyāyī of Pāṇini. Sahitya Akademi.

Karp, D., Schabes, Y., Zaidel, M., and Egedi, D. (1992). A freely available wide coverage morphological analyser for english. In *Proceedings of Coling-92, Nantes, August 23–28, 1992*, pages 950–955.

King, T. H., Crouch, R., Riezler, S., Dalrymple, M., and Kaplan, R. (2003). The PARC 700 dependency bank.

Kiparsky, P. (2009). On the architecture of Pāṇini's grammar. In Huet, G., Kulkarni, A., and Scharf, P., editors, *Sanskrit Computational Linguistics 1 & 2*. Springer-Verlag LNAI 5402.

Kulkarni, A. and Huet, G., editors (2009). *Sanskrit Computational Linguistics 3*. Springer-Verlag LNAI 5406.

Kulkarni, A. and Kumar, A. (2011). Statistical constituency parser for Sanskrit compounds. In *Proceedings of ICON 2011*. Macmillan Advanced Research Series, Macmillan Publishers India Ltd.

Kulkarni, A. and Kumar, A. (2013). Clues from Aṣṭādhyāyī for compound type identification. In Kulkarni, M., editor, *Proceedings of the International Sanskrit Computational Linguistics Symposium*. D. K. Printworld(P) Ltd.

Kulkarni, A., Pokar, S., and Shukl, D. (2010). Designing a constraint based parser for Sanskrit. In Jha, G. N., editor, *Proceedings of the 4th International Sanskrit Computational Linguistics Symposium*. Springer-Verlag LNAI 6465.

Kulkarni, A. and Ramakrishnamacharyulu, K. V. (2013). Parsing Sanskrit texts: Some relation specific issues. In Kulkarni, M., editor, *Proceedings of the 5th International Sanskrit Computational Linguistics Symposium*. D. K. Printworld(P) Ltd.

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

Kumar, A. (2012). *An automatic Sanskrit Compound Processing*. PhD thesis, University of Hyderabad, Hyderabad.

Kumar, A., Mittal, V., and Kulkarni, A. (2010). Sanskrit compound processor. In Jha, G. N., editor, *Proceedings of the 4th International Sanskrit Computational Linguistics Symposium*. Springer-Verlag LNAI 6465.

Kumar, A., SheebaSudheer, V., and Kulkarni, A. (2009). Sanskrit compound paraphrase generator. In *Proceedings of ICON 2009*.

M. Marneffe, B. M. and Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *The fifth international conference on Language Resources and Evaluation, LREC 2006, Italy*.

Mittal, V. (2010). Automatic sanskrit segmentizer using finite state transducers. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 85–90, Uppsala, Sweden. Association for Computational Linguistics.

Nair, S. (2011). *The Knowledge Structure in Amarakośa*. PhD thesis, University of Hyderabad, Hyderabad.

Nair, S. and Kulkarni, A. (2010). The knowledge structure in Amarakośa. In Jha, G. N., editor, *Proceedings of the 4th International Sanskrit Computational Linguistics Symposium*. Springer-Verlag LNAI 6465.

Ramakrishnamacharyulu, K. V. (2009). Annotating the Sanskrit texts based on the śābdabodha systems. In *Proceedings of 3rd International Sanskrit Computational Symposium*. Springer-Verlag LNAI-5406.

Rāmapriya, B. V. and Saumyanārāyaṇa, V. (2001). Saṅgaṇakayantre nyāyaśāstrīyaśābdabodhaḥ. *Journal of Foundation Research*, VI(1–2):61–68.

Razet, B. (2009). *Machines d'Eilenberg Effectives*. PhD thesis, Université Denis Diderot (Paris 7).

Scharf, P. (2009). Levels in *Pāṇini*'s *Aṣṭādyāyī*. In Kulkarni, A. and Huet, G., editors, *Proceedings, Third International Symposium on Sanskrit Computational Linguistics*, volume LNAI 5406, pages 66–77. Springer.

Scharf, P. and Hyman, M. (2009a). Enhancing access to primary cultural heritage materials of india. In Govindaraju, V. and Setlur, S., editors, *Guide to OCR for Indic Scripts: Document Recognition and Retrieval*, pages 237–247, London; Dordrecht; Heidelberg; New York. Springer-Verlag.

Scharf, P. and Hyman, M. (2009b). *Linguistic Issues in Encoding Sanskrit*. Motilal Banarsidass, Delhi.

Shailaja, N. and Kulkarni, A. (2013). Comparative study of pāṇinīya dhātuvṛttis. In Kulkarni, M., editor, *Proceedings of the 5th International Sanskrit Computational Linguistics Symposium*. D. K. Printworld.

Sharma, A., Deshpande, K., and Padhye, D. (2008). *Kāśikā: A Commentary on Pāṇini's Grammar*. Sanskrit Academy series. Sanskrit Academy, Osmania Universtiy.

Shukla, P., Shukl, D., and Kulkarni, A. (2010). Vibhakti divergence between Sanskrit and Hindi. In Jha, G. N., editor, *Proceedings of the 4th International Sanskrit Computational Linguistics Symposium*. Springer-Verlag LNAI 6465.

Whitney, W. D. (1997). *Roots, Verb-forms and Primary Derivatives of the Sanskrit Language*. Motilal Banarsidass, Delhi. (1st edition 1885).